Finding large induced sparse subgraphs in $C_{>t}$-free graphs in quasipolynomial time

Peter Gartland[*]   Daniel Lokshtanov[†]   Marcin Pilipczuk[‡]   Michał Pilipczuk[§]

Paweł Rzążewski[¶]

February 11, 2021

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 $n^{\mathcal{O}(\log^3 n)}$ 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 $n^{\mathcal{O}(\log^2 n)}$ 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 [to appear at 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.

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 $\mathcal{C}$. The goal is to understand how the complexity of MWIS, and of related problems, changes with the class $\mathcal{C}$. 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 [20], P_6-free graphs [17], claw-free graphs [28, 22], and fork-free graphs [3, 21]. 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 [8, 7]; 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 [14]. The running time was $n^{\mathcal{O}(\log^3 n)}$, which was subsequently improved to $n^{\mathcal{O}(\log^2 n)}$ by the last three authors [25].

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 5): 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. [8, 7] 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 [20, 17], 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. [24] 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 [14, 25] 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. *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 $n^{\mathcal{O}(\log^3 n)}$ 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 $n^{\mathcal{O}(\log^2 n)}$ 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 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. In Section 7.5 we give a generalization of Theorem 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 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 [12]. 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 $\mathcal{C}_d \cap (P_t$-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 $\mathcal{C}_d \cap (P_t$-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 we can conclude the following.

Theorem 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 $n^{\mathcal{O}(\log^2 n)}$ 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. 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, 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 3. *For every pair of integers d and t, there exists an integer $k = (dt)^{\mathcal{O}(t)}$ 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 3 it follows that in the statement of Theorem 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 $\mathcal{C}$, where $\mathcal{C}$ is a fixed graph class that is closed under taking disjoint unions and induced subgraphs and is *weakly hyperfinite* [23, 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 $G \in \mathcal{C}$ 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 8.

3-Coloring. In [25], 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, obtaining the following:

Theorem 4. *For every integer t there exists an algorithm that, given an n-vertex $C_{>t}$-free graph G, runs in time $n^{\mathcal{O}(\log^3 n)}$ and verifies whether G is 3-colorable.*

2 Overview

In this section we present an overview of the proof of our main result, Theorem 1. We try to keep the description non-technical, focusing on explaining the main ideas and intuitions. Complete and formal proofs follow in subsequent sections.

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 [14, 25]; we choose to follow the exposition of [25]. The main idea is to exploit the following balanced separator theorem.

Theorem 5 (Gyárfás [18], 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 5 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 [25] 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 5. 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 $\mathcal{O}(\log^2 n)$ 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 $\mathcal{O}(\log n)$ times, the claim follows.

Now the recursion tree has depth at most n and each root-to-leaf path contains at most $\mathcal{O}(\log^2 n)$ success branches. Therefore, the total size of the recursion tree is $n^{\mathcal{O}(\log^2 n)}$, 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 [25].

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 6 (Gyárfás [18], 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 $\mathcal{B}_{u,v}^G$ 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. ⌟

Claim 3 suggests that in $C_{>t}$-free graphs, cores of connectors are objects likely to be hit by balanced separators provided by Theorem 6, similarly as in P_t-free graphs, induced paths were likely to be hit by balanced separators given by Theorem 5. 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 $\mathcal{B}_{u,v,w}^G$ as the set of all cores of all induced connectors with tips u, v, w. Let us stress here that $\mathcal{B}_{u,v,w}^G$ 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 $\mathcal{B}_{u,v,w}^G$ only once. Therefore, as each tripod has $\mathcal{O}(t)$ vertices, the buckets are again

5

of size $n^{\mathcal{O}(t)}$ 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 6 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 6. 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 $\mathcal{O}(t/n)$ 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 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 6 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 6 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 $\mathcal{O}(t)$, 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 $\mathcal{O}(t)$ 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

6

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 $|B| = |C| - |D_0| \geqslant |C| - |C|/2 = |C|/2 \geqslant n/20$.[1]

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* $\mathcal{L}_{u,v}^{G'}$ 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 $\mathcal{L}_{u,v}^{G'}$ 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 6 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 $n^{\mathcal{O}(\log^2 n)}$ leaves. In each of these leaves there is no chip, so every connected component of G' contains at most $0.95n$ vertices.

---

[1] We refer to Figure 2 in Section 5.2.1 for an illustration. Note that in the formal argument of Section 5.2.1 the component C is called C_2 and the distance between X and Y is lower bounded by $8t$, not $10t$.

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 $n^{\mathcal{O}(\log^2 n)}$ 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 $n^{\mathcal{O}(\log^3 n)}$. This concludes the description of an $n^{\mathcal{O}(\log^3 n)}$-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, so far we do not introduce CMSO_2-expressible properties. Let us call the considered problem Maximum Weight Induced d-Degenerate Graph (MWID). The algorithm for MWID that we are going to sketch is formally presented in Section 4 and Section 5, and is the subject of Theorem 11 there.

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 $[n] := \{1, \ldots, n\}$.

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.

8

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,

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

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

for each $v \in A$ and $u \in N(v) \cap W$ such that $\zeta(u) \leqslant \eta^\star(v)$, we have $u \in S^\star$ and $\eta^\star(u) < \eta^\star(v)$. $\qquad$ (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 $\mathcal{O}(\log^3 n)$ success branches, which means that following each success branch we can branch further into $n^{\mathcal{O}(1)}$ options, and the size of the recursion tree will be still $n^{\mathcal{O}(\log^3 n)}$. 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 $(\eta^\star(v) : v \in L)$; and
- for each $v \in L$, its left neighbors $L_v := \{u \in W \cap N(v) \mid \eta^\star(u) < \eta^\star(v)\}$.

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 $\mathcal{O}(\log^3 n)$ 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*

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

Note that for $y \in W$, if y gets included in the solution, then the whole set

$$M_y := \{z \in A \cap N(y) \mid \eta(z) < \zeta(y)\}$$

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

$$\Phi(\mathcal{B}^G_{u,v,w}) := \sum_{T \in \mathcal{B}^G(u,v,w)} \sum_{y \in V(T)} (d - |M_y|).$$

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 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 3. This part of the reasoning is presented in Section 6.

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 [19], 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 [26, 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. By Theorem 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 $\mathrm{type}(H, \iota) \in$ 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.

See Proposition 8 for a formal statement. 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; this part of the reasoning is presented in Section 7. 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* $\mathrm{type}_a \in$ 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 of such extensions 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 as since the algorithm will always make only $\log^{\mathcal{O}(1)} n$ success branches along each root-to-leaf path, we maintain the invariant that $|A| \leqslant \log^{\mathcal{O}(1)} n$, 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 $N(D) \cap W$ (see [Lemma 7](#)). 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](#).

3   Preliminaries

We use standard graph notation. For an positive integer p, we write $[p] := \{1, \ldots, p\}$. For a set A, $\binom{A}{p}$ denotes the set of all p-element subsets of A.

We say that two vertex subsets $X_1, X_2 \subseteq V(G)$ are *adjacent* if either $X_1 \cap X_2 \neq \emptyset$ or there is an edge $x_1 x_2 \in E(G)$, such that $x_1 \in X_1$ and $x_2 \in X_2$. Otherwise, the sets are *nonadjacent*.

For a path P, the *length* of P is the number of edges of P. For a graph G, the *radius* of G is $\min_{v \in V(G)} \max_{u \in V(G)} \mathsf{dist}(u, v)$, where $\mathsf{dist}(u, v)$ denotes the *distance* between u and v, i.e., the length of a shortest u-v-path in G.

3.1   Graph minors

Let H be a graph. A *minor model* of H in a graph G is a mapping η that assigns to each $v \in V(H)$ a connected subgraph $\eta(v)$ of G so that:

- the subgraphs $\{\eta(v) : v \in V(H)\}$ are pairwise disjoint; and

- for every edge $v_1 v_2 \in E(H)$, there is an edge in G with one endpoint in $\eta(v_1)$ and the other in $\eta(v_2)$.

Such a minor model η has *depth* t if every subgraph $\eta(v)$ has radius at most t. We say that G contains H as a (depth-t) minor if there is a (depth-t) minor model of H in G.

A *topological minor model* of H in G is a mapping ψ that assigns to each vertex $v \in V(H)$ a vertex $\psi(v)$ in G and to each edge $e \in E(H)$ a path $\psi(e)$ in G so that

- vertices $\psi(v)$ are pairwise different for different $v \in V(H)$;

- for each edge $v_1 v_2 \in E(H)$, the path $\psi(v_1 v_2)$ has endpoints $\psi(v_1)$ and $\psi(v_2)$ and does not pass through any of the vertices of $\{\psi(v) \colon v \in V(H)\}$ other than $\psi(v_1)$ and $\psi(v_2)$; and

- paths $\{\psi(e) \colon e \in E(H)\}$ are pairwise disjoint apart from possibly sharing endpoints.

The vertices $\{\psi(v) \colon v \in V(H)\}$ are the *roots* of the topological minor model ψ. We say that ψ has *depth* t if each path $\psi(e)$ for $e \in E(H)$ has length at most $2t + 1$. We say that G contains H as a (depth-t) topological minor if there is a depth-t topological minor model of H in G. It is easy to see that if G contains H as a depth-t topological minor, then it also contains H as a depth-t minor.

3.2 Treewidth and tree decompositions

A *tree decomposition* of a graph G is a pair (T, β), where T is a tree and β is a function that maps every vertex of T to a subset of $V(G)$, such that the following properties hold:

- every edge of G is contained in $\beta(a)$ for some $a \in V(T)$, and

- for every $v \in V(G)$, the set $\{a \in V(T) \mid v \in \beta(a)\}$ is nonempty and induces a subtree of T.

The sets $\beta(a)$ for $a \in V(T)$ are called the *bags* of the decomposition (T, β). The *width* of the decomposition (T, β) is $\max_{a \in V(T)} |\beta(a)| - 1$ and the *treewidth* of a graph is the minimum possible width of its decomposition.

We will need the following well-known observation about graphs of bounded treewidth.

Lemma 7 (see Lemma 7.19 of [11]). *Let H be a graph of treewidth less than k and let $A \subseteq V(H)$. Then there exists a set $X \subseteq V(H)$ of size at most k such that every connected component of $H - X$ has at most $|A|/2$ vertices of A.*

3.3 MSO_2 and MSO_2 types

We assume the incidence encoding of graphs as relational structures: a graph G is encoded as a relational structure whose universe consists of vertices and edges of G (each distinguishable by a unary predicate), and there is one binary incidence relation binding each edge with its two endpoints. With this representation, the MSO_2 logic on graphs is the standard MSO logic on relational structures as above, which boils down to allowing the formulas to use vertex variables, edge variables, vertex set variables, and edge set variables, together with the ability of quantifying over them. For a positive integer p, the $\mathsf{C}_{\leqslant p}\mathsf{MSO}_2$ logic extends MSO_2 by allowing atomic formulas of the form $|X| \equiv a \bmod m$, where X is a set variable and $0 \leqslant a < m \leqslant p$ are integers; denote $\mathsf{CMSO}_2 := \bigcup_{p > 0} \mathsf{C}_{\leqslant p}\mathsf{MSO}_2$. The *quantifier rank* of a formula is the maximum number of nested quantifiers in it.

For a finite set Λ of labels, a Λ-*boundaried graph* is a pair (G, ι) where G is a graph and $\iota \colon B \to \Lambda$ is an injective function for some $B \subseteq V(G)$; then $B = \mathrm{dom}\,\iota$ is called the *boundary* of (G, ι). A k-*boundaried graph* is a shorthand for a $[k]$-boundaried graph, where we denote $[k] = \{1, \ldots, k\}$. For $v \in B$, $\iota(v)$ is the *label* of v. We define two operations on Λ-boundaried graphs.

If (G_1, ι_1) and (G_2, ι_2) are two Λ-boundaried graphs, then the result of *gluing* them is the boundaried graph $(G_1, \iota_1) \oplus_\Lambda (G_2, \iota_2)$ that is obtained from the disjoint union of (G_1, ι_1) and (G_2, ι_2) by identifying vertices of the same label (so that the resulting labelling is again injective).

If (G, ι) is a Λ-boundaried graph and $l \in \Lambda$, then the result of *forgetting* l is the Λ-boundaried graph $(G, \iota|_{\iota^{-1}(\Lambda - \{l\})})$. That is, if l belongs to the range of ι, we remove the label from the corresponding vertex (but we keep this vertex in G).

By $\mathsf{C}_{\leqslant p}\mathsf{MSO}_2$ on Λ-boundaried graphs we mean the $\mathsf{C}_{\leqslant p}\mathsf{MSO}_2$ logic over graphs enriched with $|\Lambda|$ unary predicates: for each label $l \in \Lambda$ we have a unary predicate that selects the only vertex with label l, if existent. Effectively, this boils down to the possibility of using elements of the boundary as constants in $\mathsf{C}_{\leqslant p}\mathsf{MSO}_2$ formulas.

The following folklore proposition explains the compositionality properties of CMSO_2 on boundaried graphs. The statement and the proof is standard, see e.g. [15, Lemma 6.1], so we only sketch it.

Proposition 8. *For every triple of integers k, p, q, there exists a finite set $\mathsf{Types}^{k,p,q}$ and a function that assigns to every k-boundaried graph (G, ι) a type $\mathsf{type}^{k,p,q}(G, \iota) \in \mathsf{Types}^{k,p,q}$ such that the following holds:*

1. *The types of isomorphic graphs are the same.*

2. *For every $\mathsf{C}_{\leqslant p}\mathsf{MSO}_2$ sentence φ on k-boundaried graphs, whether (G, ι) satisfies φ depends only on the type $\mathsf{type}^{k,p,q}(G, \iota)$, where q is the quantifier rank of φ. More precisely, there exists a subset $\mathsf{Types}^{k,p,q}[\varphi] \subseteq \mathsf{Types}^{k,p,q}$ such that for every k-boundaried graph (G, ι) we have*

$$(G, \iota) \models \varphi \qquad \text{if and only if} \qquad \mathsf{type}^{k,p,q}(G, \iota) \in \mathsf{Types}^{k,p,q}[\varphi].$$

3. *The types of ingredients determine the type of the result of the gluing operation. More precisely, for every two types $\tau_1, \tau_2 \in \mathsf{Types}^{k,p,q}$ there exists a type $\tau_1 \oplus_{k,p,q} \tau_2$ such that for every two k-boundaried graphs $(G_1, \iota_1), (G_2, \iota_2)$, if $\mathsf{type}^{k,p,q}(G_i, \iota_i) = \tau_i$ for $i = 1, 2$, then*

$$\mathsf{type}^{k,p,q}((G_1, \iota_1) \oplus_{[k]} (G_2, \iota_2)) = \tau_1 \oplus_{k,p,q} \tau_2.$$

   Also, the operation $\oplus_{k,p,q}$ is associative and commutative.

4. *The type of the ingredient determines the type of the result of the forget label operation. More precisely, for every type $\tau \in \mathsf{Types}^{k,p,q}$ and $l \in [k]$ there exists a type $\tau_{\neg l}$ such that for every k-boundaried graph (G, ι), if $\mathsf{type}^{k,p,q}(G, \iota) = \tau$ and (G', ι') is the result of forgetting l in (G, ι), then*

$$\mathsf{type}^{k,p,q}(G', \iota') = \tau_{\neg l}.$$

PROOF (SKETCH). It is well-known that there are only finitely many syntactically non-equivalent $\mathsf{C}_{\leqslant p}\mathsf{MSO}_2$ sentence over k-boundaried graphs and of quantifier rank at most q, so let $\mathsf{Sentences}^{k,p,q}$ be a set containing one such sentence from each equivalence class. We set $\mathsf{Types}^{k,p,q}$ as the power set (set of all subsets) of $\mathsf{Sentences}^{k,p,q}$. To each k-boundaried graph (G, ι) we define $\mathsf{type}^{k,p,q}(G, \iota) \subseteq \mathsf{Sentences}^{k,p,q}$ as the set of all sentences $\psi \in \mathsf{Sentences}^{k,p,q}$ satisfied in (G, ι). Thus, whether (G, ι) satisfies φ can be decided by verifying whether $\mathsf{type}^{k,p,q}(G, \iota)$ contains a sentence that is syntactically equivalent to φ. The remaining two assertions — about compositionality of the gluing and the forget label operations — follow from a standard argument using Ehrenfeucht-Fraïssé games. $\qquad\square$

In our algorithms, we shall work with a fixed CMSO_2 sentence φ over graphs with treewidth upper-bounded by a fixed constant k. Note that φ belongs to $\mathsf{C}_{\leqslant p}\mathsf{MSO}_2$ for a fixed constant p, and the quantifier rank of φ is a fixed constant q. Hence, when working in k-boundaried graphs, whether φ is satisfied in a k-boundaried graph (G, ι) can be read from its type $\mathsf{type}^{k,p,q}(G, \iota)$. To facilitate the computation of types,

we shall assume that our algorithms have a hard-coded set of types $\mathsf{Types}^{k,p,q}$, together with the subset $\mathsf{Types}^{k,p,q}[\varphi] \subseteq \mathsf{Types}^{k,p,q}$ and functions

$$(\tau_1, \tau_2) \mapsto \tau_1 \oplus_{k,p,q} \tau_2 \qquad \text{and} \qquad (\tau, l) \mapsto \tau_{\neg l},$$

as described in Proposition 8. Also, the algorithms have hard-coded the types of all k-boundaried graphs with $\mathcal{O}(k)$ vertices.

We will need also a simple observation that $\mathsf{CMSO_2}$ types preserve connectivity properties, as being in the same connected component can be easily expressed in $\mathsf{CMSO_2}$.

Lemma 9. *Fix integers $k \geqslant 0, p \geqslant 0$, and $q \geqslant 4$, and suppose that (G_1, ι_1) and (G_2, ι_2) are two k-boundaried graphs with $\mathsf{type}^{k,p,q}(G_1, \iota_1) = \mathsf{type}^{k,p,q}(G_2, \iota_2)$. Then the ranges of ι_1 and ι_2 are equal. Furthermore, for every two pairs $(u_1, u_2), (v_1, v_2) \in V(G_1) \times V(G_2)$ with $\iota_1(u_1) = \iota_2(u_2)$ and $\iota_1(v_1) = \iota_2(v_2)$, vertices u_1 and v_1 are in the same connected component of G_1 if and only if vertices u_2 and v_2 are in the same connected component of G_2.*

PROOF. That the ranges of ι_1 and ι_2 are equal is clear: G_1 and G_2 satisfy the same sentences of the form "there exists a vertex with label $i \in [k]$". The assertion about having the same connectivity between boundary vertices follows from an analogous argument, supplied with the observation that being in the same connected component can be expressed by an $\mathsf{MSO_2}$ formula of quantifier rank four:

$$\begin{aligned}
\mathsf{Connected}(x, y) \quad = \quad & \neg \exists_{A \subseteq V(G)} (x \in A) \wedge (y \notin A) \wedge \\
& \big(\forall_{e \in E(G)} \forall_{u \in V(G)} \forall_{v \in V(G)} (\mathrm{inc}(u, e) \wedge \mathrm{inc}(v, e)) \Rightarrow ((u \in A) \Leftrightarrow (v \in A)) \big). \quad \square
\end{aligned}$$

In our proofs, we will need to keep track of the exact value of the treewidth of a constructed induced subgraph of the given graph. For this, we use the following observation.

Lemma 10. *For every integer k there exists an $\mathsf{MSO_2}$ sentence $\varphi_{\mathrm{tw}<k}$ such that $G \models \varphi_{\mathrm{tw}<k}$ if and only if the treewidth of G is less than k.*

PROOF. Let $\mathcal{F}_{\mathrm{tw}<k}$ be the set of all minor-minimal graphs of treewidth at least k. By the Robertson-Seymour Theorem [27], $\mathcal{F}_{\mathrm{tw}<k}$ is finite. Define $\varphi_{\mathrm{tw}<k}$ to be the conjunction, over all $H \in \mathcal{F}_{\mathrm{tw}<k}$, of sentences asserting that H is not a minor of G. Note here that it is straightforward to express in $\mathsf{MSO_2}$ that a fixed graph H is a minor of a given graph G. $\square$

4 Branching framework

In this section we prove Theorem 11 stated below, which is a weaker variant of Theorem 1 that does not speak about $\mathsf{CMSO_2}$ properties.

Theorem 11. *Fix a pair of integers d and t. 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 $n^{\mathcal{O}(\log^3 n)}$ finds a subset of vertices S such that $G[S]$ is d-degenerate and, subject to the above, $\mathfrak{w}(S)$ is maximum possible. The running time can be improved to $n^{\mathcal{O}(\log^2 n)}$ if G is P_t-free.*

We present the strategy for our branching algorithm in a quite general fashion, so that it can be reused later for the proof of Theorem 1. For the description, we fix a positive integer d that is the bound on the degeneracy of the sought induced subgraph.

We will rely on the following characterization of graphs of bounded degeneracy. As the statement of Lemma 12 is a bit non-standard, we include a sketch of a proof.

Lemma 12. *A graph G has degeneracy at most d if and only if there exists a function $\eta\colon V(G) \to [|V(G)|]$ such that for every $uv \in E(G)$ we have $\eta(u) \neq \eta(v)$, and for every $v \in V(G)$, the set $\{u \in N_G(v) \mid \eta(u) < \eta(v)\}$ has size at most d.*

PROOF (SKETCH). If G is d-degenerate, then we can construct an ordering η by removing a vertex v with minimum degree, inductively ordering the vertices of $G - v$, and appending v at the last position.

On the other hand, consider any subgraph G' of G and let η' be the ordering η restricted to the vertices of G'. Let $v \in V(G')$ be a vertex at the last position in η'; if there is more than one such vertex, we choose one arbitrarily. Note that all neighbors of v in G' precede it in η', so there are at most d of them. □

Function η as in Lemma 12 shall be called a *d-degeneracy ordering* of G, and the value $\eta(v)$ is the *position* of v. We remark here that, contrary to the usual definition, we do not require η to be injective, but only to give different positions to endpoints of a single edge. Henceforth, we say that a function $\eta\colon Z \to [|V(G)|]$ for some $Z \subseteq V(G)$ is *edge-injective* if $\eta(u) \neq \eta(v)$ for every $uv \in E(G[Z])$.

Let G be an n-vertex $C_{>t}$-free graph and $\mathfrak{w}\colon V(G) \to \mathbb{N}$ be a weight function. Without loss of generality, assume $t \geqslant 6$. Fix a subset $S^* \subseteq V(G)$ such that $G[S^*]$ has degeneracy at most d and fix a d-degeneracy ordering $\eta^*\colon S^* \to [|S^*|]$ of $G[S^*]$. We think of S^* as of the intended optimum solution.

4.1 Recursion structure

A *subproblem* $\mathcal{R}$ consists of

- Two disjoint vertex sets $A^{\mathcal{R}}, X^{\mathcal{R}} \subseteq V(G)$. We additionally denote $R^{\mathcal{R}} := V(G) - (A^{\mathcal{R}} \cup X^{\mathcal{R}})$ and call $R^{\mathcal{R}}$ the *free vertices*.

- An integer $\ell^{\mathcal{R}} \geqslant 0$, called the *level* of the subproblem.

- A set $W^{\mathcal{R}} \subseteq R^{\mathcal{R}}$ that is nonadjacent to $R^{\mathcal{R}} - W^{\mathcal{R}}$ and is of size less than $0.99^{-\ell^{\mathcal{R}}}$. The vertices of $W^{\mathcal{R}}$ are called the *active vertices*;

- An edge-injective function $\eta^{\mathcal{R}}\colon A^{\mathcal{R}} \to [n]$.

- A function $\zeta^{\mathcal{R}}\colon W^{\mathcal{R}} \to [n+1]$.

The superscript can be omitted if it is clear from the context.

In our recursive branching algorithms, one recursive call will treat one subproblem. The intended meaning of the components of the subproblem is as follows:

- $A^{\mathcal{R}}$ corresponds to the vertices already decided to be in the partial solution. The value $\eta^{\mathcal{R}}(v)$ for $v \in A^{\mathcal{R}}$ indicates the final position in the degeneracy ordering.

- $X^{\mathcal{R}}$ corresponds to the vertices already decided to be not in the partial solution.

- $R^{\mathcal{R}}$ are vertices yet to be decided.

Relying on some limited dependence between the connected components of $G[R^{\mathcal{R}}]$ in the studied problems, one recursive call focuses only on a group of these components, whose union of vertex sets is denoted by $W^{\mathcal{R}}$, and seeks for a best way to extend the current partial solution into $W^{\mathcal{R}}$. The integer $\zeta^{\mathcal{R}}(v)$ for $v \in W^{\mathcal{R}}$ indicates the minimum position at which the vertex v can be placed in the final degeneracy ordering.

This intuition motivates the following definition. A subproblem $\mathcal{R}$ is *lucky* if

- $A^{\mathcal{R}} \subseteq S^*$, $X^{\mathcal{R}} \cap S^* = \emptyset$;

- $\eta^{\mathcal{R}} = \eta^*|_{A^{\mathcal{R}}}$;

- for every $u \in S^* \cap W^{\mathcal{R}}$ we have $\zeta^{\mathcal{R}}(u) \leqslant \eta^*(u)$; and

- for every $u \in W^{\mathcal{R}}$ and every $v \in N_G(u) \cap A^{\mathcal{R}}$, if $\zeta^{\mathcal{R}}(u) \leqslant \eta^*(v)$, then $u \in S^*$ and $\eta^*(u) < \eta^*(v)$.

We will also need the following notion: for an edge-injective function $\eta \colon Z \to [n]$ for some $Z \subseteq V(G)$, $v \in V(G)$, and an integer p, the *quota* of v in η at position p is defined as

$$\gamma(\eta, v, p) = d - |\{u \in N_G(v) \cap Z \mid \eta(u) < p\}|.$$

If $v \in Z$, then $\gamma(\eta, v)$ is a shorthand for $\gamma(\eta, v, \eta(v))$. Intuitively, $\gamma(\eta, v)$ measures the number of "free slots" for the neighbors of v that precede it in η. Note that an edge-injective function $\eta : V(G) \to [n]$ is a d-degeneracy ordering of G if and only if $\gamma(\eta, v) \geqslant 0$ for every $v \in V(G)$. Furthermore, if $\eta' : Z' \to [n]$ is an extension of $\eta \colon Z \to [n]$ to some $Z' \supseteq Z$, then for every $v \in V(G)$ and $p \in [n]$, it holds that $\gamma(\eta, v, p) \geqslant \gamma(\eta', v, p)$.

Extending. For subproblems $\mathcal{R}_1$ and $\mathcal{R}_2$, we say that

- $\mathcal{R}_2$ *extends* $\mathcal{R}_1$ if

  - $A^{\mathcal{R}_1} \subseteq A^{\mathcal{R}_2}$, $X^{\mathcal{R}_1} \subseteq X^{\mathcal{R}_2}$ (and hence $R^{\mathcal{R}_1} \supseteq R^{\mathcal{R}_2}$);
  - $A^{\mathcal{R}_2} - A^{\mathcal{R}_1} \subseteq W^{\mathcal{R}_1}$ and $X^{\mathcal{R}_2} - X^{\mathcal{R}_1} \subseteq W^{\mathcal{R}_1}$;
  - $W^{\mathcal{R}_1} \supseteq W^{\mathcal{R}_2}$;
  - $\eta^{\mathcal{R}_1} = \eta^{\mathcal{R}_2}|_{A^{\mathcal{R}_1}}$;
  - for every $v \in W^{\mathcal{R}_2}$ we have $\zeta^{\mathcal{R}_2}(v) \geqslant \zeta^{\mathcal{R}_1}(v)$; and
  - for every $v \in A^{\mathcal{R}_2} \cap W^{\mathcal{R}_1}$ we have $\eta^{\mathcal{R}_2}(v) \geqslant \zeta^{\mathcal{R}_1}(v)$.

- $\mathcal{R}_2$ *extends* $\mathcal{R}_1$ *completely* if additionally $(A^{\mathcal{R}_2} - A^{\mathcal{R}_1}) \cup (X^{\mathcal{R}_2} - X^{\mathcal{R}_1}) = W^{\mathcal{R}_1}$ and $\ell^{\mathcal{R}_2} = 0$ (in particular, $W^{\mathcal{R}_2} = \emptyset$).

One easy way to extend a subproblem $\mathcal{R}$ is to select a set $Z \subseteq W^{\mathcal{R}}$ and move it to X; formally, the operation of *deleting* Z creates a new subproblem $\mathcal{R}'$ extending $\mathcal{R}$ by keeping all the ingredients the same, except for $X^{\mathcal{R}'} = X^{\mathcal{R}} \cup Z$, $W^{\mathcal{R}'} = W^{\mathcal{R}} - Z$, and $\zeta^{\mathcal{R}'} = \zeta^{\mathcal{R}}|_{W^{\mathcal{R}'}}$. Clearly, if $\mathcal{R}$ is lucky and $Z \cap S^* = \emptyset$, then $\mathcal{R}'$ is lucky as well.

A second (a bit more complicated way) to extend a subproblem $\mathcal{R}$ is the following. First, select a set $Z \subseteq W^{\mathcal{R}}$. Second, select an edge-injective function $\eta \colon A^{\mathcal{R}} \cup Z \to [n]$ extending $\eta^{\mathcal{R}}$ such that $\eta(u) \geqslant \zeta^{\mathcal{R}}(u)$ for every $u \in Z$; we henceforth call such a function a *position guess* for Z and $\mathcal{R}$. Third, for every $u \in Z$ select a set $D_u \subseteq W^{\mathcal{R}} - Z$ of size at most $\gamma(\eta, u)$. The family $(D_u)_{u \in Z}$ is called the *left neighbor guess* for Z, $\mathcal{R}$, and η. Finally, define the operation of *taking Z at positions η with left neighbors $(D_u)_{u \in Z}$* as constructing a new subproblem $\mathcal{R}'$ created from $\mathcal{R}$ by keeping all the ingredients the same, except for $A^{\mathcal{R}'} = A^{\mathcal{R}} \cup Z$, $W^{\mathcal{R}'} = W^{\mathcal{R}} - Z$, $\eta^{\mathcal{R}'} = \eta$, and, for every $w \in W^{\mathcal{R}'}$, taking

$$\zeta^{\mathcal{R}'}(w) = \max(\zeta^{\mathcal{R}}(w), \max\{1 + \eta(u) \colon u \in Z \wedge w \in N_G(u) - D_u\}).$$

We have the following immediate observation:

Lemma 13. *Let $\mathcal{R}$ be lucky and $Z \subseteq S^* \cap W^{\mathcal{R}}$. Then $\eta|_{A^{\mathcal{R}} \cup Z} := \eta^*|_{A^{\mathcal{R}} \cup Z}$ is a position guess for $\mathcal{R}$ and Z and $(D_u)_{u \in Z}$ defined as*

$$D_u = \{w \in N_G(u) \cap S^* \cap W^{\mathcal{R}} \mid \eta^*(w) < \eta^*(u)\}$$

is a left neighbor guess for $\mathcal{R}$, Z, and η. Furthermore, the result of taking Z at positions η and left neighbors $(D_u)_{u \in Z}$ is lucky as well.

Filtering. We will need a simple filtering step. Consider a subproblem $\mathcal{R}$.

- A vertex $v \in A^{\mathcal{R}}$ is *offending* if there are more than d vertices $u \in N_G(v)$ such that either $u \in A^{\mathcal{R}}$ and $\eta^{\mathcal{R}}(u) < \eta^{\mathcal{R}}(v)$, or $u \in W^{\mathcal{R}}$ and $\zeta^{\mathcal{R}}(u) \leqslant \eta^{\mathcal{R}}(v)$.

- A vertex $v \in W^{\mathcal{R}}$ is *offending* if $\zeta^{\mathcal{R}}(v) > n$ or $\gamma(\eta^{\mathcal{R}}, v, \zeta^{\mathcal{R}}(v)) < 0$, where γ is defined w.r.t. $Z = A^{\mathcal{R}}$.

- The subproblem $\mathcal{R}$ is *clean* if there are no offending vertices.

We observe the following.

Lemma 14. *If in a subproblem $\mathcal{R}$ there is an offending vertex $v \in A^{\mathcal{R}}$ or $v \in W^{\mathcal{R}} \cap S^*$, then $\mathcal{R}$ is not lucky.*

PROOF. For contradiction, suppose $\mathcal{R}$ is lucky. Assume first there is an offending vertex $v \in W^{\mathcal{R}} \cap S^*$. Since $\mathcal{R}$ is lucky, we have $\zeta^{\mathcal{R}}(v) \leqslant \eta^*(v) \leqslant n$. Furthermore,

$$
\begin{aligned}
d &\geqslant |\{u \in N_G(v) \cap S^* \mid \eta^*(u) < \eta^*(v)\}| \\
&\geqslant |\{u \in N_G(v) \cap A^{\mathcal{R}} \mid \eta^*(u) < \eta^*(v)\}| \\
&\geqslant |\{u \in N_G(v) \cap A^{\mathcal{R}} \mid \eta^*(u) < \zeta^{\mathcal{R}}(v)\}| \\
&= d - \gamma(\eta^{\mathcal{R}}, v, \zeta^{\mathcal{R}}(v)),
\end{aligned}
$$

which implies that $\gamma(\eta^{\mathcal{R}}, v, \zeta^{\mathcal{R}}(v)) \geqslant 0$. This contradicts the assumption that v is offending.

Assume now that there is an offending vertex $v \in A^{\mathcal{R}}$. Note that if $u \in N_G(v) \cap A^{\mathcal{R}}$ satisfies $\eta^{\mathcal{R}}(u) < \eta^{\mathcal{R}}(v)$, then $u \in S^*$ and $\eta^*(u) < \eta^*(v)$. Further, if $u \in N_G(v) \cap W^{\mathcal{R}}$ satisfies $\zeta^{\mathcal{R}}(u) \leqslant \eta^{\mathcal{R}}(v)$, then by the last item of the definition of being lucky we infer that $u \in S^*$ and $\eta^*(u) < \eta^*(v)$. Consequently, if v is offending, then $|\{u \in N_G(v) \cap S^* \mid \eta^*(u) < \eta^*(v)\}| > d$, a contradiction. □

A *filtering step* applied to a subproblem $\mathcal{R}$ creates a subproblem $\mathcal{R}'$ that is the result of deleting all offending vertices $v \in W^{\mathcal{R}}$ from $\mathcal{R}$. By Lemma 14 we infer that if $\mathcal{R}$ is lucky, then $\mathcal{R}'$ is lucky, too.

4.2 Subproblem tree

Note that subproblems of level 0 have necessarily $W^{\mathcal{R}} = \emptyset$. These subproblems shall correspond to the leaves of the recursion. Let us now proceed to the description of the recursion tree.

A *subproblem tree* is a rooted tree where every node x is labelled with a subproblem $\mathcal{R}(x)$ and is one of the following five types: leaf node, filter node, split node, branch node, and free node. We require that the root of the tree is labelled with a clean subproblem and that for every x and its child y, $\mathcal{R}(y)$ extends $\mathcal{R}(x)$.

For brevity, we say that x is clean if $\mathcal{R}(x)$ is clean. Similarly, we say that y (completely) extends x if $\mathcal{R}(y)$ (completely) extends $\mathcal{R}(x)$. Also, the level of x is the level of $\mathcal{R}(x)$.

Leaf node. A leaf node x has no children, is of level $\ell^{\mathcal{R}(x)} = 0$, and hence has $W^{\mathcal{R}(x)} = \emptyset$.

Filter node. A filter node x has no or one child. If $\mathcal{R}(x)$ contains an offending vertex $v \in A^{\mathcal{R}(x)}$, then x has no children. Otherwise, x contains a single child y with $\mathcal{R}(y)$ being the result of the filtering step applied to $\mathcal{R}(x)$. Note that the child of a filter node is always clean.

We additionally require that a parent of a filter node is not a filter node.

Split node. We say that a subproblem $\mathcal{R}$ is *splittable* if $\ell^{\mathcal{R}} \geqslant 1$ and every connected component of $G[W^{\mathcal{R}}]$ has size less than $0.99^{-\ell^{\mathcal{R}}+1}$. We observe that it is straightforward to split a splittable subproblem into a constant number of subproblems of smaller level.

Lemma 15. *If $\mathcal{R}$ is a splittable subproblem, then there exists a family $\mathcal{F}$ of one or two subproblems of level $\ell^{\mathcal{R}} - 1$ that all extend $\mathcal{R}$, so that for every $\mathcal{Q} \in \mathcal{F}$ we have $A^{\mathcal{Q}} = A^{\mathcal{R}}$, $X^{\mathcal{Q}} = X^{\mathcal{R}}$, and $\eta^{\mathcal{Q}} = \eta^{\mathcal{R}}$, and furthermore $\{W^{\mathcal{Q}} : \mathcal{Q} \in \mathcal{F}\}$ is a partition of $W^{\mathcal{R}}$ with $\zeta^{\mathcal{Q}} = \zeta^{\mathcal{R}}|_{W^{\mathcal{Q}}}$ for every $\mathcal{Q} \in \mathcal{F}$.*

PROOF. Let $C_1, \ldots, C_p$ be connected components of $G[W^{\mathcal{R}}]$ in a non-increasing order of their number of vertices. Since $\mathcal{R}$ is of level $\ell^{\mathcal{R}}$, it holds that $\sum_{i=1}^{p} |C_i| < 0.99^{-\ell^{\mathcal{R}}}$. Let j be the maximum index for which it holds that $\sum_{i=1}^{j} |C_i| < 0.99^{-\ell^{\mathcal{R}}+1}$. Then the assumption that $\mathcal{R}$ is splittable implies $j \geqslant 1$.

If $j = p$, then we can set $\mathcal{F}$ to be a singleton that contains a copy of $\mathcal{R}$. In particular, the set of active vertices remains unchanged. So now assume $j < p$. Since we ordered C_is in a non-increasing order of the number of vertices, the maximality of j implies that $\sum_{i=1}^{j} |C_i| \geqslant \frac{1}{2} \cdot 0.99^{-\ell^{\mathcal{R}}+1}$. Consequently,

$$\sum_{i=j+1}^{p} |C_i| < (1 - 0.495) \cdot 0.99^{-\ell^{\mathcal{R}}} < 0.99^{-\ell^{\mathcal{R}}+1}.$$

Hence, we can split $W^{\mathcal{R}}$ into $\bigcup_{i=1}^{j} C_i$ and $\bigcup_{i=j+1}^{p} C_i$ for the active sets of the subproblems of $\mathcal{F}$. Note that in this case $\mathcal{F}$ is of size two. $\square$

A split node x satisfies the following properties: $\mathcal{R}(x)$ is a splittable subproblem and the family of subproblems of children of x satisfy the properties promised by [Lemma 15](#) for the family $\mathcal{F}$. Moreover, we require that all the children of a split node are free nodes.

We remark that if a split node x is clean, then all the children of x are clean as well.

We also make the following observation.

Lemma 16. *Let x be a clean split node with children $\mathcal{Y}$. For every $y \in \mathcal{Y}$, let $\mathcal{R}'_y$ be a clean subproblem extending $\mathcal{R}(y)$ completely. Define a subproblem $\mathcal{R}'$ as*

$$A^{\mathcal{R}'} = \bigcup_{y \in \mathcal{Y}} A^{\mathcal{R}'_y}, \qquad X^{\mathcal{R}'} = \bigcup_{y \in \mathcal{Y}} X^{\mathcal{R}'_y}, \qquad \eta^{\mathcal{R}'} = \bigcup_{y \in \mathcal{Y}} \eta^{\mathcal{R}'_y},$$

$$W^{\mathcal{R}'} = \emptyset, \qquad\qquad \zeta^{\mathcal{R}'} = \emptyset, \qquad\qquad \ell^{\mathcal{R}'} = 0.$$

Then, $\mathcal{R}'$ is clean and extends completely $\mathcal{R}(x)$.

PROOF. Most of the asserted properties follow from the definitions in a straightforward manner; here we only discuss the nontrivial ones.

First, note that $\eta^{\mathcal{R}'}$ is a well-defined function with domain $A^{\mathcal{R}'}$. This is because each $\eta^{\mathcal{R}'_y}$ extends $\eta^{\mathcal{R}(x)}$, and the sets $\{A^{\mathcal{R}'_y} - A^{\mathcal{R}(x)} : y \in \mathcal{Y}\}$ are pairwise disjoint, because $\{W^{\mathcal{R}(y)} : y \in \mathcal{Y}\}$ is a partition

of $W^{\mathcal{R}(x)}$. Thus, $\mathcal{R}'$ is indeed a subproblem extending completely $\mathcal{R}(x)$. It remains to show that it is clean, that is, there are no offending vertices in $A^{\mathcal{R}'}$ (note here that $W^{\mathcal{R}'}$ is empty).

Consider first a vertex $v \in A^{\mathcal{R}'_y} - A^{\mathcal{R}(x)}$ for some $y \in \mathcal{Y}$. Then, since $W^{\mathcal{R}(y)}$ is nonadjacent to $R^{\mathcal{R}(y)} - W^{\mathcal{R}(y)}$, we have that $N_G(v) \subseteq W^{\mathcal{R}(y)} \cup A^{\mathcal{R}(x)} \cup X^{\mathcal{R}(x)}$. We infer that since v is not offending in $\mathcal{R}'_y$, due to this subproblem being clean, it is also not offending in $\mathcal{R}'$.

Consider now a vertex $v \in A^{\mathcal{R}(x)}$. Let $u \in N_G(v) \cap A^{\mathcal{R}'}$ be such that $\eta^{\mathcal{R}'}(u) < \eta^{\mathcal{R}'}(v)$. Then either $u \in A^{\mathcal{R}(x)}$ and $\eta^{\mathcal{R}(x)}(u) < \eta^{\mathcal{R}(x)}(v)$, or $u \in W^{\mathcal{R}(x)}$ and $\zeta^{\mathcal{R}(x)}(u) < \eta^{\mathcal{R}(x)}(v)$. Since v is not offending in $\mathcal{R}(x)$, the number of such vertices u is bounded by d. Hence, v is not offending in $\mathcal{R}'$. This completes the proof of the lemma. $\square$

Branch node. Branch node x is additionally labelled with a *pivot* $\nu^x \in W^{\mathcal{R}(x)}$. Intuitively, in the branching we decide whether ν^x belongs to the constructed solution or not. In case we decide to include ν^x in A, we also guess its left neighbors and their positions in our fixed degeneracy ordering, as well as several other useful pieces of information.

Formally, a branch node x has a child y^x such that $\mathcal{R}(y^x)$ is the result of deleting ν^x from $\mathcal{R}(x)$. We call the edge pair xy^x of the subproblem tree the *failure branch* at node x, while y^x is the *failure child* of x. Note that if x is clean, then the failure child y^x is also clean.

Additionally, for every tuple $\mathcal{D} = (D, \eta, (D_u)_{u \in D'})$, where

- $D \subseteq W^{\mathcal{R}(x)} \cap N_G(\nu^x)$ is of size at most $\gamma(\eta^{\mathcal{R}(x)}, \nu^x, \zeta^{\mathcal{R}(x)}(\nu^x))$,

- η is a position guess for $\mathcal{R}$ and D', where $D' := D \cup \{\nu^x\}$, and

- $(D_u)_{u \in D'}$ is a left neighbor guess for $\mathcal{R}$ and D' such that $D_{\nu^x} = \emptyset$,

the branch node x has a child $z^x_{\mathcal{D}}$. This child is associated with the subproblem $\mathcal{R}(z^x_{\mathcal{D}})$ that is the result of taking D' at positions η with left neighbors $(D_u)_{u \in D'}$ in the subproblem $\mathcal{R}$. We call each edge $xz^x_{\mathcal{D}}$ of the subproblem tree a *success branch* at node x, and $z^x_{\mathcal{D}}$ is a *success child* of x.

We remark that the children $z^x_{\mathcal{D}}$ may not be clean even if x is clean. Therefore, we require that every child $z^x_{\mathcal{D}}$ is a filter node. The child of $z^x_{\mathcal{D}}$, if present, is denoted as $s^x_{\mathcal{D}}$ and is called a *success grandchild* of x. We require that all the success grandchildren of x are free nodes.

Observe that for success children of a branch node, there are at most n^d choices for D, at most n^{d+1} choices for η, and at most n^d choices for each D_u, $u \in D$. This gives at most $n^{d^2+2d+1} = n^{(d+1)^2}$ choices for the tuple $\mathcal{D}$, and consequently this is an upper bound on the number of success children.

Finally, we make the following observation that follows immediately from Lemma 13:

Lemma 17. *Assume that a branch node x is lucky. If $\nu^x \notin S^*$, then the failure child y^x is lucky. Otherwise, if $\nu^x \in S^*$, then for*

$$D = \{u \in N_G(\nu^x) \cap W^{\mathcal{R}(x)} \cap S^* \mid \eta^*(u) < \eta^*(\nu^x)\},$$
$$\eta = \eta^*|_{A^{\mathcal{R}(x)} \cup D'},$$
$$D_u = \{w \in N_G(u) \cap W^{\mathcal{R}(x)} \cap S^* - D' \mid \eta^*(w) < \eta^*(u)\},$$

we have $|D| \leqslant d$, η is a position guess for $\mathcal{R}(x)$ and D', $(D_u)_{u \in D'}$ is a left neighbor guess for $\mathcal{R}(x)$, D', and η satisfying $D_{\nu^x} = \emptyset$, and $z^x_{\mathcal{D}}$ is a lucky success child of x.

Free node.   For free nodes, we put three restrictions:

- a free node is a success grandchild of a branch node or a child of a split node;

- the children of a free node have the same level as the free node itself; and

- a child of a free node is clean or is a filter node.

Recall also the requirements stated in the above sections:

- Every child of a split node is a free node.

- Every success grandchild of a branch node is a free node.

Free nodes are essentially not used in the proof of Theorem 11. Precisely, we use a trivial subroutine of handling them that just passes the same instance to the child, which can be a node of a different type. Free nodes will play a role in further results, precisely in the proof of Theorem 1, where they will be used as placeholders for additional branching steps, implemented using non-trivial subroutines for handling free nodes. The requirements stated above boil down to placing free nodes as children of split nodes and as success grandchildren of branch nodes, which means that the algorithm is allowed to perform the discussed additional guessing at these points. As we will see in the analysis, both passing through a split node and through a success branch correspond to some substantial progress in the recursion, which gives space for those extra branching steps.

Final remarks.   The requirements that the root node is clean, that success children of a branch node are filter nodes, and that a child of a free node is either clean or a filter node, ensure the following property: every node of the subproblem tree is clean unless it is a filter node.

In a subproblem tree, the level of a child node equals the level of the parent, unless the parent node is a split node, in which case the level of a child is one less than the level of a parent. In particular, on a root-to-leaf path in a subproblem tree the levels do not increase.

Also, if y is a child of x, then $W^{\mathcal{R}(y)} \subseteq W^{\mathcal{R}(x)}$. Furthermore, $W^{\mathcal{R}(y)} = W^{\mathcal{R}(x)}$ can happen only when x is a split node, a filter node, or a free node. Taking into account the restrictions on the parents of filter and free nodes, we infer the following:

Lemma 18. *The depth of a subproblem tree rooted in a node x is at most $\mathcal{O}(|W^{\mathcal{R}(x)}| + \ell^{\mathcal{R}(x)})$.*

4.3   Branching strategy

A *branching strategy* is a recursive algorithm that, given a subproblem $\mathcal{R}$, creates a node x of a subproblem tree with $\mathcal{R}(x) = \mathcal{R}$, decides the type of x, appropriately constructs subproblems for the children of x, and recurses on those children subproblems. The subproblem trees returned by the recursive calls are then attached by making their roots children of x. A branching strategy can pass some additional information to the child subcalls; for instance, after creating a split node, the call should ask all the subcalls to create free nodes, while a branch node should tell all its success children to be filter nodes with free nodes as their children. We require that the subproblem tree constructed by a branching strategy is a subproblem tree, as defined in the previous section.

A few remarks are in place. If the level of $\mathcal{R}$ is 0, the branching strategy has no option but to create a leaf node and stop (unless the parent or grandparent asks it to perform a filter or free node first). If a branching strategy makes a decision to create a filter node or a split node, there are no more decisions to

make: the filtering step works deterministically, and for the split node we always create child subproblems using Lemma 15.

If a branching strategy decides to make a branch node, the only remaining decision is to choose the pivot ν^x; after this, the failure branch and the success branches are defined deterministically. The method of choosing the pivot is the cornerstone of our combinatorial analysis, and is presented in the remainder of this section. (We will also use the freedom of sometimes *not* choosing to create a split node, even if the current subproblem is splittable.)

Finally, the definition of a subproblem tree allows only a limited freedom of creating free nodes: they have to be children of a split node or success grandchildren of a branch node. On the other hand, we would like to leave the freedom of how to handle free nodes to applications. Thus, a branching strategy is parameterized by a subroutine that handles free nodes: the subroutine is called by the branching strategy when handling a child of a split node or a success grandchild of a branch node, and asked to create the family of subproblems for children.

For Theorem 11, we do not use the possibility of creating free nodes. Formally, the subroutine handling free nodes, invoked at a subproblem $\mathcal{R}$, returns a one-element family $\{\mathcal{R}\}$, that is, creates a dummy free node x with a single child y such that $\mathcal{R}(x) = \mathcal{R}(y)$.

An application of a branching strategy to a graph G is a shorthand for applying the branching strategy to a subproblem $\mathcal{R}$ defined by:

- $A^{\mathcal{R}} = X^{\mathcal{R}} = \emptyset$;

- $W^{\mathcal{R}} = V(G)$;

- $\ell^{\mathcal{R}} = \lceil -\log_{0.99}(|V(G)| + 1) \rceil$, and

- $\zeta^{\mathcal{R}}(v) = 1$ for every $v \in V(G)$.

Note that such a subproblem $\mathcal{R}$ is clean and lucky, regardless of the choice of S^* and η^*.

Section 5 is devoted to branching strategies that lead to a quasipolynomial running time bounds. Formally, we prove there the following:

Lemma 19. *For every fixed pair of integers d and t, and every subroutine handling free nodes, there exists a branching strategy that, when applied to an n-vertex $C_{>t}$-free graph G, creates a recursion tree that is a subproblem tree such that on every root-to-leaf path there are $\mathcal{O}(\log n)$ split nodes and $\mathcal{O}(\log^3 n)$ success branches. If the input graph is P_t-free, the bound on the number of success branches on a single root-to-leaf path improves to $\mathcal{O}(\log^2 n)$.*

Furthermore, the branching strategy takes polynomial time to decide on the type of the node and on the choice of the branching pivot (in case the node is decided to be a branch node).

Let us now show how Lemma 19 implies Theorem 11

PROOF (OF THEOREM 11.). Fix a set $S^* \subseteq V(G)$ maximizing $\mathfrak{w}(S^*)$ subject to $G[S^*]$ being of degeneracy at most d. Fix also a degeneracy ordering η^* of $G[S^*]$. This allows us to speak about lucky nodes.

Consider the branching strategy provided by Lemma 19 with the discussed dummy subroutine handling free nodes.

By Lemma 18, the depth of the subproblem tree is $\mathcal{O}(n)$. Furthermore, every node of the subproblem tree has at most $1 + n^{(d+1)^2}$ children, and the only nodes that may have more than one child are branch nodes and split nodes. Since each branch node has only one failure child, while on every root-to-leaf path

22

there are $\mathcal{O}(\log n)$ split nodes and $\mathcal{O}(\log^3 n)$ success branches, it follows that the subproblem tree has $n^{\mathcal{O}(\log^3 n)}$ nodes.

For every clean node x in the generated subproblem tree, we would like to compute a clean subproblem $\mathcal{R}'(x)$ that extends $\mathcal{R}(x)$ completely and, subject to that, maximizes $\mathfrak{w}(A^{\mathcal{R}'(x)})$. Observe that for the answer to $\mathcal{R}'(r)$ at the root node r we can return $A^{\mathcal{R}'(r)}$ as the sought subset S: the cleanness of $\mathcal{R}'(r)$ implies that $G[A^{\mathcal{R}'(r)}]$ is d-degenerate, while a subproblem of level 0 with $A = S^*$, $X = V(G) - S^*$, and $\eta = \eta^*$ extends completely $\mathcal{R}(r)$.

For a leaf node x, the only option is to return $\mathcal{R}(x)$. For a split node x with children $\mathcal{Y}$, we apply Lemma 16 to $\{\mathcal{R}'(y) \mid y \in \mathcal{Y}\}$, thus obtaining a clean subproblem $\mathcal{R}'(x)$. For a branch node x, we take $\mathcal{R}'(x)$ to be the subproblem with the maximum weight of A among subproblem $\mathcal{R}'(y^x)$ for the failure child y^x and subproblems $\mathcal{R}'(s_{\mathcal{D}}^x)$ for success grandchildren $s_{\mathcal{D}}^x$. Note that both $\mathcal{R}'(y^x)$ and every $\mathcal{R}'(s_{\mathcal{D}}^x)$ are clean subproblems that extending completely $\mathcal{R}(x)$. Finally, for a dummy free node x with a child y, $\mathcal{R}'(x)$ equals $\mathcal{R}'(y)$.

To finish the proof, it suffices to argue that for every lucky node x, we have

$$\mathfrak{w}(A^{\mathcal{R}'(x)}) \geqslant \mathfrak{w}(S^* \cap (A^{\mathcal{R}(x)} \cup W^{\mathcal{R}(x)})). \tag{2}$$

We prove this fact by a bottom-up induction on the subproblem tree.

For a lucky leaf node the claim is straightforward, as being lucky implies that $A^{\mathcal{R}(x)} \subseteq S^*$ while being a leaf node implies $W^{\mathcal{R}(x)} = \emptyset$. For a lucky split node, note that all children are also lucky; the claim is immediate by the inductive assumption for the children and the construction of Lemma 16. For a lucky branch node x, Lemma 17 implies that x has a lucky failure child or a lucky success grandchild; the claim follows from the inductive assumption of the said lucky (grand)child. Finally, for a lucky dummy free node, the claim is immediate by the inductive assumption on its child.

This finishes the proof of Theorem 11. $\qquad\square$

5 Branching strategies: choosing pivots in P_t-free and $C_{>t}$-free graphs

This section is devoted to the proof of Lemma 19. Recall that a branching strategy is essentially responsible for choosing whether the current node of the tree is a split node or a branch node and, in the latter case, choosing the branching pivot.

The following observation, immediate from the definition of a success branch, will be the basic source of progress.

Lemma 20. *Let x be a branch node and $y := s_{\mathcal{D}}^x$ be a successful grandchild of x, where $\mathcal{D} = (D, \eta, (D_u)_{u \in D'})$ and $D' = D \cup \{\nu^x\}$. Then, for every $u \in N_G(\nu^x)$, we have either $u \notin W^{\mathcal{R}(y)}$ or $\zeta^{\mathcal{R}(y)}(u) > \eta^{\mathcal{R}(y)}(\nu^x)$. Consequently, for every $u \in N_G(\nu^x) \cap W^{\mathcal{R}(y)}$, it holds that*

$$\gamma(\eta^{\mathcal{R}(y)}, u, \zeta^{\mathcal{R}(y)}(u)) \leqslant \gamma(\eta^{\mathcal{R}(x)}, u, \zeta^{\mathcal{R}(x)}(u)) - 1.$$

PROOF. If $u \in W^{\mathcal{R}(y)}$, then in particular $y \notin D$. Then, in the definition of $\zeta^{\mathcal{R}(z_{\mathcal{D}}^x)}(u)$, one term over which the maximum is taken is equal to $1 + \eta(\nu^x)$ and $\eta(\nu^x) = \eta^{\mathcal{R}(y)}(\nu^x)$. Thus, ν^x, which is in $W^{\mathcal{R}(x)} \cap A^{\mathcal{R}(y)}$, is taken into account in $\gamma(\eta^{\mathcal{R}(y)}, u, \zeta^{\mathcal{R}(y)}(u))$, but does not contribute to $\gamma(\eta^{\mathcal{R}(x)}, u, \zeta^{\mathcal{R}(x)}(u))$ due to being not included in $A^{\mathcal{R}(x)}$. $\qquad\square$

5.1 Quasipolynomial branching strategy in P_t-free graphs

To obtain the promised bounds for P_t-free graphs, we closely follow the arguments for MWIS from [25].

Let us fix d, t, and a subroutine handling free nodes.

The branching strategy chooses a leaf node whenever possible (the level of the subproblem is zero) and a split node whenever possible (the subproblem is splittable). Given a subproblem of positive level that is not splittable, the branching strategy creates a branch node.

For a P_t-free graph G and a pair $\{u, v\} \in \binom{V(G)}{2}$, the *bucket* of $\{u, v\}$ is a set $\mathcal{B}^G_{u,v}$ that consists of all induced paths with endpoints u and v. Two remarks are in place. First, $\mathcal{B}^G_{u,v} \neq \emptyset$ if and only if u and v are in the same connected component of G. Second, an n-vertex P_t-free graph has at most n^{t-1} induced paths; hence, all buckets can be enumerated in polynomial time.

For $\varepsilon > 0$, a vertex $x \in V(G)$ is ε-*heavy* if the neighborhood $N[x]$ intersects more that ε fraction of paths from more than ε fraction of buckets, that is,

$$\left| \left\{ \{u, v\} \in \binom{V(G)}{2} \,\middle|\, |\{P \in \mathcal{B}^G_{u,v} \mid N[x] \cap V(P) \neq \emptyset\}| > \varepsilon |\mathcal{B}^G_{u,v}| \right\} \right| > \varepsilon \binom{|V(G)|}{2}.$$

Note that empty buckets cannot contribute towards the left hand side of the inequality above.

We make use of the following lemma.

Lemma 21 ([25]). *A connected P_t-free graph contains a $\frac{1}{2t}$-heavy vertex.*

In our case, a subproblem $\mathcal{R}$ at a branch node may not have $G[W^{\mathcal{R}}]$ connected, but $G[W^{\mathcal{R}}]$ contains a large connected component if $\mathcal{R}$ is not splittable.

Corollary 22. *If $\mathcal{R}$ is a not splittable subproblem of positive level, then $G[W^{\mathcal{R}}]$ contains a $\frac{1}{3t}$-heavy vertex.*

PROOF. Recall that $n^{\mathcal{R}} := |W^{\mathcal{R}}| < 0.99^{-\ell^{\mathcal{R}}}$. Since $\mathcal{R}$ is of positive level but not splittable, there is a connected component D of $G[W^{\mathcal{R}}]$ of size at least $0.99^{-\ell^{\mathcal{R}}+1}$. That is, $|D| \geqslant 0.99 n^{\mathcal{R}}$.

Lemma 21 asserts there is a $\frac{1}{2t}$-heavy vertex in $G[D]$. We have

$$\binom{|D|}{2} \geqslant \binom{\lceil 0.99 n^{\mathcal{R}} \rceil}{2} \geqslant 0.9 \binom{n^{\mathcal{R}}}{2}.$$

Thus, x is $\frac{0.9}{2t}$-heavy in G and $\frac{0.9}{2t} > \frac{1}{3t}$. $\square$

The branching strategy chooses a $\frac{1}{3t}$-heavy vertex of $G[W^{\mathcal{R}(x)}]$ as a branching pivot ν^x at a branch node x. Recall that at a branch node x the subproblem $\mathcal{R}(x)$ is of positive level and not splittable. Hence, Corollary 22 asserts the existence of such a pivot. As already discussed, all buckets can be enumerated in polynomial time and thus such a heavy vertex can be identified.

The bound on the number of split nodes on any root-to-leaf path in the subproblem tree is immediate from the fact that there are $\mathcal{O}(\log n)$ levels. It remains to argue that any root-to-leaf path has $\mathcal{O}(\log^2 n)$ success branches.

To this end, let Q be a maximal upward path in the subproblem tree that consists of nodes of the same level ℓ. As there are $\mathcal{O}(\log n)$ possible levels, it suffices to prove that Q contains $\mathcal{O}(\log n)$ success branches.

Consider then a success branch on Q: a branch node x and its success grandchild $y := s^x_{\mathcal{D}}$ for $\mathcal{D} = (D, \eta, (D_u)_{u \in D'})$, $D' = D \cup \{\nu^x\}$. Lemma 20 suggests the following potential at node x:

24

$$\mu(x) := \sum_{\{u,v\} \in \binom{W^{\mathcal{R}(x)}}{2}} \log_2 \left[1 + \sum_{P \in \mathcal{B}_{u,v}^{G[W^{\mathcal{R}(x)}]}} \sum_{u \in V(P)} (1 + \gamma(\eta^{\mathcal{R}(x)}, u, \zeta^{\mathcal{R}(x)}(u))) \right].$$

Since we branch on a $\frac{1}{3t}$-heavy vertex and the innermost sums in the definition above are bounded by $\mathcal{O}(dt)$, we have that for some universal constant c:

$$\mu(x) - \mu(y) \geqslant \frac{c}{dt^3} \binom{|W^{\mathcal{R}(x)}|}{3}.$$

Let x_0 be the topmost node of Q and $n_Q = |W^{\mathcal{R}(x_0)}|$. Since all nodes of x_0 are of the same level, we have $|W^{\mathcal{R}(x)}| \geqslant 0.99 n_Q$ for every branch node x on Q. We infer that

$$\mu(x) - \mu(y) = \Omega(n_P^2).$$

We have $\mu(x_0) = \mathcal{O}(n_P^2 \log n_P)$ while $\mu(x) \geqslant 0$ for any node x. Consequently, Q may contain $\mathcal{O}(\log n_P) = \mathcal{O}(\log n)$ success branches, as desired.

This finishes the proof of Lemma 19 for P_t-free graphs.

5.2 Quasipolynomial branching strategy in $C_{>t}$-free graphs

We now prove Lemma 19 for $C_{>t}$-free graphs. Hence, let us fix d, t, and a subroutine handling free nodes. W.l.o.g. assume t is even and $t \geqslant 6$. Recall that our goal is to design a branching strategy, which given a subproblem $\mathcal{R}$ should decide the type of the node created for $\mathcal{R}$ and, in case this type is the branch node, choose a suitable branching pivot. We will measure the progress of our algorithm by keeping track of the number of some suitably defined objects in the graph.

A *connector* is a graph with three designated vertices, called *tips*, obtained in the following way. Take three induced paths Q_1, Q_2, Q_3; here we allow degenerated, one-vertex paths. The paths Q_1, Q_2, Q_3 will be called the *legs* of the connector. The endvertices of Q_i are called a_i and b_i. Now, join these paths in one of the following ways:

a) identify a_1, a_2, and a_3 into a single vertex, i.e., $a_1 = a_2 = a_3$, or

b) add edges $a_1 a_2$, $a_2 a_3$, and $a_1 a_3$.

Furthermore, if the endvertices are identified, then at most one leg may be degenerated; thus, b_1, b_2, b_3 are pairwise different after the joining. There are no other edges between the legs of the connector. The vertices b_1, b_2, and b_3 are the *tips* of the connector, and the set $\{a_1, a_2, a_3\}$ is called the *center* (this set can have either three or one element); see Figure 1. If one of the paths forming the connector has only one vertex, and the endvertices were identified, then the connector is just an induced path with the tips being the endpoints of the path plus one internal vertex of the path. Note that, given a connector as a graph and its tips, the legs and the center of the connector are defined uniquely.

We will need the following folklore observation.

Lemma 23. *Let G be a graph, $A \subseteq V(G)$ be a set consisting of exactly three vertices in the same connected component of G, and let $A \subseteq B \subseteq V(G)$ be an inclusion-wise minimal set such that $G[B]$ is connected. Then the graph $G[B]$ with the set A as tips is a connector.*
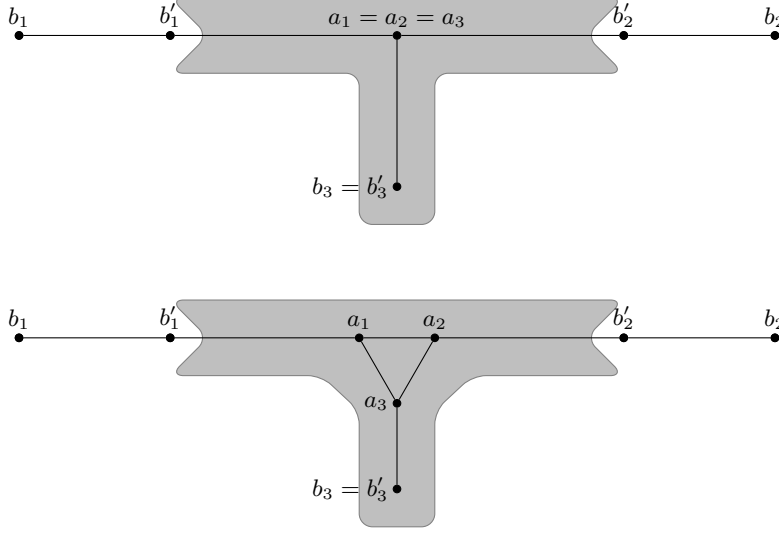
Figure 1: Two connectors with two long legs and one short leg; one connector with a_is identified and one with a_is forming a triangle. Vertices b_i' are the tips of the core T of the connector. The gray area depicts the set T^*.

Proof. Let $A = \{u, v, w\}$, let P_{uv} be a shortest path from u to v in $G[B]$ and let P_w be a shortest path from w to $V(P_{uv})$ in $G[B]$. By minimality of B, we have $B = V(P_{uv}) \cup V(P_w)$.

If $w \in V(P_{uv})$ (equivalently, $|V(P_w)| = 1$), then $G[B]$ is a path and we are done. Otherwise, let $q \in V(P_w) \cap V(P_{uv})$ be the endpoint of P_w distinct than w and let p be the unique neighbor of q on P_w. By the minimality of P_w, p and q are the only vertices of P_w that may have neighbors on P_{uv}. If p has two neighbors $x, y \in V(P_{uv})$ that are not consecutive on P_{uv}, then $G[B]$ remains connected after the deletion from B of all vertices on P_{uv} between x and y (exclusive), a contradiction to the choice of B. Thus, $N(p) \cap V(P_{uv})$ consists of q and possibly one neighbor of q on P_{uv}. We infer that $G[B]$ is a connector with tips u, v, and w, as desired. □

A *tripod* is a connector where each of the paths Q_1, Q_2, Q_3 has at most $t/2 + 1$ vertices. A leg of a tripod is *long* if it contains exactly $t/2 + 1$ vertices, and *short* otherwise. The *core* of a connector C with legs Q_1, Q_2, Q_3, is the tripod consisting of the first $t/2 + 1$ (or all of them, if the corresponding path Q_i is shorter) vertices of each path Q_i, starting from a_i. A tripod *in G* is a tripod that is an induced subgraph of G. Note that each tripod has at most $3t/2 + 3$ vertices, hence given G, we can enumerate all tripods in G in time $n^{O(t)}$.

Let T be a tripod in G with legs Q_1, Q_2, Q_3. Let $L(T)$ denote the tips of T which are endvertices of the long legs. We denote

$$T^* := N_G[V(T) - L(T)] - L(T).$$

In other words, T^* is the closed neighborhood of the tripod T in G, except that we do not necessarily include the neighbors of tips of long legs and we exclude those tips themselves. Note that tips of short legs together with their neighborhoods are included in T^*.

We have the following simple observation.

Lemma 24. *For every tripod T in G and for every connected component C of $G - T^*$, the component C contains at most one tip of $L(T)$ and no tip of a short leg.*

PROOF. First, note that tips of short legs are contained in T^*, hence they are not contained in $G - T^*$.

For contradiction, without loss of generality assume that $b_1, b_2 \in C \cap L(T)$. Then, Q_1, Q_2, and a shortest path from b_1 to b_2 in C yield an induced cycle on more than t vertices in G, a contradiction. □

Suppose T is a tripod in G. With each tip b_i of T we associate a *bag* B_i defined as follows:

- if b_i is the endpoint of a long leg, then B_i is the vertex set of the connected component of $G - T^*$ that contains b_i; and

- otherwise, $B_i = \{b_i\}$.

Note that Lemma 24 implies that the bags B_1, B_2, B_3 are pairwise disjoint and nonadjacent in G, except for the corner case of two adjacent tips of short legs.

We now define *buckets* that group tripods. Each bucket will be indexed by an unordered triple of distinct vertices of G. Every tripod T in G with bags B_1, B_2, B_3 belongs to the bucket $\mathcal{B}^G_{u,v,w}$ for all triples u, v, w such that $u \in B_1$, $v \in B_2$, and $w \in B_3$. Note that thus, the buckets do not have to be pairwise disjoint. The superscript G can be omitted if the graph is clear from the context.

Observe the following.

Lemma 25. *For every $\{u, v, w\} \in \binom{V(G)}{3}$ and every $T \in \mathcal{B}_{u,v,w}$, there exists a connector T' with tips u, v, w whose core equals T. Consequently, the bucket $\mathcal{B}_{u,v,w}$ is nonempty if and only if u, v, w lie in the same connected component of G.*

PROOF. Let Q_u, Q_v, Q_w be the legs of T with tips b_u, b_v, b_w and bags B_u, B_v, and B_w, respectively, such that $u \in B_u, v \in B_v, w \in B_w$. Let Q'_u be the concatenation of Q_u and a shortest path from b_u to u in $G[B_u]$. Similarly define Q'_v and Q'_w.

Recall that the bags B_u, B_v, and B_w are pairwise distinct and nonadjacent (except for the case of two adjacent tips of short legs). Hence, Q'_u, Q'_v, and Q'_w form a connector T' with tips u, v, and w. Since $B_u \neq \{b_u\}$ only if the leg Q_u is long, T is the core of T'. □

The next combinatorial observation is critical to the complexity analysis.

Lemma 26. *Let G be a connected $C_{>t}$-free graph, and let u, v, w be three distinct vertices of G. Let $X \subseteq V(G)$ be such that $G[X]$ is connected and no two of u, v, w are in the same connected component of $G - N[X]$. Then $N[X]$ intersects all tripods in $\mathcal{B}_{u,v,w}$.*

PROOF. Let $T \in \mathcal{B}_{u,v,w}$. Let Q_u, Q_v, Q_w be the legs of T with tips b_u, b_v, b_w and bags B_u, B_v, and B_w, respectively. Let T' be the connector for T given by Lemma 25 with legs Q'_u, Q'_v, and Q'_w. Since no two of u, v, w lie in the same connected component of $G - N[X]$, the set $N[X]$ intersects at least two legs of Q'_u, Q'_v, and Q'_w. Without loss of generality, assume that $N[X]$ intersects Q'_u and Q'_v. Let $u' \in N[X] \cap V(Q'_u)$ be the vertex of $N[X] \cap V(Q'_u)$ that is farthest from u on Q'_u and similarly define $v' \in N[X] \cap V(Q'_v)$. Then, the subpath of Q'_u from u' to the center of T', the subpath of Q'_v from v' to the center of T', and a shortest path from u' to v' with all internal vertices in X form an induced cycle in G. This cycle has more than t vertices unless $u' \in V(Q_u)$ or $v' \in V(Q_v)$. Hence, $V(T) \cap N[X] \neq \emptyset$, as desired. □

For $\varepsilon > 0$, we will say that a vertex v is ε-*heavy* (or just *heavy*, if ε is clear from the context) if $N[v]$ intersects strictly more than ε fraction of tripods in at least ε fraction of buckets. Note that the "strictly more" part makes empty buckets not count toward the ε fraction of the buckets hit.

Intuitively, our branching strategy chooses as the pivot a $(10^{-8}/t)$-heavy vertex of $G[W^{\mathcal{R}}]$. Lemma 20 captures the source of the gain in a success branch: the quotas of the neighbors of the pivot get reduced.

Unfortunately, it may happen that there is no $(10^{-8}/t)$-heavy vertex. For an example, consider the case when $H := G[W^{\mathcal{R}}]$ is a long path. Then $\mathcal{B}_{u,v,w}$ consists of subpaths of H of length at most t containing the middle vertex of $\{u, v, w\}$, and an arbitrary neighborhood $N[x]$ intersects tripods in roughly $(t/|V(H)|)$ fraction of all buckets.

However, there is one particular scenario where a heavy vertex is guaranteed to exist. In a connected graph G, a set $X \subseteq V(G)$ is a *connected three-way balanced separator* (C3WBS) if

- $G[X]$ is connected;

- the family of connected components of $G - N[X]$ can be partitioned into three sets $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ such that for every $i = 1, 2, 3$ we have $|N[X] \cup \bigcup \mathcal{C}_i| \geqslant 0.1|V(G)|$.

We observe that a small connected three-way balanced separator in a connected $C_{>t}$-free graph guarantees the existence of a heavy vertex.

Lemma 27. *Let G be a connected $C_{>t}$-free graph and let $X \subseteq V(G)$ be a C3WBS. Then there exists a $(10^{-6}/|X|)$-heavy vertex in G.*

PROOF. Let $n = |V(G)|$. If $n \leqslant 2$, then any vertex of G is heavy, as there are no buckets. If $2 < n \leqslant 100$, then any vertex $v \in V(G)$ with at least two neighbors, say u_1, u_2, is 10^{-6}-heavy as $N[v]$ hits all tripods in the bucket $\mathcal{B}_{u_1,u_2,v}$ by Theorem 6 and there are fewer than 10^6 buckets. Hence, we can assume $n > 100$.

Let $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ be the partition of the connected components of $G - N[X]$ promised by the definition of the C3WBS X. Let $C_i = \bigcup \mathcal{C}_i$ for $i = 1, 2, 3$. Let $\mathcal{X}$ be the family of triples $\{u_1, u_2, u_3\} \in \binom{V(G)}{3}$ with $u_i \in C_i \cup N[X]$ for $i = 1, 2, 3$. We can estimate the size of $\mathcal{X}$ as follows:

$$
\begin{aligned}
|\mathcal{X}| = & |C_1| \cdot |C_2| \cdot |C_3| + |N[X]| \cdot (|C_1| \cdot |C_2| + |C_2| \cdot |C_3| + |C_3| \cdot |C_1|) \\
& + \binom{|N[X]|}{2} \cdot (|C_1| + |C_2| + |C_3|) + \binom{|N[X]|}{3} \\
\geqslant & \frac{1}{6} \left(|C_1| + |N[X]|\right) \cdot \left(|C_2| + |N[X]| - 1\right) \cdot \left(|C_3| + |N[X]| - 2\right) \\
\geqslant & \frac{1}{6} \, 0.1n \cdot (0.1n - 1) \cdot (0.1n - 2) \\
\geqslant & \frac{10^{-3}}{6} \, n \cdot (n - 10) \cdot (n - 20) \\
\geqslant & 10^{-4} \cdot \binom{n}{3}.
\end{aligned}
$$

In the last inequality we have used the assumption $n > 100$.

By Lemma 26, the set $N[X]$ intersects all tripods in at least 10^{-4} fraction of the buckets. Hence, there exists $w \in X$ such that $N[w]$ intersects at least $1/|X|$ fraction of tripods in at least $10^{-4}/|X|$ fraction of the buckets. $\qquad \square$

Unfortunately, Theorem 6 for $A = V(G)$ does not give us a C3WBS, but only a connected set X of size at most t such that every component of $G - N[X]$ has at most $|V(G)|/2$ vertices. The example of a long path shows that the fraction $1/2$ cannot be improved while keeping X both connected and of constant size. The next lemma describes the scenario when Theorem 6 does not return a C3WBS.

Lemma 28. *Let G be a connected graph and $X \subseteq V(G)$ be such that $G[X]$ is connected and every connected component of $G - N[X]$ has at most $|V(G)|/2$ vertices. If X is not a C3WBS, then there exist exactly two connected components of $G - N[X]$, each containing at least $0.4|V(G)|$ vertices.*

PROOF. Clearly, there is not enough vertices in G for three such components. Assume then there is at most one such component; we show that X is a C3WBS. Let $C_1, C_2, \ldots, C_k$ be the connected components of $G - N[X]$ in the nonincreasing order of their sizes, that is, $|C_1| \geqslant |C_2| \geqslant \ldots \geqslant |C_k|$.

Let $i_1 \geqslant 1$ be the minimum index such that

$$\left| N[X] \cup \bigcup_{j=1}^{i_1} C_j \right| \geqslant 0.1|V(G)|.$$

If $i_1 > 1$, then $|C_{i_1}| \leqslant |C_1| < 0.1|V(G)|$. By the minimality of i_1,

$$\left| N[X] \cup \bigcup_{j=1}^{i_1-1} C_j \right| < 0.1|V(G)|.$$

Hence, $|\bigcup_{j=1}^{i_1} C_j| \leqslant 0.2|V(G)|$. If $i_1 = 1$, then $|\bigcup_{j=1}^{i_1} C_j| = |C_1| \leqslant 0.5|V(G)|$. Hence, in both cases,

$$|\bigcup_{j=1}^{i_1} C_j| \leqslant 0.5|V(G)|.$$

This allows us to define $i_2 > i_1$ to be the minimum index such that

$$\left| N[X] \cup \bigcup_{j=i_1+1}^{i_2} C_j \right| \geqslant 0.1|V(G)|.$$

Similarly as before, if $i_2 > i_1 + 1$, then $|\bigcup_{j=i_1+1}^{i_2} C_j| \leqslant 0.2|V(G)|$. If $i_2 = i_1 + 1$, then, since only C_1 is allowed to be of size at least $0.4|V(G)|$, we have $|\bigcup_{j=i_1+1}^{i_2} C_j| = |C_{i_1+1}| \leqslant 0.4|V(G)|$. Thus, in both cases,

$$|\bigcup_{j=i_1+1}^{i_2} C_j| \leqslant 0.4|V(G)|.$$

We infer that:

$$|N[X] \cup \bigcup_{j=i_2+1}^{k} C_j| \geqslant 0.1|V(G)|.$$

Hence the partition

$$\mathcal{C}_1 = \{C_j \mid 1 \leqslant j \leqslant i_1\}, \quad \mathcal{C}_2 = \{C_j \mid i_1 + 1 \leqslant j \leqslant i_2\}, \quad \mathcal{C}_3 = \{C_j \mid i_2 + 1 \leqslant j \leqslant k\}$$

proves that X is a C3WBS, as desired. □

In the absence of a heavy vertex, we shift to a secondary branching strategy. The secondary branching strategy:

(1) is initiated with a subproblem $\mathcal{R}$ and a set $X \subseteq W^{\mathcal{R}}$ of size at most t such that

- $\mathcal{R}$ is not splittable;
- $G[W^{\mathcal{R}}]$ does not admit a $(10^{-8}/t)$-heavy vertex,
- $G[X]$ is connected, and
- every connected component of $G[W^{\mathcal{R}} - N[X]]$ has at most $|C_0|/2$ vertices, where C_0 is the largest connected component of $G[W^{\mathcal{R}}]$;

(2) terminates (i.e., falls back to the primary branching strategy) if and only if when called at a subproblem $\mathcal{R}'$ that is splittable;

(3) on every root-to-leaf path in the subproblem tree created by the recursion there are $\mathcal{O}(\log^2 n^{\mathcal{R}})$ success branches.

Note that, in particular, the secondary branching strategy never forms a split node; at every node it either terminates or selects a branching pivot, makes a branch node and successive filter and free nodes for success branches. Hence, all subproblems in a subproblem tree created by the secondary branching strategy are of the same level.

We postpone the description of the secondary branching strategy to Section 5.2.1. Now, using it as a blackbox, we describe our primary strategy.

For a subproblem $\mathcal{R}$, the primary branching strategy makes the following decisions:

1. If $\mathcal{R}$ is of level 0, make a leaf node and terminate.

2. If $\mathcal{R}$ is splittable, make a split node using Lemma 15 and recurse on the constructed children.

3. If $G[W^{\mathcal{R}}]$ contains a $(10^{-8}/t)$-heavy vertex w, create a branch node x and choose w as the branching pivot ν^x.

4. Otherwise, let C_0 be the largest connected component of $G[W^{\mathcal{R}}]$ (as $\mathcal{R}$ is not splittable, there is such connected component with at least $0.99|W^{\mathcal{R}}|$ vertices), construct a set X from Theorem 6 for the graph $G[C_0]$ and invoke the secondary branching strategy on $\mathcal{R}$ and X.

We proceed with the analysis. Consider the subproblem tree of the algorithm applied to the graph G. The claim that every root-to-leaf path contains $\mathcal{O}(\log n)$ split nodes is straightforward, because the root node has level $\lceil -\log_{0.99}(n+1) \rceil = \mathcal{O}(\log n)$, the level of a child is never higher than the level of the parent, and the level of the split node is one higher than the level of its children. To show the more difficult claim that every root-to-leaf path contains $\mathcal{O}(\log^3 n)$ success branches, it suffices to show that any upward path in the subproblem tree consisting of nodes of the same level ℓ contains $\mathcal{O}(\log^2 n)$ success branches.

Consider such a maximal upward path P with all nodes of level ℓ. If the path P contains a node x where the branching strategy invoked the secondary branching strategy, then, since the secondary branching strategy terminates at a splittable node, the entire subpath P_2 of P from x downwards is contained in the subtree corresponding to the call to the secondary branching strategy. Hence, P_2 contains $\mathcal{O}(\log^2 n)$ success branches. Let P_1 be the subpath of P from x upwards, or $P_1 = P$ if P does not contain a node where the secondary branching strategy is invoked.

It suffices to show that P_1 contains $\mathcal{O}(\log^2 n)$ success branches; note that these success branches correspond to branch nodes of the primary branching strategy. We will actually show a stronger bound of $\mathcal{O}(\log n)$ success branches.

Let $n_P = |W^{\mathcal{R}(r)}|$ where r is the topmost node of P_1. By the threshold at which we use split node in the primary strategy, for every x on P_1 except for possibly the bottom-most one, we have

$$|W^{\mathcal{R}(x)}| \geqslant 0.99 n_P. \tag{3}$$

Motivated by Lemma 20, we measure the progress using the following potential at a node x on P_1:

$$\mu(x) := \sum_{\{u,v,w\} \in \binom{W^{\mathcal{R}(x)}}{3}} \log_2 \left[1 + \sum_{T \in \mathcal{B}_{u,v,w}^{G[W^{\mathcal{R}(x)}]}} \sum_{u \in V(T)} \left(1 + \gamma(\eta^{\mathcal{R}(x)}, u, \zeta^{\mathcal{R}(x)}(u)) \right) \right].$$

Observe that when y is a child of x on P, we have $\mu(x) \geqslant \mu(y)$.

Consider a branch node x with a success grandchild y in P_1. Recall that x is a branch node of the primary branching strategy and corresponds to branching on a $(10^{-8}/t)$-heavy pivot ν^x. Since each of the innermost sums in the definition of μ is upper bounded by $\mathcal{O}(dt)$, by Lemma 20 we may infer that

$$\mu(x) - \mu(y) \geqslant \frac{c}{dt^3} \binom{|W^{\mathcal{R}(x)}|}{3},$$

for some universal constant $c > 0$.

On the other hand, since every bucket is of size $n^{\mathcal{O}(t)}$, we have that $\mu(x) = \mathcal{O}(n_P^3 \log n_P)$ for every x on P_1. Hence, there are $\mathcal{O}(\log n_P)$ success branches on P_1, as desired.

5.2.1 Secondary branching strategy

We now move to the explanation of the secondary branching strategy.

For a graph H, a set $C \subseteq V(H)$ such that $H[C]$ is connected, and distinct vertices $u, v \in N_H(C)$, a C-*link* between u and v is a path P in H with the following properties:

- P has endpoints u and v and length at least 2;

- all internal vertices of P belong to C; and

- P is an induced path in $H - E(H[N_H(C)])$ (i.e. P is an induced path in H, except that we allow the existence of the edge uv).

We now make a combinatorial observation that is critical to the analysis:

Lemma 29. *Let H be a $C_{>t}$-free graph, let $X \subseteq V(H)$ be such that $H[X]$ is connected, and let C be a connected component of $H - N[X]$. Then every C-link has at most t vertices.*

PROOF. Let P be the C-link in question, let u, v be its endpoints, and let Q be a shortest path with endpoints u, v and all internal vertices in X; such Q exists because $u, v \in N[X]$ and $H[X]$ is connected. Then $P \cup Q$ is an induced cycle in H. Thus, both P and Q have at most t vertices. $\square$

Recall that the setting of the secondary branching strategy is as follows: we have a subproblem $\mathcal{R}$ and a set $X \subseteq W^{\mathcal{R}}$ such that $\mathcal{R}$ is not splittable (in particular, there is a connected component C_1 of $G[W^{\mathcal{R}}]$ if size at least $0.99|W^{\mathcal{R}}|$), $G[W^{\mathcal{R}}]$ contains no $(10^{-8}/t)$-heavy vertex, $X \subseteq C_1$, $G[X]$ is connected, $|X| \leqslant t$, and every connected component of $G[C_1] - N[X]$ is of size at most $|C_1|/2$. Let $n^{\mathcal{R}} = |W^{\mathcal{R}}|$ and $K = N_{G[W^{\mathcal{R}}]}[X]$.

As $G[W^{\mathcal{R}}]$ contains no $(10^{-8}/t)$-heavy vertex and $|C_1| \geqslant 0.99n^{\mathcal{R}}$, $G[C_1]$ contains no $(10^{-7}/t)$-heavy vertex.

If $G[C_1] - K$ contains no connected component with at least $0.4 \cdot |C_1|$ vertices, then by Lemma 28 X is a C3WBS of $G[C_1]$ and Lemma 27 implies that G_1 contains a $(10^{-6}/t)$-heavy vertex, a contradiction. Hence, there exists a component C_2 of $G[C_1] - K$ with at least $0.4 \cdot |C_1|$ vertices.

Let Y be the result of the application of Theorem 6 to $G[C_2]$; that is, $|Y| \leqslant t$, $G[Y]$ is connected, and every connected component of $G[C_2] - N[Y]$ is of size at most $0.5|C_2|$. Let $L := N_{G[C_2]}(Y)$. We make the following two observations:

Lemma 30. *The distance, in $G[C_1]$, between X and Y, is more than $8t$.*

PROOF. Let P be a shortest path in $G[C_1]$ between a vertex of K and a vertex of L. Then, $X' := X \cup Y \cup V(P)$ is connected in $G[C_1]$. Since every connected component of $G[C_1] - K$ is of size at most $0.5|C_1|$ but X is not a C3WBS of $G[C_1]$, Lemma 28 implies that there are exactly two connected components of $G[C_1] - K$ of size at least $0.4|C_1|$, one of which is C_2. Since every connected component of $G[C_2] - N[Y]$ is of size at most $0.5|C_2| \leqslant 0.25|C_1|$, there is at most one connected component of $G[C_1] - N[X']$ that is of size at least $0.4|C_1|$. Lemma 28 implies that X' is a C3WBS of $G[C_1]$. As $G[C_1]$ does not admit a $(10^{-7}/t)$-heavy vertex, Lemma 27 implies that $|X'| \geqslant 10t$. Consequently, $|V(P)| \geqslant 8t$, as desired. □

Lemma 31. *Let $\mathcal{D}$ be the family of connected components of $G[C_2] - L$. For every $D \in \mathcal{D}$, we have $N[D] \cap L \neq \emptyset$ and there exists exactly one $D_0 \in \mathcal{D}$ with $N[D_0] \cap K \neq \emptyset$.*

PROOF. The first claim follows from the connectivity of $G[C_2]$. The existence of at least one component $D_0 \in \mathcal{D}$ with $N[D_0] \cap K \neq \emptyset$ follows from the connectivity of $G[C_1]$.

Assume now there are two components $D_0, D_1 \in \mathcal{D}$ with $N[D_i] \cap K \neq \emptyset$ for $i = 0, 1$. Let P_X be a shortest path between D_0 and D_1 with internal vertices in K and let P_Y be a shortest path between D_0 and D_1 in L. Then, connecting the endpoints of P_X and P_Y via D_0 and D_1 creates an induced cycle in $G[C_1]$ and Lemma 30 implies that this cycle is longer than t. This is the desired contradiction. □

Let

$$B = L \cup \bigcup(\mathcal{D} - \{D_0\}) = C_2 - D_0,$$

where $\mathcal{D}$ and D_0 come from Lemma 31. See Figure 2 for an illustration. Recall that $|D_0| \leqslant 0.25|C_1|$ and $|C_2| \geqslant 0.4|C_1|$, which implies that $|D_0| \leqslant 0.5|C_2|$ and thus we have $|B| \geqslant 0.2|C_1|$. Furthermore, B is connected, by Lemma 31 $N_{G[C_1]}(C_1 - B) \subseteq L$ and by Lemma 30, the distance in $G[C_1]$ between K and B is more than $8t - 2 > 7t$.

Using the same argumentation as in the proof of Lemma 31, we have the following observation:

Lemma 32. *For every $C' \subseteq C$, there exists at most one connected component of $G[C']$ that contains both a vertex of B and a vertex adjacent to a vertex of K in $G[C_1]$.*

PROOF. If there are two such components D_0 and D_1, then going via D_0, K, D_1, and B, one obtains an induced cycle that is longer than t due to Lemma 30. □
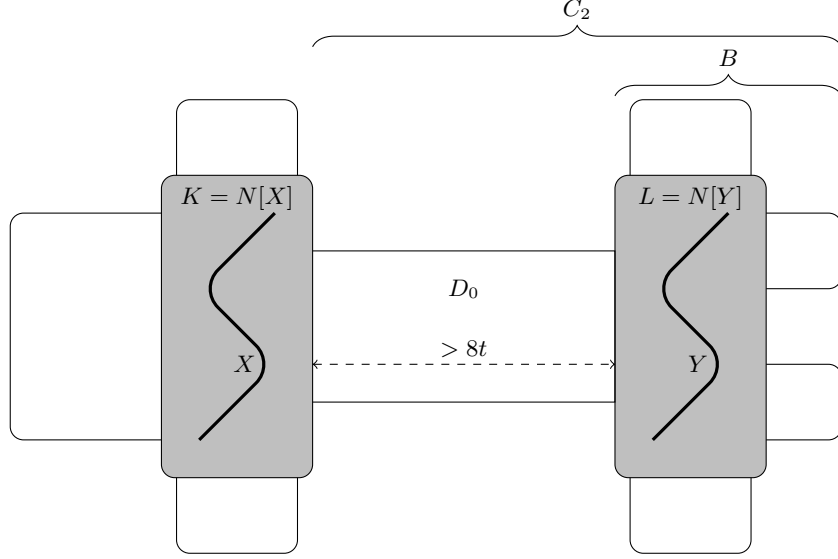
Figure 2: The component C_1 in the scenario when the secondary branching is invoked.

The secondary branching strategy is allowed only to make branch nodes (and subsequent filter and free nodes at success branches) and it always terminates whenever the current subproblem is splittable. The crux is to describe the choice of the branching pivot when the current subproblem is not splittable.

For an induced subgraph H of $G[W^{\mathcal{R}}]$, the *chip* in H is the vertex set C' of a connected component of $G[V(H) \cap C_2]$ that contains both a vertex of B and a vertex adjacent to a vertex of K. Lemma 32 implies that H contains at most one chip.

Assume that we are now considering a subproblem $\mathcal{R}'$ extending $\mathcal{R}$. Observe that if $H := G[W^{\mathcal{R}'}]$ has no chip, then $\mathcal{R}'$ is splittable as $|C_1| \geqslant 0.99n^{\mathcal{R}}$ and $|B| \geqslant 0.2|C_1|$. Hence, the secondary branching strategy terminates at $\mathcal{R}'$.

If H contains a chip C' with $|N_H(C')| = 1$, we choose the unique element of $N_H(C')$ as the branching pivot. Note that after branching on such a pivot, for every child z of the current branch node, both in the failure and in the success branches, (the remainder of) the chip C' in $G[W^{\mathcal{R}(z)}]$ is in a different connected component than any vertex of K. Hence, the failure child and the success grandchildren of x are splittable.

We are left with the following case: in H, there is exactly one chip C' and C' satisfies $|N_H(C')| \geqslant 2$. We define (secondary branching) buckets as follows. The buckets are indexed by an unordered pair $\{u, v\} \in \binom{N_H(C')}{2}$. For such a choice of u, v, the bucket $\mathcal{L}_{u,v}$ contains all C'-links with endpoints u and v. Lemma 29 ensures that every link in a bucket has at most t vertices and, consequently, every bucket has size $\mathcal{O}(n^t)$ and can be enumerated in polynomial time. Note that $\mathcal{L}_{u,v}$ is nonempty for every $\{u, v\} \in \binom{N_H(C')}{2}$.

For $\varepsilon > 0$, a vertex $x \in V(H)$ is ε-*heavy* if $N_H[x]$ intersects strictly more than an ε fraction of C'-links in at least an ε fraction of buckets. We prove the following.

Lemma 33. *There exists a $\frac{1}{2t}$-heavy vertex.*

PROOF. Let $H' := H[N_H[C']]$; note that $C' \subseteq N_H[C'] \subseteq C' \cup K$. Apply Theorem 6 to H' with $A = N_H(C')$, obtaining a set Y' of size at most t such that every connected component of $H' - N[Y']$ contains at most $|N_H(C')|/2$ vertices of $N_H(C')$. Consequently, $N_{H'}[Y']$ intersects all links in at least half of the

buckets $\mathcal{L}_{u,v}$. We infer that there is $y \in Y'$ such that $N_{H'}[y]$ intersects at least a $\frac{1}{t}$ fraction of links in at least $\frac{1}{2t}$ fraction of all nonempty buckets. This completes the proof. $\qquad\square$

Lemma 33 allows us to choose a $\frac{1}{2t}$-heavy vertex as the branching pivot.

It remains to show that with this choice of the branching pivot, the subproblem tree generated by the secondary branching strategy has $\mathcal{O}(\log^2 n)$ success branches on any root-to-leaf path.

As argued, for a branch node x, if there exists a chip C' in $H := G[W^{\mathcal{R}(x)}]$ with $|N_H(C')| = 1$, then the pivot is the unique element of $N_H(C')$ and the secondary branching strategy terminates both in the failure child and in all success grandchildren. Thus, on any root-to-leaf path there is at most one such branch node.

For every other branch node x, the pivot is a $\frac{1}{2t}$-heavy vertex. Let $H(x) := G[W^{\mathcal{R}(x)}]$ and let $C'(x)$ be the chip of $H(x)$. The *secondary level* of a node x is defined as

$$\lambda(x) := \left\lfloor \log_2\left(1 + \binom{|N_{H(x)}(C'(x))|}{2}\right)\right\rfloor.$$

Note that the secondary level is positive if and only if there is a nonempty bucket. Since at the root node there are $\mathcal{O}((n^{\mathcal{R}})^2)$ nonempty buckets, there are $\mathcal{O}(\log n^{\mathcal{R}})$ possible secondary levels. Furthermore, during the recursion the buckets can only shrink, so the secondary level of an ancestor is never lower than the secondary level of a descendant. Hence, it suffices to prove that any upward path in the subproblem tree on which all branching nodes have the same secondary level contains $\mathcal{O}(\log n^{\mathcal{R}})$ success branches.

To this end, consider the following potential for a node x.

$$\widehat{\mu}(x) := \sum_{\{u,v\} \in \binom{N_{H(x)}(C'(x))}{2}} \log_2\left[\sum_{Q \in \mathcal{L}_{u,v}^{G[W^{\mathcal{R}(x)}]}} \sum_{q \in Q}\left(1 + \gamma(\eta^{\mathcal{R}(x)}, q, \zeta^{\mathcal{R}(x)}(q))\right)\right].$$

For every node x of secondary level λ, there are between $2^\lambda - 1$ and $2^{\lambda+1} - 2$ nonempty buckets of $G[W^{\mathcal{R}(x)}]$. Hence, for node x of secondary level λ,

$$\widehat{\mu}(x) \leqslant (2^{\lambda+1} - 2) \cdot \mathcal{O}(\log n^{\mathcal{R}}) = 2(2^\lambda - 1) \cdot \mathcal{O}(\log n^{\mathcal{R}}).$$

Let x be a branch node and y its successful grandchild, both with the same secondary level. Noting that the innermost sums in the definition of $\widehat{\mu}$ are lower-bounded by 1 and upper-bounded by $(d+1)t$, from the definition of a $\frac{1}{2t}$-heavy vertex and Lemma 20 we infer that

$$\widehat{\mu}(x) - \widehat{\mu}(y) \geqslant \frac{c}{dt^3} \cdot (2^\lambda - 1),$$

for some universal constant $c > 0$. Since the potential μ never becomes negative, it can decrease only $\mathcal{O}(\log n^{\mathcal{R}})$ times at a successful branch when the secondary level λ is fixed. Since there are $\mathcal{O}(\log n^{\mathcal{R}})$ secondary levels in total, we conclude that every root-to-leaf path contains $\mathcal{O}(\log^2 n^{\mathcal{R}})$ success branches.

This completes the proof of the properties of the secondary branching strategy and thus of Lemma 19.

6 $C_{>t}$-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 [16]. 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 3.

Theorem 3. *For every pair of integers d and t, there exists an integer $k = (dt)^{\mathcal{O}(t)}$ 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 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.* [19], 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 34 (Hatzel *et al.* [19]). *There exists a polynomial $\mathsf{p}(\cdot)$ such that for every positive integer k, every graph G of treewidth at least k contains a bramble $\mathcal{B}$ of order at least $\sqrt{k}/\mathsf{p}(\log k)$ such that each vertex of G is in at most two branch sets of $\mathcal{B}$.*

We now proceed to the proof of Theorem 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

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

where $\mathsf{p}(\cdot)$ is the polynomial provided by Theorem 34. Thus, by applying Theorem 34 to G we obtain a bramble $\mathcal{B} = (B_1, B_2, \ldots, B_p)$ of order

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

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 34, 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. [26, Lemma 2.19 and Corollary 2.20].

Proposition 35. *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 35, 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

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

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.

36

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 $0 \leqslant i \leqslant t/2$, $v_i v_{i+1}$ 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 $v_i v_{i+1}$ 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 $v_i v_{i+1} \in E(Z)$ 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 $v_i v_{i+1} \in E(Z)$ 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 $v_j v_{j+1} \notin E(Z)$ 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 $v_j v_{j+1} \in E(Z)$ 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.

7 MSO$_2$ and $C_{>t}$-free graphs

In this section we prove Theorem 1 using the branching strategy described in Section 4.

7.1 Extending subproblems

Fix integers $k \geqslant 1$ and $t \geqslant 3$, CMSO_2 sentence φ, and a $C_{>t}$-free graph G. Let q be the maximum of the quantifier rank of φ, the quantifier rank of $\varphi_{\mathrm{tw}<k}$, where $\varphi_{\mathrm{tw}<k}$ comes from Lemma 10, and 4 (so that we can use Lemma 9). Let p be the largest modulus used in φ, or 0 if φ does not contain any modular atomic expression. We also denote $k' \coloneqq 6k$. In what follows, we will be using CMSO_2 types $\mathsf{Types}^{k',p,q}$, as given by Proposition 8. Hence, by "CMSO_2 types" we mean elements of $\mathsf{Types}^{k',p,q}$, and we drop the super- or subscript k', p, q when it is clear from the context.

Recall that every graph of treewidth less than k is $(k-1)$-degenerate. Hence, we will use the branching strategy provided Lemma 19 for $d \coloneqq k - 1$ and t. More precisely, the algorithm executes the branching, i.e. decides on the types of nodes, chooses branching pivots, etc., exactly as prescribed by the strategy given by Lemma 19. However, we enrich the subproblems with some additional piece of information, which intuitively encodes a skeleton of a decomposition of the subgraph induced by the constructed solution. From the recursive subcalls, we expect returning quite an elaborate result: intuitively, optimum solutions to the subproblem of every possible CMSO_2 type. As we will check at the end, the properties asserted by Lemma 19 ensure a quasipolynomial running time bound.

Let us start by formally augmenting the notion of a subproblem with extra information. At every leaf, split, and branch node x, the subproblem $\mathcal{R} \coloneqq \mathcal{R}(x)$ contains additionally the following:

1. a tree decomposition $(T^{\mathcal{R}}, \beta^{\mathcal{R}})$ of $G[A^{\mathcal{R}}]$ with maximum bag size at most $k' = 6k$ and the root node having an empty bag;

2. a labelling $\iota^{\mathcal{R}} \colon A^{\mathcal{R}} \to [k']$ such that for every $a \in V(T^{\mathcal{R}})$, $\iota^{\mathcal{R}}$ restricted to $\beta^{\mathcal{R}}(a)$ is injective.

We call such a subproblem an *extended subproblem.* Leaf, split, and branch nodes are called *extendable nodes.*

For such an extended subproblem $\mathcal{R}$, a set $S \subseteq W^{\mathcal{R}}$ is a *feasible solution* if for every connected component C of $G[S]$, we have $|N_G(C) \cap A^{\mathcal{R}}| \leqslant 4k$ and there exists a node $a \in V(T^{\mathcal{R}})$ such that $N_G(C) \cap A^{\mathcal{R}} \subseteq \beta^{\mathcal{R}}(a)$. Observe that from the properties of a tree decomposition it follows that the family of those nodes $a \in V(T^{\mathcal{R}})$ for which $N_G(C) \cap A^{\mathcal{R}} \subseteq \beta^{\mathcal{R}}(a)$ is a connected subtree of T. By $a^{\mathcal{R}}(C)$ we denote the highest such node a.

For an extended subproblem $\mathcal{R}$, a *type assignment* is a function $\mathsf{TypeTree} \colon V(T^{\mathcal{R}}) \to \mathsf{Types}$. A feasible solution S is *of type* $\mathsf{TypeTree}$ if for every $a \in V(T^{\mathcal{R}})$, the subgraph of G induced by

$$\mathsf{extS}^{\mathcal{R}}(a, S) \coloneqq \beta^{\mathcal{R}}(a) \cup \bigcup \{ C \in \mathrm{cc}(G[S]) \mid a^{\mathcal{R}}(C) = a \}$$

equipped with the labelling $\iota^{\mathcal{R}}|_{\beta^{\mathcal{R}}(a)}$ on boundary $\beta^{\mathcal{R}}(a)$ is of CMSO_2 type $\mathsf{TypeTree}(a)$.

The branching strategy will return, at every extendable node x, for every type assignment $\mathsf{TypeTree}$ at x, a feasible solution $\mathsf{S}[x, \mathsf{TypeTree}]$ of type $\mathsf{TypeTree}$. We allow $\mathsf{S}[x, \mathsf{TypeTree}] = \bot$, indicating that no such feasible solution was found, and we use the convention that the weight of $\bot$ is $-\infty$. In the algorithm description the following operation will be useful when defining $\mathsf{S}[x, \cdot]$ for a fixed node x: given a current state of the table $\mathsf{S}[x, \cdot]$ and a feasible solution S of type $\mathsf{TypeTree}$, *updating* $\mathsf{S}[x, \mathsf{TypeTree}]$ *with* S is an operation that sets $\mathsf{S}[x, \mathsf{TypeTree}] \coloneqq S$ if the weight of S is larger than the weight of the former value of $\mathsf{S}[x, \mathsf{TypeTree}]$.

In the root of the recursion r, the considered extension is the trivial one: the tree decomposition $(T^{\mathcal{R}(r)}, \beta^{\mathcal{R}(r)})$ consists of a single node a^r with an empty bag, and labelling $\iota^{\mathcal{R}(r)}$ is empty. After the computation if finished, we iterate over all type assignments $\mathsf{TypeTree}$ for r for which $\mathsf{TypeTree}(a^r) \in \mathsf{type}[\varphi] \cap \mathsf{type}[\varphi_{\mathrm{tw}<k}]$ and return the set $\mathsf{S}[r, \mathsf{TypeTree}]$ of maximum weight found (ignoring values $\bot$). If no such set $\mathsf{S}[r, \mathsf{TypeTree}]$ is found (all values are $\bot$), the algorithm returns that there is no such set S.

Note that, assuming the recursive strategy indeed maintains the invariant that $S[x, \mathsf{TypeTree}]$ is of type $\mathsf{TypeTree}$, for the returned set S we have $G[S] \models \varphi$ and $G[S]$ is of treewidth less than k.

7.2 Extended computation at nodes of the subproblem tree

Recall that now, the setting is that each extendable node (lead, split, or branch node) is assigned an extended subproblem. We need to describe (a) how at each node we handle the extended subproblem and what extended subproblems are passed down the subproblem tree; (b) what is the subroutine for handling free nodes; and (c) how the tables $S[\cdot, \cdot]$ are computed along the recursion. As before, each type of a node is handled differently.

7.2.1 Leaf nodes

For a leaf node x, there is little choice the algorithm could do.

We have $W^{\mathcal{R}(x)} = \emptyset$, so the only feasible solution is $S = \emptyset$. There is exactly one type assignment $\mathsf{TypeTree}$ such that for every $a \in V(T^{\mathcal{R}(x)})$ we have that $G[\mathsf{extS}^{\mathcal{R}(x)}(a, \emptyset)]$ with the labelling $\iota^{\mathcal{R}}|_{\beta^{\mathcal{R}}(a)}$ is of type $\mathsf{TypeTree}(a)$. For this labelling, we set $S[x, \mathsf{TypeTree}] = \emptyset$ and for every other type assignment $\mathsf{TypeTree}'$ we set $S[x, \mathsf{TypeTree}'] = \bot$.

7.2.2 Split nodes

Let x be a split node. We do not use the power of free nodes below split nodes; formally, they are dummy free nodes as in the proof of [Theorem 11](#) that only pass the subproblem to their single child. Therefore, in what follows we speak about grandchildren of a split node.

If x has one grandchild y, then $\mathcal{R}(y)$ differs from $\mathcal{R}(x)$ only by its level. Therefore, given an extension of $\mathcal{R}(x)$, we pass the same extension to the grandchild y and for the return value at x, we copy the result returned by the grandchild y.

The situation is more interesting if x has two grandchildren y_1 and y_2. Recall that then $W^{\mathcal{R}(y_1)}$ and $W^{\mathcal{R}(y_2)}$ is a partition of $W^{\mathcal{R}(x)}$, while $A^{\mathcal{R}(y_1)} = A^{\mathcal{R}(y_2)} = A^{\mathcal{R}(x)}$. Given a subproblem extension of $\mathcal{R}(x)$, we pass it without modifications to both y_1 and y_2. To compute $S[x, \cdot]$ based on $S[y_1, \cdot]$ and $S[y_2, \cdot]$, we proceed as follows.

First, we initiate $S[x, \mathsf{TypeTree}] = \bot$ for every tree assignment $\mathsf{TypeTree}$ at x. Then, we iterate over all pairs $\mathsf{TypeTree}_1$ and $\mathsf{TypeTree}_2$ of type assignments for $\mathcal{R}(x)$ such that both $S[y_1, \mathsf{TypeTree}_1] \neq \bot$ and $S[y_2, \mathsf{TypeTree}_2] \neq \bot$. Let $\mathsf{TypeTree}$ be the type assignment for $\mathcal{R}(x)$ defined as

$$\mathsf{TypeTree}(a) = \mathsf{TypeTree}_1(a) \oplus \mathsf{TypeTree}_2(a) \qquad \text{for every } a \in T^{\mathcal{R}(x)}.$$

Then observe that as $W^{\mathcal{R}(y_1)}$ is nonadjacent to $W^{\mathcal{R}(y_2)}$, $S[y_1, \mathsf{TypeTree}_1] \cup S[y_2, \mathsf{TypeTree}_2]$ is a feasible solution for x of type $\mathsf{TypeTree}$. We update $S[x, \mathsf{TypeTree}]$ with $S[y_1, \mathsf{TypeTree}_1] \cup S[y_2, \mathsf{TypeTree}_2]$.

7.2.3 Branch and subsequent free nodes

Let x be a branch node and recall that we are given a subproblem extension of $\mathcal{R}(x)$. For the failure child y^x of x, we pass this subproblem extension to y^x without modifications.

Consider now a success grandchild $s := s^x_{\mathcal{D}}$ for $\mathcal{D} = (D, \eta, (D_u)_{u \in D'})$, $D' = D \cup \{\nu^x\}$. We now use the power of the free node $s^x_{\mathcal{D}}$ to guess how the extension at x should be enhanced.

Formally, we iterate over every possibility of:

- a partition $\mathcal{B}$ of D' into nonempty subsets;

- a node $a_B \in V(T^{\mathcal{R}(x)})$ and a subset $N_B \subseteq \beta^{\mathcal{R}(x)}(a_B)$ of size at most $4k$ for every $B \in \mathcal{B}$;

- a set $X_B \subseteq W^{\mathcal{R}(s)}$ of size at most k for every $B \in \mathcal{B}$ so that the sets $(X_B)_{B \in \mathcal{B}}$ are pairwise vertex-disjoint; we denote $X_\mathcal{B} := \bigcup_{B \in \mathcal{B}} X_B$;

- a position guess $\eta^\mathcal{B}$ for $X_\mathcal{B}$ and a left neighbor guess $(D_u^\mathcal{B})_{u \in X_\mathcal{B}}$ for $X_\mathcal{B}$ and $\eta^\mathcal{B}$.

For every choice $\mathcal{C} = (\mathcal{B}, (a_B, N_B, X_B)_{B \in \mathcal{B}}, \eta^\mathcal{B}, (D_u^\mathcal{B})_{u \in X_\mathcal{B}})$ as above, we construct a child $\hat{s}_{\mathcal{D},\mathcal{C}}^x$ of $s_\mathcal{D}^x$ that is created from $s_\mathcal{D}^x$ by taking $X_\mathcal{B}$ at positions $\eta^\mathcal{B}$ with left neighbors $(D_u^\mathcal{B})_{u \in X_\mathcal{B}}$. Every created child $\hat{s}_{\mathcal{D},\mathcal{C}}^x$ is a filter node; we denote the resulting grandchild (if present) as $\tilde{s}_{\mathcal{D},\mathcal{C}}^x$.

Denoting $\tilde{s} = \tilde{s}_{\mathcal{D},\mathcal{C}}^x$ for brevity, the grandchild's subproblem is extended as follows:

- create $(T^{\mathcal{R}(\tilde{s})}, \beta^{\mathcal{R}(\tilde{s})})$ from $(T^{\mathcal{R}(x)}, \beta^{\mathcal{R}(x)})$ by adding, for every $B \in \mathcal{B}$, a new node $\tilde{a}_B$ with bag $N_B \cup B \cup X_B$ that is a child of a_B;

- create $\iota^{\mathcal{R}(\tilde{s})}$ by extending $\iota^{\mathcal{R}(x)}$ to the new elements of $W^{\mathcal{R}(\tilde{s})}$ in any manner that is injective on the newly created bags.

Observe that every newly created bag is of size bounded by $4k + k + k = k'$. Since $\mathcal{B}$ is a partition of D' and we assume that the sets $(X_B)_{B \in \mathcal{B}}$ are pairwise disjoint, for every two newly created nodes a_{B_1}' and a_{B_2}', we have $\beta^{\mathcal{R}(\tilde{s})}(a_{B_1}') \cap \beta^{\mathcal{R}(\tilde{s})}(a_{B_2}') \subseteq N_{B_1} \cap N_{B_2} \subseteq A^{\mathcal{R}(x)}$, the pair $(T^{\mathcal{R}(\tilde{s})}, \beta^{\mathcal{R}(\tilde{s})})$ is indeed a tree decomposition of $G[A^{\mathcal{R}(\tilde{s})}]$ with maximum bag size at most k', and it is straightforward to extend $\iota^{\mathcal{R}(x)}$ to obtain $\iota^{\mathcal{R}(\tilde{s})}$.

To complete the description of the algorithm, it remains to show how to assemble the table $\mathsf{S}[x, \cdot]$ at the branch node x from the tables computed for its great-great-grandchildren $\tilde{s}_{\mathcal{D},\mathcal{C}}^x$ and the failure child y^x.

To this end, we initiate $\mathsf{S}[x, \mathsf{TypeTree}] = \mathsf{S}[y^x, \mathsf{TypeTree}]$ for every type assignment $\mathsf{TypeTree}$ at x (note that every feasible solution at y^x is also a feasible solution at x and is of the same type). Then, for every great-great-grandchild $\tilde{s} := \tilde{s}_{\mathcal{D},\mathcal{C}}^x$ and type assignment $\widetilde{\mathsf{TypeTree}}$ at $\tilde{s}$ such that $\tilde{S} := \mathsf{S}[\tilde{s}, \widetilde{\mathsf{TypeTree}}] \neq \bot$, we proceed as follows. Let $\mathcal{D} = (D, \eta, (D_u)_{u \in D'})$, $D' = D \cup \{\nu^x\}$, $\mathcal{C} = (\mathcal{B}, (a_B, N_B, X_B)_{B \in \mathcal{B}}, \eta^\mathcal{B}, (D_u^\mathcal{B})_{u \in X_\mathcal{B}})$, and $X_\mathcal{B} = \bigcup_{B \in \mathcal{B}} X_B$. Observe that $A^{\mathcal{R}(\tilde{s})} - A^{\mathcal{R}(x)} = D' \cup X_\mathcal{B}$. Let $S := \tilde{S} \cup D' \cup X_\mathcal{B}$.

We say that the pair $(\tilde{s}, \tilde{S})$ is *liftable* if the set S obtained as above is a feasible solution at x and, furthermore, for every $B \in \mathcal{B}$ there exists a connected component $\tilde{C}_B$ of $G[S]$ such that

$$\tilde{C}_B = B \cup X_B \cup \bigcup \{\tilde{C} \in \mathrm{cc}(G[\tilde{S}]) \mid a^{\tilde{s}}(\tilde{C}) = \tilde{a}_B\},$$

the components $(\tilde{C}_B)_{B \in \mathcal{B}}$ are pairwise distinct, and $a^x(\tilde{C}_B) = a_B$ for every $B \in \mathcal{B}$. For a liftable pair $(\tilde{s}, \tilde{S})$, the set S is called the *lift of* $(\tilde{s}, \tilde{S})$.

If $(\tilde{s}, \tilde{S})$ is liftable, then we can use Proposition 8 to compute the type $\mathsf{TypeTree}_S$ of the lift S at x as follows. First, for every $B \in \mathcal{B}$, the CMSO_2 type of $G[C_B \cup \beta^{\mathcal{R}(x)}(a_B)]$ with labelling $\iota^{\mathcal{R}(x)}|_{\beta^{\mathcal{R}(x)}(a_B)}$ can be computed from $\widetilde{\mathsf{TypeTree}}(\tilde{a}_B)$ by forgetting the labels of $B \cup X_B$. Second, for each $a \in V(T^{\mathcal{R}(x)})$, the type $\mathsf{TypeTree}_S(a)$ is the composition of $\widetilde{\mathsf{TypeTree}}(a)$ and the types of all graphs $(G[C_B \cup \beta^{\mathcal{R}(x)}(a_B)], \iota^{\mathcal{R}(x)}|_{\beta^{\mathcal{R}(x)}(a_B)})$ for those B for which $a_B = a$.

For every liftable pair $(\tilde{s}, \tilde{S})$, we compute the lift S and its type $\mathsf{TypeTree}_S$ at x as above and we update $\mathsf{S}[x, \mathsf{TypeTree}_S]$ with S. This finishes the description of the algorithm at branch nodes.

We conclude this section with an immediate, yet important corollary of the way how we compute the type $\mathsf{TypeTree}_S$.

Lemma 36. *Let x be a branch node, $\tilde{s}$ be its great-great-grandchild, and $\tilde{S}_1$ and $\tilde{S}_2$ be two feasible solutions at $\tilde{s}$ of the same type such that both $(\tilde{s}, \tilde{S}_1)$ and $(\tilde{s}, \tilde{S}_2)$ are liftable. Let S_i be the lift of $(\tilde{s}, \tilde{S}_i)$ for $i = 1, 2$. Then S_1 and S_2 are of the same type at x.*

PROOF. The aforementioned algorithm to compute the type of the lift of $(\tilde{s}, \tilde{S}_i)$ uses only the type of $\tilde{S}_i$ at $\tilde{s}$ and the subproblems $\mathcal{R}(x)$ and $\mathcal{R}(\tilde{s})$. The claim follows. □

7.3 Correctness

Fix a subset $S^* \subseteq V(G)$ such that $G[S^*]$ is of treewidth less than k and $G[S^*] \models \varphi$. We would like to show that the algorithm returns a set S of weight at least the weight of S^* (not necessarily S^*). Clearly, since $G[S^*]$ is $(k-1)$-degenerate, we can speak about lucky nodes of the subproblem tree, defined in the same manner as in Section 4.

A lucky extendable node x is called a *gander* if $S^* \cap W^{\mathcal{R}(x)}$ is a feasible solution for $\mathcal{R}(x)$. For a gander x, define the type assignment $\mathsf{TypeTree}^x$ as the type of the feasible solution $S^* \cap W^{\mathcal{R}(x)}$.

Note that the root r of the subproblem tree is a gander and $\mathsf{TypeTree}^r(a^r) \in \mathsf{type}[\varphi] \cap \mathsf{type}[\varphi_{\mathrm{tw}<k}]$. Thus, it suffices to show the following:

Lemma 37. *For every gander x, we have that $\mathsf{S}[x, \mathsf{TypeTree}^x] \neq \bot$ and the weight of $\mathsf{S}[x, \mathsf{TypeTree}^x]$ is at least the weight of $S^* \cap W^{\mathcal{R}(x)}$.*

PROOF. The proof proceeds by a bottom-up induction on the subproblem tree.

For a leaf gander x, since x is lucky, from the definition of $\mathsf{TypeTree}^x$ the only type assignment $\mathsf{TypeTree}$ for which $\mathsf{S}[x, \mathsf{TypeTree}] \neq \bot$ is exactly $\mathsf{TypeTree}^x$. Consequently, $\mathsf{S}[x, \mathsf{TypeTree}^x] = \emptyset$ and the claim is proven.

For a split gander x, the claim is straightforward if x has one grandchild. Assume then x has two grandchildren y_1 and y_2. Since $W^{\mathcal{R}(y_1)}$ is nonadjacent to $W^{\mathcal{R}(y_2)}$, every connected component of $G[S^* \cap W^{\mathcal{R}(x)}]$ is contained either in $W^{\mathcal{R}(y_1)}$ or in $W^{\mathcal{R}(y_2)}$. It follows that both y_1 and y_2 are ganders, too.

By induction, for $i = 1, 2$ the weight of $\mathsf{S}[y_i, \mathsf{TypeTree}^{y_i}]$ is at least the weight of $S^* \cap W^{\mathcal{R}(y_i)}$. Denote

$$\mathsf{TypeTree}(a) := \mathsf{TypeTree}^{y_1}(a) \oplus \mathsf{TypeTree}^{y_2}(a) \qquad \text{for } a \in V(T^{\mathcal{R}(x)}).$$

Then, on one hand from Proposition 8 we have that $\mathsf{TypeTree} = \mathsf{TypeTree}^x$, and on the other hand the handling of split nodes will update $\mathsf{S}[x, \mathsf{TypeTree}]$ with $\mathsf{S}[y_1, \mathsf{TypeTree}^{y_1}] \cup \mathsf{S}[y_2, \mathsf{TypeTree}^{y_2}]$. The claim for split ganders follows.

It remains to analyse branch ganders. Let x be a branch gander. If $\nu^x \notin S^*$, then y^x is lucky and it is immediate that it is a gander, too. By the inductive assumption, $\mathfrak{w}(\mathsf{S}[y^x, \mathsf{TypeTree}^{y^x}]) \geqslant \mathfrak{w}(S^* \cap W^{\mathcal{R}(y^x)})$. Since in this case $\mathsf{TypeTree}^x = \mathsf{TypeTree}^{y^x}$ and we initiated $\mathsf{S}[x, \mathsf{TypeTree}^x]$ with $\mathsf{S}[y^x, \mathsf{TypeTree}^x]$, the inductive claim follows.

Assume then that $\nu^x \in S^*$. Define D, η, and $(D_u)_{u \in D'}$ for $D' = D \cup \{\nu^x\}$ as in Lemma 17, that is:

$$D = \{u \in N_G(\nu^x) \cap W^{\mathcal{R}(x)} \cap S^* \mid \eta^*(u) < \eta^*(\nu^x)\},$$
$$\eta = \eta^*|_{A^{\mathcal{R}(x)} \cup D'},$$
$$D_u = \{w \in N_G(u) \cap W^{\mathcal{R}(x)} \cap S^* - D' \mid \eta^*(w) < \eta^*(u)\}.$$

Then, by Lemma 17, the branch node x creates a child $z_{\mathcal{D}}^x$ for $\mathcal{D} = (D, \eta, (D_u)_{u \in D'})$ and $z_{\mathcal{D}}^x$ is lucky. Hence, the grandchild $s_{\mathcal{D}}^x$ exists and is lucky as well.

41

We now want to look at some particular child of the free node $s_{\mathcal{D}}^x$. Define the partition $\mathcal{B}$ of D' as the partition induced by the connected components of $G[S^* \cap W^{\mathcal{R}(x)}]$ on D', that is:

$$\mathcal{B} = \{C \cap D' \colon C \in \text{cc}(G[S^* \cap W^{\mathcal{R}(x)}]) \text{ such that } C \cap D' \neq \emptyset\}.$$

For every $B \in \mathcal{B}$, let C_B be the connected component of $G[S^* \cap W^{\mathcal{R}(x)}]$ that contains B. Define $a_B := a^x(C_B)$ for $B \in \mathcal{B}$ and $N_B := N_G(C_B) \cap A^{\mathcal{R}(x)}$. Since x is a gander, we have $|N_B| \leqslant 4k$ and $N_B \subseteq \beta^{\mathcal{R}(x)}(a_B)$. Let $X_B^{\circ} \subseteq C_B \cup N_B$ be a set of size at most k promised by Lemma 7 for the graph $G[C_B \cup N_B]$ and $A = N_B$; note here that $G[C_B \cup N_B]$ is of treewidth less than k, because $C_B, N_B \subseteq S^*$. Let $X_B := X_B^{\circ} \cap C_B$. Finally, define $\eta^{\mathcal{B}}$ and $(D_u^{\mathcal{B}})_{u \in X_{\mathcal{B}}}$ similarly as for the grandchildren of x:

$$\eta^{\mathcal{B}} = \eta^*|_{A^{\mathcal{R}(x)} \cup D' \cup X_{\mathcal{B}}},$$
$$D_u^{\mathcal{B}} = \{w \in N_G(u) \cap W^{\mathcal{R}(x)} \cap S^* - (D' \cup X_{\mathcal{B}}) \mid \eta^*(w) < \eta^*(u)\}.$$

Let $\mathcal{C} = (\mathcal{B}, (a_B, N_B, X_B)_{B \in \mathcal{B}}, \eta^{\mathcal{B}}, (D_u^{\mathcal{B}})_{u \in X_{\mathcal{B}}})$. Observe that $\hat{s} := \hat{s}_{\mathcal{D},\mathcal{C}}^x$ is lucky by Lemma 13, as $X_{\mathcal{B}} \subseteq S^*$. Hence, $\tilde{s} := \tilde{s}_{\mathcal{D},\mathcal{C}}^x$ exists and is lucky as well. We claim that $\tilde{s}$ is a gander.

To this end, consider a connected component $\tilde{C}$ of $G[S^* \cap W^{\mathcal{R}(\tilde{s})}]$. We want to show that $|N_G(\tilde{C}) \cap S^*| \leqslant 4k$ and there exists a node $\tilde{a} \in V(T^{\mathcal{R}(\tilde{s})})$ such that $N_G(\tilde{C}) \cap S^* \subseteq \beta^{\mathcal{R}(\tilde{s})}(\tilde{a})$. The claim is straightforward if $\tilde{C}$ is also a connected component of $G[S^* \cap W^{\mathcal{R}(x)}]$, because x is a gander. Otherwise, $\tilde{C} \subsetneq C$ for some connected component C of $G[S^* \cap W^{\mathcal{R}(x)}]$. Note that there exists $B \in \mathcal{B}$ such that $B \subseteq C$, that is, $C = C_B$ for some $B \in \mathcal{B}$.

By the choice of X_B°, the connected component of $G[C_B \cup N_B] - X_B^{\circ}$ that contains $\tilde{C}$ contains at most $|N_B|/2 \leqslant 2k$ vertices of N_B. Consequently,

$$|N_G(\tilde{C}) \cap S^*| \leqslant |X_B| + |B| + |N_B|/2 \leqslant k + k + 2k = 4k.$$

Furthermore,

$$N_G(\tilde{C}) \cap S^* \subseteq X_B \cup B \cup N_B \subseteq \beta^{\mathcal{R}(\tilde{s})}(\tilde{a}_B).$$

We infer that $\tilde{s}$ is indeed a gander.

Since $S^* \cap W^{\mathcal{R}(\tilde{s})}$ is a feasible solution at $\tilde{s}$, it follows immediately from the definition of $\mathcal{C}$ that $(\tilde{s}, S^* \cap W^{\mathcal{R}(\tilde{s})})$ is liftable with the lift $S^* \cap W^{\mathcal{R}(x)}$.

Let now $\tilde{S} = \mathsf{S}[\tilde{s}, \mathsf{TypeTree}^{\tilde{s}}]$ and $S = \tilde{S} \cup D' \cup X_{\mathcal{B}}$; note that $A^{\mathcal{R}(\tilde{s})} - A^{\mathcal{R}(x)} = D' \cup X_{\mathcal{B}}$.

We claim that $(\tilde{s}, \tilde{S})$ is liftable. To this end, fix $B \in \mathcal{B}$. Observe that:

$$\mathsf{extS}^{\tilde{s}}(\tilde{a}_B, S^* \cap W^{\mathcal{R}(\tilde{s})}) = \beta^{\mathcal{R}(\tilde{s})}(a) \cup C_B.$$

Consider the connected component C_B of $G[S^* \cap C^{\mathcal{R}(x)}]$: it contains $X_B \cup B = \beta^{\mathcal{R}(\tilde{s})}(a_B) - \beta^{\mathcal{R}(\tilde{s})}(\tilde{a}_B)$, which is nonempty due to $B \neq \emptyset$, and its neighborhood in $A^{\mathcal{R}(x)}$ is the set $N_B \subseteq \beta^{\mathcal{R}(x)}$. Observe that the graph induced by $\mathsf{extS}^{\tilde{s}}(\tilde{a}_B, \tilde{S})$ is of the same type as the graph induced by $\mathsf{extS}^{\tilde{s}}(\tilde{a}_B, S^* \cap W^{\mathcal{R}(\tilde{s})})$ (both with the boundary labelling $\iota^{\tilde{s}}|_{\beta^{\mathcal{R}(\tilde{s})}(\tilde{a}_B)}$). Therefore, due to $X_B \cup B$ being nonempty, by Lemma 9 we infer that there exists a connected component $\tilde{C}_B$ of $G[S]$ such that

$$\tilde{C}_B = X_B \cup B \cup \bigcup \{C' \in \text{cc}(G[\tilde{S}]) \mid a^{\tilde{s}}(C') = \tilde{a}_B\}$$

and

$$N_G(\tilde{C}_B) \cap A^{\mathcal{R}(x)} = N_B = N_G(C_B) \cap A^{\mathcal{R}(x)}. \tag{4}$$

Consequently, every connected component C of $G[S]$ that is disjoint with D' is also a connected component of $G[\tilde{S}]$, and hence $N_G(C) \cap A^{\mathcal{R}(\tilde{s})} = N_G(C) \cap A^{\mathcal{R}(x)}$. Since $\tilde{S}$ is a feasible solution at $\tilde{s}$, we infer that S is a feasible solution at x. Furthermore, from (4) if follows that for every $B \in \mathcal{B}$ we have $a^x(\tilde{C}_B) = a$. Hence, $(\tilde{s}, \tilde{S})$ is liftable and S is the lift.

By induction, the weight of $\tilde{S}$ is not smaller than the weight of $S^* \cap W^{\mathcal{R}(\tilde{s})}$. As $D' \cup X_{\mathcal{B}} \subseteq S^*$, the weight of S is not smaller than the weight of $S^* \cap W^{\mathcal{R}(x)}$. Since S is liftable, the algorithm updates $\mathsf{S}[x, \mathsf{TypeTree}_S]$ with S, where $\mathsf{TypeTree}_S$ is the type assignment of S at node x.

Since $\tilde{S}$ and $S^* \cap W^{\mathcal{R}(\tilde{s})}$ are of the same type $\mathsf{TypeTree}^{\tilde{s}}$ at $\tilde{s}$, it follows from Lemma 36 that S and $S^* \cap W^{\mathcal{R}(x)}$ are of the same type at x, that is, $\mathsf{TypeTree}_S = \mathsf{TypeTree}^x$. This finishes the induction step for branch ganders and completes the proof of the lemma. $\square$

7.4 Complexity analysis

We are left with arguing that the time complexity is as promised. By Lemma 19, the subproblem tree generated by the recursion has depth $\mathcal{O}(n)$, $\mathcal{O}(\log^2 n)$ or $\mathcal{O}(\log^3 n)$ success branches on any root-to-leaf path depending on whether we work in P_t-free or $C_{>t}$-free regime, and $\mathcal{O}(\log n)$ split nodes on any root-to-leaf path.

Note that success branches are the only places where we add nodes to the tree decomposition $(T^{\mathcal{R}}, \beta^{\mathcal{R}})$. Furthermore, a success branch adds at most k nodes to the tree decomposition, one for each element of $\mathcal{B}$. Hence, at every node x we have $|V(T^{\mathcal{R}(x)})| = \mathcal{O}(\log^3 n)$ and $|V(T^{\mathcal{R}(x)})| = \mathcal{O}(\log^2 n)$ if G is P_t-free. As $|\mathsf{Types}| = \mathcal{O}(1)$, there are $2^{\mathcal{O}(\log^3 n)}$ type assignments to consider ($2^{\mathcal{O}(\log^2 n)}$ if G is P_t-free).

At a free node that is a grandchild of a branch node, the sets D', $\mathcal{B}$, X_B, N_B are of constant size. Consequently, every free node has a number of children bounded polynomially in n.

We infer that the whole subproblem tree has size $n^{\mathcal{O}(\log^3 n)}$. At every node, the algorithm spends time $n^{\mathcal{O}(\log^3 n)}$ inspecting all type assignments (or pairs of type assignments in the case of a split node). Both bounds improve to $n^{\mathcal{O}(\log^2 n)}$ if G is P_t-free. The running time bound follows, and hence the proof of Theorem 1 is complete.

7.5 A generalization

We now give a slight generalization of Theorem 1 that can be useful for expressing some problems that do not fall directly under its regime. The idea is that together with the solution S we would like to distinguish a subset $M \subseteq S$ that satisfies some CMSO_2-expressible predicate, and only the vertices of M contribute to the weight of the solution. The proof is a simple gadget reduction to Theorem 1.

Theorem 38. *Fix a pair of integers d and t and a CMSO_2 formula $\varphi(X)$ with one free vertex subset variable. 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 $n^{\mathcal{O}(\log^3 n)}$ finds subsets of vertices $M \subseteq S$ such that $G[S]$ is d-degenerate, $G[S] \models \varphi(M)$, and, subject to the above, $\mathfrak{w}(M)$ is maximum possible; the algorithm may also conclude that no such vertex subsets exist. The running time can be improved to $n^{\mathcal{O}(\log^2 n)}$ if G is P_t-free.*

PROOF. For a graph G and a set $M \subseteq V(G)$, the *forked version* of (G, M) is the graph $\check{G}^M$ created from G by attaching three degree-one neighbors to every vertex of $V(G)$ and, additionally, a two-edge path to every vertex of M. If G is weighted, then we assign weights to the vertices of $\check{G}^M$ so that all of them are zero, except that for every $v \in M$ the other endpoint of the attached two-edge path inherits the weight of v.

Note that the vertices of $V(\breve{G}^M) - V(G)$ are exactly the vertices of degree one or two in $\breve{G}^M$; the vertices of $V(G)$ are of degree at least three in $\breve{G}^M$. This implies that if H' is a forked version of some other graph H and $M \subseteq V(H)$, then the pair (H, M) is defined uniquely and is easy to decode:

- The vertices of H are exactly the vertices of H' that are of degree at least three, and H is the induced subgraph of H' induced by those vertices.

- Every vertex of H needs to be adjacent to exactly three vertices of degree 1 or to three vertices of degree 1 and one vertex of degree 2, which in turn has another neighbor of degree 1. The vertices of the latter category are exactly the vertices of M.

Note that if G is $C_{>t}$-free for some $t \geqslant 3$, then $\breve{G}^M$ is $C_{>t}$-free as well. Moreover, $\mathrm{tw}(\breve{G}^M) \leqslant \max(\mathrm{tw}(G), 1)$.

Let k, t, and φ be as in Theorem 1. Construct a CMSO$_2$ sentence $\breve{\varphi}$ that for a graph H' behaves as follows:

- If H' is a forked version of some pair (H, M), then $H' \models \breve{\varphi}$ if and only if $H \models \varphi(M)$.

- Otherwise, $H' \not\models \breve{\varphi}$.

Writing $\breve{\varphi}$ in CMSO$_2$ is straightforward: we distinguish vertices of H and M as described above, and then apply φ relativized to those subsets of vertices.

Let $\varphi' := \breve{\varphi}$ and $k' = k$ if $k \geqslant 2$, and $\varphi' := \breve{\varphi} \wedge \varphi_{\mathrm{tw}<k}$ and $k' = 2$ if $k < 2$ (where the sentence $\varphi_{\mathrm{tw}<k}$ comes from Lemma 10). We apply Theorem 38 to k', t, and φ', and, given a graph G with weight function $\mathfrak{w}$, apply the asserted algorithm to $G' := \breve{G}^{V(G)}$ (with weight function $\mathfrak{w}'$), obtaining a set S'. By the construction of G' and φ', $G'[S']$ must be a forked version of $(G[S], M)$ for some $M \subseteq S \subseteq V(G)$ such that $G[S] \models \varphi(M)$ and $G[S]$ has treewidth less than k. Moreover, $\mathfrak{w}'(S') = \mathfrak{w}(M)$.

We return (S, M). To see the correctness of this output, note that for every $M \subseteq S \subseteq V(G)$ such that $G[S]$ is of treewidth less than k and $G[S] \models \varphi(M)$, if we denote $H = G[S]$, then $\breve{H}^M$ is an induced subgraph of G' of treewidth less than k', $\mathfrak{w}'(V(\breve{H}^M)) = \mathfrak{w}(M)$, and $\breve{H}^M \models \varphi'$. □

For an example application of Theorem 38, consider the MAXIMUM INDUCED CYCLE PACKING problem: given an (unweighted) graph G, find the largest (in terms of cardinality) collection of pairwise non-adjacent induced cycles in G. Consider the following property of a graph G and a vertex subset $M \subseteq V(G)$: G is a disjoint union of cycles and every connected component of G contains exactly one vertex of M. It is straightforward to write a CMSO$_2$ formula $\varphi(M)$ such that $G \models \varphi(M)$ if and only if G and M have this property. Noting that disjoint unions of cycles are 2-degenerate, we can apply Theorem 38 for the formula $\varphi(X)$ to conclude that the MAXIMUM INDUCED CYCLE PACKING problem admits a $n^{\mathcal{O}(\log^3 n)}$-time algorithm on $C_{>t}$-free graphs, for every fixed t. Here, we endow the input graph with a weight function that assigns a unit weight to every vertex.

8 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:

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

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 39. *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

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

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

PROOF OF CLAIM. It is clear that $\text{set}(i+1) \geqslant \text{set}(i)$, so suppose $\text{set}(i+1) \geqslant \text{set}(i) + 2$. Since $v_i v_{i+1}$ is an edge of G, we conclude that there is an edge in G° between the sets $\{X_j \mid j \leqslant \text{set}(i)\}$ and $\{X_j \mid j \geqslant \text{set}(i) + 2\}$, 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 $\text{set}(1) = 1$ and $\text{set}(p) = t$, 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 $t \geqslant 3$.

45

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. $\square$

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

- $\sum_{C: \text{ component of } G[X]} \mathfrak{w}(C)$ is maximized.

We observe the following.

Theorem 40. *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 $N^{\mathcal{O}(\log^2 N)}$.*

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

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

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 39 and the fact that MWIS can be solved in time $n^{\mathcal{O}(\log^2 n)}$ in n-vertex P_t-free graphs [14, 25], in time $n^{\mathcal{O}(\log^3 n)}$ in n-vertex $C_{>t}$-free graphs, using Theorem 1 only for MWIS, and in polynomial time in P_6-free [17] or $C_{>4}$-free graphs [1]. $\square$

As an example of an application of Theorem 40, we obtain the following corollary.

Corollary 41. *For every fixed d and t, given an n-vertex P_t-free graph G, in time $n^{\mathcal{O}(\log^2 n)}$ 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 40 to solve the instance $(G, \mathcal{F}, \mathfrak{w})$ of MAXIMUM INDUCED PACKING in time $N^{\mathcal{O}(\log^2 N)} = n^{\mathcal{O}(\log^2 n)}$. $\qquad\square$

Note that the strategy we used to prove Theorem 40 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 $\mathcal{C}$ is called *weakly hyperfinite* if for every $\varepsilon > 0$ there is $c(\varepsilon) \in \mathbb{N}$, such that in every graph $F \in \mathcal{C}$ 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 [23, Section 16.2]. Weakly hyperfinite classes are also known under the name *fragmentable* [13]. Every class closed under edge and vertex deletion which has sublinear separators is weakly hyperfinite [23, 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 $\mathcal{C}$ of graphs, by LARGEST INDUCED $\mathcal{C}$-GRAPH we denote the following problem: given a graph G, find a largest induced subgraph of G, which belongs to $\mathcal{C}$. To make the problem well defined, we will always assume that $K_1 \in \mathcal{C}$. We can now conclude the following.

Theorem 42. *Let $\mathcal{C}$ be a nonempty, weakly hyperfinite class of graphs, which is closed under vertex deletion and disjoint union operations. Then, the LARGEST INDUCED $\mathcal{C}$-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 $\mathcal{C}$, 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 $\mathcal{C}$. 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 40 to solve the instance $(G, \mathcal{F}, \mathfrak{w})$ of MAXIMUM INDUCED PACKING in time $n^{\mathcal{O}(\log^3 n^c)} = n^{\mathcal{O}(\log^3 n)}$ 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 $\mathcal{C}$ is closed under the disjoint union operation, we observe that $G[X]$ is a feasible solution to LARGEST INDUCED $\mathcal{C}$-GRAPH. Moreover we have $|X| \geqslant |X'| \geqslant (1 - \varepsilon)|X^*|$. $\square$

References

[1] T. Abrishami, M. Chudnovsky, M. Pilipczuk, P. Rzążewski, and P. D. Seymour. Induced subgraphs of bounded treewidth and the container method. *CoRR*, abs/2003.05185, 2020. Accepted to SODA 2021.

[2] V. 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).

[3] V. E. Alekseev. Polynomial algorithm for finding the largest independent sets in graphs without forks. *Discrete Applied Mathematics*, 135(1–3):3–16, 2004.

[4] A. Atminas, V. V. Lozin, and I. Razgon. 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*, volume 7357 of *Lecture Notes in Computer Science*, pages 142–152. Springer, 2012.

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

[6] K. Cameron and P. Hell. Independent packings in structured graphs. *Math. Program.*, 105(2-3):201–213, 2006.

[7] M. Chudnovsky, M. Pilipczuk, M. Pilipczuk, and S. Thomassé. Quasi-polynomial time approximation schemes for the Maximum Weight Independent Set problem in H-free graphs. *CoRR*, abs/1907.04585, 2019.

[8] M. Chudnovsky, M. Pilipczuk, M. Pilipczuk, and S. Thomassé. 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*, pages 2260–2278. SIAM, 2020.

[9] B. Courcelle. The Monadic Second-Order Logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[10] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic — A Language-Theoretic Approach*, volume 138 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2012.

[11] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[12] G. Ding. Subgraphs and well-quasi-ordering. *J. Graph Theory*, 16(5):489–502, 1992.

[13] K. Edwards and G. Farr. Fragmentability of graphs. *Journal of Combinatorial Theory, Series B*, 82(1):30 – 37, 2001.

[14] P. Gartland and D. Lokshtanov. Independent Set on P_k-free graphs in quasi-polynomial time. *CoRR*, abs/2005.00690, 2020. Accepted to FOCS 2020.

[15] M. Grohe and S. Kreutzer. Methods for algorithmic meta theorems. In *Model Theoretic Methods in Finite Combinatorics - AMS-ASL Joint Special Session, Washington, DC, USA, January 5-8, 2009*, volume 558 of *Contemporary Mathematics*, pages 181–206. American Mathematical Society, 2009.

[16] M. Grohe and D. Marx. On tree width, bramble size, and expansion. *J. Comb. Theory, Ser. B*, 99(1):218–228, 2009.

[17] A. Grzesik, T. Klimošová, M. Pilipczuk, and M. Pilipczuk. 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*, pages 1257–1271. SIAM, 2019.

[18] A. Gyárfás. Problems from the world surrounding perfect graphs. *Applicationes Mathematicae*, 19:413–441, 1987.

[19] M. Hatzel, P. Komosa, M. Pilipczuk, and M. Sorge. Constant congestion brambles. *CoRR*, abs/2008.02133, 2020.

[20] D. Lokshtanov, M. Vatshelle, and Y. Villanger. 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*, pages 570–581. SIAM, 2014.

[21] V. V. Lozin and M. 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.

[22] G. J. Minty. On maximal independent sets of vertices in claw-free graphs. *J. Comb. Theory, Ser. B*, 28(3):284–304, 1980.

[23] J. Nešetřil and P. Ossona de Mendez. *Sparsity — Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[24] J. Novotná, K. Okrasa, M. Pilipczuk, P. Rzążewski, E. J. van Leeuwen, and B. Walczak. Subexponential-time algorithms for finding large induced sparse subgraphs. *Algorithmica*, July 2020.

[25] M. Pilipczuk, M. Pilipczuk, and P. Rzążewski. Quasi-polynomial-time algorithm for Independent Set in P_t-free graphs via shrinking the space of induced paths. In *SOSA 2021 (to appear)*, 2021. Available at https://arxiv.org/abs/2009.13494.

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

[27] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004.

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