# Homomorphisms Are a Good Basis for Counting Small Subgraphs*

Radu Curticapean[a], Holger Dell[b], and Dániel Marx[a]

[a]Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
[b]Saarland University and Cluster of Excellence (MMCI), Saarbrücken, Germany

May 5, 2017

**Abstract**

We introduce *graph motif parameters*, a class of graph parameters that depend only on the frequencies of constant-size induced subgraphs. Classical works by Lovász show that many interesting quantities have this form, including, for fixed graphs $H$, the number of $H$-copies (induced or not) in an input graph $G$, and the number of homomorphisms from $H$ to $G$.

Using the framework of graph motif parameters, we obtain faster algorithms for counting subgraph copies of fixed graphs $H$ in host graphs $G$: For graphs $H$ on $k$ edges, we show how to count subgraph copies of $H$ in time $k^{O(k)} \cdot n^{0.174k+o(k)}$ by a surprisingly simple algorithm. This improves upon previously known running times, such as $O(n^{0.91k+c})$ time for $k$-edge matchings or $O(n^{0.46k+c})$ time for $k$-cycles.

Furthermore, we prove a general complexity dichotomy for evaluating graph motif parameters: Given a class $\mathcal{C}$ of such parameters, we consider the problem of evaluating $f \in \mathcal{C}$ on input graphs $G$, parameterized by the number of induced subgraphs that $f$ depends upon. For every recursively enumerable class $\mathcal{C}$, we prove the above problem to be either FPT or #W[1]-hard, with an explicit dichotomy criterion. This allows us to recover known dichotomies for counting subgraphs, induced subgraphs, and homomorphisms in a uniform and simplified way, together with improved lower bounds.

Finally, we extend graph motif parameters to colored subgraphs and prove a complexity trichotomy: For vertex-colored graphs $H$ and $G$, where $H$ is from a fixed class $\mathcal{H}$, we want to count color-preserving $H$-copies in $G$. We show that this problem is either polynomial-time solvable or FPT or #W[1]-hard, and that the FPT cases indeed need FPT time under reasonable assumptions.

## 1 Introduction

Deciding the existence of subgraph patterns $H$ in input graphs $G$ constitutes the classical *subgraph isomorphism problem* [14, 53], which generalizes NP-complete problems like the Hamiltonian cycle problem or the clique problem. In some applications however, it is not sufficient to merely know whether $H$ occurs in $G$, but instead one wishes to determine the *number* of such occurrences. This is clearly at least as hard as deciding their existence, but it can be much harder: The existence of perfect matchings can be tested in polynomial time, but the counting version is #P-hard [54].

Subgraph counting problems have applications in areas like statistical physics, probabilistic inference, and network analysis. In particular, in network analysis, such problems arise in the context of discovering *network motifs*. These are small patterns that occur more often in a network than would be expected if the network was random. Through network motifs, the problem of counting subgraphs has found applications in the study of gene transcription networks, neural networks, and social networks [46], and there is a large body of work dedicated to the algorithmic discovery of network motifs [27, 2, 49, 36, 51, 11, 37, 55, 50].

Inspired by these applications, we study the algorithmic problem of counting occurrences of *small* patterns $H$ in large host graphs $G$. The abstract notion of a "pattern occurrence" may be formalized in

*An extended abstract of this paper appears at STOC 2017. Part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing and the Dagstuhl Seminar 17041 – "Randomization in Parameterized Complexity".

various ways, which may result in vastly different problems: To state only some examples, we may be interested in counting subgraph copies of a graph $H$, or induced subgraph copies of $H$, or homomorphisms from $H$ to $G$, and we can also consider settings where both pattern $H$ and host graph $G$ are colored and we wish to count subgraphs of $G$ that are color-preserving isomorphic to $H$.

It may seem daunting at first to try to deal with all different types of pattern occurrences. Fortunately, Lovász [42, 43] defined a framework that allows us to express virtually all kinds of pattern types in a unified way. As it turns out, graph parameters such as the number of subgraph copies of $H$ (induced or not) in a host graph $G$, or the number of graph homomorphisms from $H$ to $G$ are actually just "linear combinations" of each other in a well-defined sense. We build on this and define a general framework of so-called *graph motif parameters* to capture counting linear combinations of small patterns, into which (induced) subgraph or homomorphism numbers embed naturally as special cases.

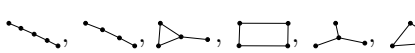
In the remainder of the introduction, we first discuss algorithmic and complexity-theoretic aspects of counting (induced) subgraphs and homomorphisms in §1.1–§1.3 and state the results we derive for these special cases. In §1.4, we then give an introduction into the general framework of graph motif parameters, our interpretation of Lovász's unified framework, which also provides the main techniques for our proofs. Finally, in §1.5 we give an exposition of our results for vertex-colored subgraphs.

## 1.1 Counting small subgraphs

For any fixed $k$-vertex pattern graph $H$, we can count all subgraph copies of $H$ in an $n$-vertex host graph $G$ using brute-force for a running time of $O(n^k)$. While this running time is polynomial for any fixed $H$, it quickly becomes infeasible as $k$ grows. Fortunately enough, non-trivial improvements on the exponent are known, albeit only for specific classes of patterns:

- We can count triangles in the same time $O(n^\omega)$ that it takes to multiply two $(n \times n)$-matrices [31]. It is known that $\omega < 2.373$ holds [56, 26]. This approach can be generalized from triangles to $k$-cliques with $k \in \mathbb{N}$ [47], for a running time of $n^{\omega k/3+O(1)}$. Fast matrix multiplication is also used to improve on exhaustive search for counting cycles of length at most seven [3] and various other problems [38, 23].
- For $k$-edge paths or generally any pattern of bounded pathwidth, a "meet in the middle" approach yields $n^{k/2+O(1)}$ time algorithms [39, 4]. For a while, this approach appeared to be a barrier for faster algorithms, until Björklund et al. [5] gave an algorithm for counting $k$-paths, matchings on $k$ vertices, and other $k$-vertex patterns of bounded pathwidth in time $n^{0.455k+O(1)}$.
- If $\mathrm{vc}(H)$ is the *vertex-cover number of* $H$, that is, the size of its smallest vertex-cover, then we can count $H$-copies in time $n^{\mathrm{vc}(H)+O(1)}$ [58] (also cf. [40, 17]). Essentially, one can exhaustively iterate over the image of the minimum vertex-cover in $G$, which gives rise to $n^{\mathrm{vc}(G)}$ choices; the rest of $H$ can then be embedded by dynamic programming. Note that $\mathrm{vc}(H)$ may be constant even for large graphs $H$, e.g., if $H$ is a star.

In this paper, we unify some of the algorithms above and generalize them to arbitrary subgraph patterns; in many cases our algorithms are faster. For two graphs $H$ and $G$, let $\#\mathrm{Sub}(H \to G)$ be the number of subgraphs of $G$ that are isomorphic to $H$. Our main algorithmic result states that $\#\mathrm{Sub}(H \to G)$ can be determined in time $O(n^{t+1})$, where $t$ is the maximum treewidth (a very popular measure of tree-likeness) among the homomorphic images of $H$. For our purposes, a homomorphic image of $H$ is any simple graph that can be obtained from $H$ by possibly merging non-adjacent vertices. For instance, identifying the first and the last vertex in the 4-path  yields the 4-cycle  , and further identifying two non-adjacent vertices in the 4-cycle yields the 2-path  . We define the *spasm of* $H$ as the set of all homomorphic images of $H$, that is, as the set of "all possible non-edge contractions" of $H$. As an example, for the 4-path, we have

$$\mathrm{Spasm}\left(\text{  }\right) = \left\{\text{  }\right\}. \quad (1)$$

Our main algorithmic result can then be stated as follows:

**Theorem 1.1.** *Given as input a $k$-edge graph $H$ and an $n$-vertex graph $G$, we can compute the number $\#\mathrm{Sub}(H \to G)$ in time $k^{O(k)} \cdot n^{t+1}$, where $t$ is the maximum treewidth in the spasm of $H$.*

As an example, for the 4-path, the largest treewidth among the graphs in the spasm is 2, and so Theorem 1.1 yields a running time of $O(n^3)$ for counting 4-paths. In fact, even the 6-path has only graphs with treewidth at most 2 in its spasm, so the same cubic running time applies.

Theorem 1.1 generalizes the vertex-cover based algorithm [58] mentioned before: Merging vertices can never increase the size of the smallest vertex-cover, and so the maximum treewidth in the spasm of $H$ is bounded by $\mathrm{vc}(H)$, since the vertex-cover number is an upper bound for the treewidth of a graph. Note that, while contracting edges cannot increase the treewidth of a graph, it is apparent from (1) that contracting non-edges might. In fact, if $H$ is the $k$-edge matching, *all* $k$-edge graphs can be obtained by contracting non-edges, including expander graphs with treewidth $\Omega(k)$. However, Scott and Sorkin [52, Corollary 21] proved that every graph with at most $k$ edges has treewidth at most $0.174 \cdot k + o(k)$. This bound enables the following immediate corollary to Theorem 1.1.

Spasm:
VC >= tw

contr edge
tw not inc

contr n-edg
tw can inc

**Corollary 1.2.** *Given as input a $k$-edge graph $H$ and an $n$-vertex graph $G$, we can compute the number $\#\mathrm{Sub}(H \to G)$ in time $k^{O(k)} \cdot n^{0.174 \cdot k + o(k)}$.*

Our exponent is obviously smaller than the previously known $0.455 \cdot |V(H)| = 0.91 \cdot k$ for the $k$-edge matching and $0.455 \cdot k$ for the $k$-path, but somewhat surprisingly, our algorithm is also significantly simpler.

When $H$ is a triangle, the algorithm from Theorem 1.1 matches the running time $O(n^3)$ of the exhaustive search method. To achieve a smaller exponent, we have to use matrix multiplication, since faster triangle detection is equivalent to faster Boolean matrix multiplication [57]. Indeed, we are able to generalize the $O(n^\omega)$ time algorithm for counting triangles to arbitrary graphs $H$ whose spasm has treewidth at most 2.

**Theorem 1.3.** *If all graphs in the spasm of $H$ have treewidth at most two, we can compute $\#\mathrm{Sub}(H \to G)$ in time $f(H) \cdot |V(G)|^\omega$, where $f(H)$ is a function that only depends on $H$.*

#Sub()
tw <= 2

This algorithm applies, for example, to paths of length at most 6, thus providing an alternative and simplified way to obtain the corresponding results of [3].

We now turn to hardness results for counting subgraphs. Here, the vertex-cover number of the pattern $H$ plays a special role: When it is bounded by a fixed constant $b \in \mathbb{N}$, we have an $n^{b+O(1)}$ time algorithm even when the size of the pattern is otherwise unbounded. However when it is unbounded, e.g., for $k$-paths, the best known running times are of the form $n^{\epsilon k}$ for some $\epsilon \in (0, 1)$. Given these modest improvements for counting subgraph patterns of unbounded vertex-cover number, it is tempting to conjecture that "the exponent cannot remain constant" for such patterns. A result by a subset of the authors [17] shows that this conjecture is indeed true–for an appropriate formalization of the respective computational problem and under appropriate complexity-theoretic assumptions.

The *counting exponential time hypothesis* (#ETH) by Impagliazzo and Paturi [30], adapted to the counting setting [20], states that there is no $\exp(o(n)) \cdot \mathrm{poly}(m)$-time algorithm to count all satisfying assignments of a given 3-CNF formula with $n$ variables and $m$ clauses. More convenient for us is the following consequence of #ETH: There is no $f(k) \cdot n^{o(k)}$ time algorithm to count all cliques of size exactly $k$ in a given $n$-vertex graph [10] — thus, the $k$-clique problem is a hard special case of the subgraph counting problem and clearly $k$-cliques have large vertex-cover number. Of course, this worst-case hardness of the most general subgraph counting problem does not directly help us understand the complexity of particular cases, such as counting $k$-matchings or $k$-paths.

We can model our interest in special cases by restricting the pattern graphs $H$ to be from a fixed class $\mathcal{H}$ of graphs: The computational problem $\#\mathsf{Sub}(\mathcal{H})$ is to compute the number $\#\mathrm{Sub}(H \to G)$ of $H$-copies in $G$ when given two graphs $H \in \mathcal{H}$ and $G$ as input. To prove that #ETH implies that no fixed-parameter tractable algorithm can exist for $\#\mathsf{Sub}(\mathcal{H})$, one ultimately establishes a *parameterized reduction* from the $k$-clique problem to the $H$-subgraph counting problem, which has the important property that $\mathrm{vc}(H)$ is bounded by $g(k)$, a function only depending on $k$.

ETH
~
FPT

The parameterized reduction in [17] was very complex with various special cases and a Ramsey argument that made $g$ a very large function. While it was sufficient to conditionally rule out $f(k) \cdot n^c$ time algorithms for $\#\mathsf{Sub}(\mathcal{H})$ for any constant $c$ and graph class $\mathcal{H}$ of unbounded vertex-cover number, it left open the possibility of, for example, $n^{\sqrt{\mathrm{vc}(H)}}$ time algorithms. Running times of the form $n^{o(k/\log k)}$ could be ruled out under #ETH only for certain special cases, such as counting $k$-matchings or $k$-paths. In this paper, we obtain the stronger hardness result for all hard families $\mathcal{H}$. For technical reasons, we assume that $\mathcal{H}$

can be recursively enumerated; without this assumption, we would however obtain a similar result under a non-uniform version of #ETH.

**Theorem 1.4.** *Let $\mathcal{H}$ be a recursively enumerable class of graphs of unbounded vertex-cover number. If #ETH holds, then $\#\mathsf{Sub}(\mathcal{H})$ cannot be computed in time $f(H) \cdot n^{o(\mathrm{vc}(H)/\log \mathrm{vc}(H))}$.*

The log-factor here is related to an open problem in parameterized complexity, namely whether you can "beat treewidth" [44], i.e., whether there is an algorithm to find $H$ as a subgraph in time $f(H) \cdot n^{o(\mathrm{tw}(H))}$, or whether such an algorithm is ruled out by ETH for every graph class $\mathcal{H}$ of unbounded treewidth. Indeed, replacing $\mathrm{vc}(H)$ with $\mathrm{tw}(H)$ in Theorem 1.4 essentially yields the hardness result in [44, Corollary 6.3], and since $\mathrm{tw}(H) \leq \mathrm{vc}(H)$, our theorem can be seen as a strengthening of this result in the counting world. In fact, our hardness proof is based on this weaker version.

Instead of relying on #ETH, we can also consider $\#\mathsf{Sub}(\mathcal{H})$ from the viewpoint of fixed-parameter tractability. In this framework, the problem is parameterized by $|V(H)|$. A problem is #W[1]-complete if it is equivalent under parameterized reductions to the problem of counting $k$-cliques, where the allowed reductions are Turing reductions that run in time $f(k) \cdot \mathrm{poly}(n)$. As mentioned before, it is known that #ETH implies $\mathsf{FPT} \neq \#\mathsf{W}[1]$. In this setting, Theorem 1.4 takes on the following form:

**Theorem 1.5 ([17]).** *Let $\mathcal{H}$ be a recursively enumerable class of graphs. If $\mathcal{H}$ has bounded vertex-cover number, then $\#\mathsf{Sub}(\mathcal{H})$ is polynomial-time computable. Otherwise, it is #W[1]-complete when parameterized by the pattern size $|V(H)|$.*

The original proof of Theorem 1.5 relied on the #W[1]-completeness of counting $k$-matchings, which was nontrivial on its own [15, 6]. Then an extensive graph-theoretic analysis was used to find "$k$-matching gadgets" in any graph class $\mathcal{H}$ of unbounded vertex-cover number. These gadgets enable a parameterized reduction from counting $k$-matchings to $\#\mathsf{Sub}(\mathcal{H})$, but they are also responsible for the uncontrollable blowup in the parameter that lead to highly non-tight results under #ETH. We obtain a much simpler proof of Theorem 1.5 that does not assign a special role to $k$-matchings.

Interestingly, Theorem 1.5 implies that no problem of the form $\#\mathsf{Sub}(\mathcal{H})$ is "truly" FPT. That is, every such problem that is not #W[1]-complete is in fact already polynomial-time solvable. Thus, if we assume the widely-believed claim that $\mathsf{FPT} \neq \#\mathsf{W}[1]$ holds, then Theorem 1.5 exhaustively classifies the polynomial-time solvable problems $\#\mathsf{Sub}(\mathcal{H})$. Indeed, such a sweeping dichotomy would not have been possible by merely assuming that $\mathsf{P} \neq \mathsf{P}^{\#\mathsf{P}}$, since there exist artificial classes $\mathcal{H}$ with #P-intermediate [13] $\#\mathsf{Sub}(\mathcal{H})$.

We remark that a decision version of Theorem 1.5 is an open problem. It is known only for graph classes that are hereditary, that is, closed under induced subgraphs [32], where the dichotomy criterion is different. Moreover, certain non-hereditary cases, such as the W[1]-completeness of deciding the existence of a bipartite clique or a grid, have been resolved only recently [41, 12].

## 1.2 Counting small homomorphisms

Similar classifications as Theorem 1.5 for counting subgraphs were previously known for counting homomorphisms [19] from a given class $\mathcal{H}$. Recall that a homomorphism from a pattern $H$ to a host $G$ is a mapping $f : V(H) \to V(G)$ such that $uv \in E(H)$ implies $f(u)f(v) \in E(G)$; we write $\#\mathrm{Hom}(H \to G)$ for the number of such homomorphisms. In the context of pattern counting problems, we can interpret homomorphisms as a relaxation of the subgraph notion: If a homomorphism $f$ from $H$ to $G$ is injective, then it constitutes a subgraph embedding from $H$ to $G$. However, since we generally do *not* require injectivity in homomorphisms, these may well map into other *homomorphic images*. (Recall that the spasm of $H$ contains exactly the loop-free homomorphic images of $H$.)

From an algorithmic viewpoint, not requiring injectivity makes counting patterns easier: One can now use separators to divide $H$ into subpatterns and compose mappings of different subpatterns to a global one for $H$ via dynamic programming. In this process, only the locations of separators under the subpattern mapping need to be memorized, while non-separator vertices of subpatterns may be forgotten. As an example, note that counting $k$-paths is #W[1]-complete by Theorem 1.5, but counting homomorphisms from a $k$-path to a host graph $G$ is polynomial-time solvable. Indeed, the latter problem amounts to counting $k$-walks in $G$, which can be achieved easily by taking the $k$-th power of the adjacency matrix of $G$, a process that can be

#ETH unb VC => #Sub > ...

tw <= vc

Q: why not sap when inj?

A: inj => no subpattern?

interpreted as a dynamic programming algorithm: Given a table with the number of $\ell$-walks from $s$ to $u$ for each $u$, we can compute a table with the number of $(\ell+1)$-walks from $s$ to $v$ for all $v$. That is, we only need to store the last vertex seen in a walk. This idea can be generalized easily to graphs of bounded treewidth using a straightforward dynamic programming approach on the tree decomposition of $H$.

**Proposition 1.6 (Díaz et al. [21]).** *There is a deterministic* $\exp(O(k))+\mathrm{poly}(k)\cdot n^{\mathrm{tw}(H)+1}$ *time algorithm to compute the number of homomorphisms from a given graph* $H$ *to a given graph* $G$*, where* $k=|V(H)|$*,* $n=|V(G)|$*, and* $\mathrm{tw}(H)$ *denotes the treewidth of* $H$*.*

Algo n^tw+1

This proposition is the basis of our main algorithmic result for subgraphs (Theorem 1.1) and other graph motif parameters, so we include a formal proof in the appendix. For graphs $H$ of treewidth at most two, we can speed up the dynamic programming algorithm by using fast matrix multiplication:

**Theorem 1.7.** *If* $H$ *has treewidth at most* 2*, we can compute* $\#\mathrm{Hom}(H\to G)$ *in time* $\mathrm{poly}(|V(H)|)\cdot|V(G)|^{\omega}$*.*

Apart from being more well-behaved than subgraphs in terms of algorithmic tractability, homomorphisms also allow for simpler hardness proofs: Several constructions are significantly easier to analyze for homomorphisms than for subgraphs, as we will see in our proofs. This might explain why a dichotomy for counting homomorphisms from a fixed class $\mathcal{H}$ was obtained an entire decade before its counterpart for subgraphs; it establishes the treewidth of $\mathcal{H}$ as the tractability criterion for the problems $\#\mathsf{Hom}(\mathcal{H})$.

**Theorem 1.8 (Dalmau and Jonsson [19]).** *Let* $\mathcal{H}$ *be a recursively enumerable class of graphs. If* $\mathcal{H}$ *has bounded treewidth, then the problem* $\#\mathsf{Hom}(\mathcal{H})$ *of counting homomorphisms from graphs in* $\mathcal{H}$ *into host graphs is polynomial-time solvable. Otherwise, it is* $\#\mathsf{W}[1]$*-complete when parameterized by* $|V(H)|$*.*

In the remainder of the paper, we will also require the following lower bound under #ETH, the proof of which is a simple corollary of [44, Corollary 6.2 and 6.3].

**Proposition 1.9.** *Let* $\mathcal{H}$ *be a recursively enumerable class of graphs of unbounded treewidth. If* #ETH *holds, then there is no* $f(H)\cdot|V(G)|^{o(\mathrm{tw}(H)/\log \mathrm{tw}(H))}$ *time algorithm to compute* $\#\mathsf{Hom}(\mathcal{H})$ *for graphs* $H\in\mathcal{H}$ *and* $G$*.*

Finally, we remark that the decision version of Theorem 1.8, that is, the dichotomy theorem for deciding the existence of a homomorphism from $H\in\mathcal{H}$ to $G$ is known and has a different criterion [29]: The decision problem is polynomial-time computable even when only the homomorphic cores of all graphs in $\mathcal{H}$ have bounded treewidth, and it is W[1]-hard otherwise.

## 1.3 Counting small induced subgraphs

Let us also address the problem of counting small induced subgraphs from a class $\mathcal{H}$. This is a natural and well-studied variant of counting subgraph copies [38, 13, 33, 34, 35, 45], and for several applications it represents a more appropriate notion of "pattern occurrence". From the perspective of dichotomy results however, it is less intricate than subgraphs or homomorphisms: Counting induced subgraphs is known to be #W[1]-hard for any infinite pattern class $\mathcal{H}$, and even the corresponding decision version is W[1]-hard.

**Theorem 1.10 ([13]).** *Let* $\mathcal{H}$ *be a recursively enumerable class of graphs. If* $\mathcal{H}$ *is finite, then the problem* $\#\mathsf{Ind}(\mathcal{H})$ *of counting induced subgraphs from* $\mathcal{H}$ *is polynomial-time solvable. Otherwise, it is* $\#\mathsf{W}[1]$*-complete when parameterized by* $|V(H)|$*.*

#Ind(H) finite => poly-time sonst => #W[1]

Jerrum and Meeks [33, 34, 35, 45] introduced the following generalization of the problems $\#\mathsf{Ind}(\mathcal{H})$ to fixed graph properties $\Phi$: Given a graph $G$ and $k\in\mathbb{N}$, the task is to compute the number of induced $k$-vertex subgraphs that have property $\Phi$. Let us call this problem $\#\mathsf{IndProp}(\Phi)$. They identified some classes of properties $\Phi$ that render this problem #W[1]-hard. Using our machinery, we get a full dichotomy theorem for this class of problems.

**Theorem 1.11 (simple version).** *If* $\Phi$ *is a decidable graph property, then* $\#\mathsf{IndProp}(\Phi)$ *is fixed-parameter tractable or* $\#\mathsf{W}[1]$*-hard when parameterized by* $|V(H)|$*.*

## 1.4 A unified view: Graph motif parameters

We now discuss our proof techniques on a high level. From a conceptual perspective, our most important contribution lies in finding a framework for understanding the parameterized complexity of subgraphs, induced subgraphs, and homomorphisms in a uniform context. Note that we quite literally *find* this framework: That is, we do not develop it ourselves, but we rather adapt works by Lovász et al. dating back to the 1960s [42, 8]. The most important observation is the following:

*Many counting problems are actually linear combinations of homomorphisms in disguise!*

That is, there are elementary transformations to express, say, linear combinations of subgraphs as linear combinations of homomorphisms, and vice versa.

The algorithms for subgraphs (Theorem 1.1 and Corollary 1.2) are based on a reduction from subgraph counting to homomorphism counting, so we want to find relations between the number of subgraphs and the number of homomorphisms. To get things started, note that injective homomorphisms from $H$ to $G$, also called *embeddings*, correspond to a subgraph $F$ of $G$ that is isomorphic to $H$, and in fact, the number $\#\mathrm{Emb}(H \to G)$ of embeddings is equal to the number $\#\mathrm{Sub}(H \to G)$ of subgraphs times $\#\mathrm{Aut}(H)$, the number of automorphisms of $H$.

injective H = #Emb = #Sub*#Aut

Homomorphisms cannot map two adjacent vertices of $H$ to the same vertex of $G$, assuming that $G$ does not have any loops. For instance, every homomorphism from  to $G$ must be injective, and therefore the number of triangles in $G$ is equal to the number of such homomorphisms, up to a factor of 6: the number of automorphisms of the triangle. Formally, we have $\#\mathrm{Sub}($  $\to G) = \frac{1}{6} \cdot \#\mathrm{Hom}($  $\to G)$ for every graph $G$ that does not have loops.

More interesting cases occur when homomorphisms from $H$ to $G$ are not automatically injective. Clearly, the set of all homomorphisms contains the injective ones, which suggests we should simply count all homomorphisms and then subtract the ones that are not injective. Any non-injective homomorphism $h$ from $H$ to $G$ has the property that there are at least two (non-adjacent) vertices that it maps to the same vertex; in other words, its image $h(H)$ is isomorphic to some member of $\mathrm{Spasm}(H)$ other than $H$ itself. For example, $\#\mathrm{Hom}($  $\to G) - \#\mathrm{Hom}($  $\to G)$ is the number of injective homomorphisms from  to $G$ since the only way for such a homomorphism to be non-injective is that it merges the two degree-1 vertices. In general, the number $\#\mathrm{Emb}(H \to G)$ of injective homomorphisms is

$$\#\mathrm{Hom}(H \to G) - \sum_{F \in \mathrm{Spasm}(H) \setminus \{H\}} \#\mathrm{Emb}(F \to G) .$$

Since each such $F$ is strictly smaller than $H$, this fact yields a recursive procedure to compute $\#\mathrm{Emb}(H \to G)$. However, there is a better way: We can use Möbius inversion over the partition lattice to obtain a closed formula. We already mentioned that the spasm of $H$ can be obtained by consolidating non-adjacent vertices of $H$ in all possible ways. This means that we consider partitions $\rho$ of $V(H)$ in which each block is an independent set, and then form the *quotient graph* $H/\rho$ obtained from $H$ by merging each block of $\rho$ into a single vertex. To express the injective homomorphisms from $H$ to $G$ (and hence the number of $H$-subgraphs) as a linear combination of homomorphisms, we consider all possible types in which a homomorphism $h$ from $H$ to $G$ can fail to be injective. More precisely, we define this type $\rho_h$ of $h$ to be the partition of $V(H)$, where each block is the set of vertices of $H$ that map to the same vertex of $G$ under $h$. The homomorphism $h$ is injective if and only if $\rho_h$ is the finest partition, i.e., the partition where each block has size one.

The homomorphisms from $H/\rho$ to $G$ are precisely those homomorphisms from $H$ to $G$ that fail to be injective "at least as badly as $\rho$", that is, those homomorphisms $f$ whose type $\rho_f$ is a coarsening of $\rho$. As remarked by Lovász et al. [42, 8], one can then use Möbius inversion, a generalization of the inclusion–exclusion principle, to turn this observation into the "inverse" identity

$$\#\mathrm{Sub}(H \to G) = \sum_{\rho} \frac{(-1)^{|V(H)|-|V(H/\rho)|} \cdot \prod_{B \in \rho} (|B|-1)!}{\#\mathrm{Aut}(H)} \cdot \#\mathrm{Hom}(H/\rho \to G) . \tag{2}$$

The sum in (2) ranges over all partitions $\rho$ of $V(H)$. Hence, the number of $H$-subgraphs in $G$ is equal to a linear combination of the numbers of homomorphisms from graphs $H/\rho$ to $G$. Each $H/\rho$ is isomorphic to a graph in $\mathrm{Spasm}(H)$, and so by collecting terms for isomorphic graphs, (2) represents the number of

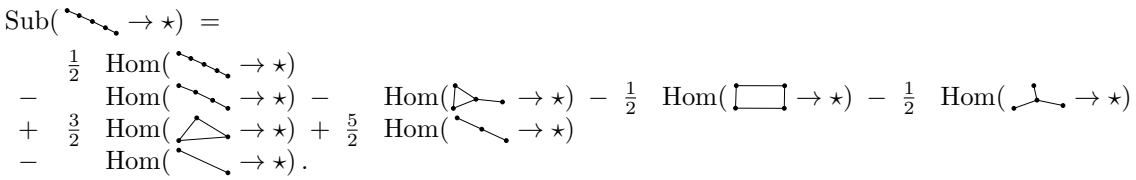


Figure 1: An example for (2), where $H$ is the path with four edges. The number of subgraphs is represented as a linear combination of homomorphisms from graphs $F \in \mathrm{Spasm}(H)$. Each such $F$ has treewidth at most two, so we can compute the homomorphism numbers in time $O(n^3)$ via Proposition 1.6. Computing the linear combination on the right side yields an $O(n^3)$-time algorithm to count 4-paths, and in fact this is the algorithm in Theorem 1.1.

$H$-subgraphs as a linear combination of homomorphism numbers from graphs $F$ in the spasm of $H$; see Figure 1 for an example.

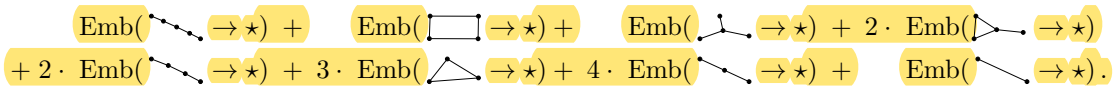
The identity (2) can be viewed as a basis transformation in a certain vector space of graph parameters, and we formalize this perspective in §3. A similar identity turns out to hold for counting induced subgraphs as well, so all three graph parameter types can we written as finite linear combinations of each other. This motivates the notion of a *graph motif parameter*, which is any graph parameter $f$ that is a finite linear combination of induced subgraph numbers. That is, there are coefficients $\alpha_1, \ldots, \alpha_t \in \mathbb{Q}$ and graphs $H_1, \ldots, H_t$ such that, for all graphs $G$, we have

$$f(G) = \sum_{i=1}^{t} \alpha_i \cdot \#\mathrm{IndSub}(H_i \to G). \tag{3}$$

We study the problem of computing graph motif parameters $f$. For our results in parameterized complexity, we parameterize this problem by the description length $k$ of $\alpha_1, \ldots, \alpha_t$ and $H_1, \ldots, H_t$. Due to the basis transformation between induced subgraphs, subgraphs, and homomorphisms, writing $\#\mathrm{Hom}(H_i \to G)$ instead of $\#\mathrm{IndSub}(H_i \to G)$ in (3) yields an equivalent class of problems — switching bases only leads to a factor $g(k)$ overhead in the running time for some computable function $g$, which we can neglect for our purposes.

Our main result is that the complexity of computing any graph parameter $f$ is exactly governed by the maximum complexity of counting the homomorphisms occurring in its representation over the homomorphism basis. More precisely, let $\alpha_1, \ldots, \alpha_t \in \mathbb{Q}$ and $H_1, \ldots, H_t$ be graphs with $f(G) = \sum_i \alpha_i \cdot \#\mathrm{Hom}(H_i \to G)$ for all graphs $G$. Our algorithmic results are based on the following observation: If each $\#\mathrm{Hom}(H_i \to G)$ can be computed in time $O(n^c)$ for $n = |V(G)|$ and some constant $c \geq 0$, then $f(G)$ can be computed in time $O(n^c)$ for the same constant $c$. However, we show that the reverse direction also holds: If $f$ can be computed in time $O(n^c)$ for some $c \geq 0$, then each $\#\mathrm{Hom}(H_i \to G)$ with $\alpha_i \neq 0$ can be computed in time $O(n^c)$ for the same constant $c$. The reduction that establishes this fine-grained equivalence gives rise to our results under #ETH and our new #W[1]-hardness proof. Note that such an equivalence is *not* true for linear combinations of embedding numbers, as can be seen from the following example.

**Example 1.12.** *Consider the following linear combination:*



*When this linear combination of embeddings is transformed into the homomorphism basis via* (2)*, most terms cancel, and it turns out that it is equal to* $\mathrm{Hom}($  $\to \star)$*, that is, it counts the number of walks of length* 4*. Counting* 4*-walks can be done in time* $O(n^2)$ *via Proposition 1.6, but counting, for example, triangles is not known to be possible faster than* $O(n^\omega)$*. More generally, counting walks of length* $k$ *is in* $O(n^2)$*-time, but counting* paths *of length* $k$ *is* #W[1]*-hard and not in time* $g(k) \cdot n^{o(k/\log k)}$ *under* #ETH*.*

Thus, even a linear combination of subgraph numbers that looks complex at first and contains as summands embedding numbers that are fairly hard to compute can actually be quite a bit easier due to cancellation effects that occur when rewriting it as linear combination of homomorphism numbers.

As in the case of subgraphs in §1.1, we consider classes $\mathcal{A}$ of linear combinations to get more expressive hardness results. That is, each element of $\mathcal{A}$ is a pattern-coefficient list $(\alpha_1, H_1), \ldots, (\alpha_t, H_t)$ as above. The evaluation problem $\#\mathsf{Ind}(\mathcal{A})$ for graph motif parameters from $\mathcal{A}$ is then given a pattern-coefficient list from $\mathcal{A}$ and a graph $G$, and is supposed to compute the linear combination (3). We have the following result for the complexity of computing graph motif parameters.

**Theorem 1.13 (intuitive version).** *Let $\mathcal{A}$ be a recursively enumerable class of pattern-coefficient lists. If the linear combinations (3), re-expressed as linear combinations of homomorphisms, contain non-zero coefficients only for graphs of treewidth at most $t$, then the problem $\#\mathsf{Ind}(\mathcal{A})$ can be computed in time $f(\alpha) \cdot n^{t+1}$. Otherwise, the problem is $\#\mathsf{W}[1]$-hard parameterized by $|\alpha|$ and does not have $f(\alpha) \cdot n^{o(t/\log t)}$ time algorithms under $\#\mathsf{ETH}$.*

With respect to fixed-parameter tractability vs. $\#\mathsf{W}[1]$-hardness, this theorem fully classifies the problems $\#\mathsf{Ind}(\mathcal{A})$ for fixed classes of linear combinations $\mathcal{A}$. Of course we have similar (equivalent) formulations for $\#\mathsf{Sub}(\mathcal{A})$ and $\#\mathsf{Hom}(\mathcal{A})$, and thus we generalize the dichotomy theorems for subgraphs (Theorem 1.5), homomorphisms (Theorem 1.8), and induced subgraphs (Theorem 1.10). The dichotomy criterion is somewhat indirect; it addresses $\mathcal{A}$ only through its representation as a linear combination of homomorphism numbers. However, we do not believe that there is a more 'native' dichotomy criterion on $\mathcal{A}$ since seemingly complicated linear combinations can turn out to be easy – Example 1.12 gives an indication of this phenomenon; perturbing the coefficients just a tiny bit can turn a computationally easy linear combination into one that is hard.

Nevertheless, we can exhibit some interesting sufficient conditions. For example, for the problem $\#\mathsf{Sub}(\mathcal{A})$, if all linear combinations of $\mathcal{A}$ in fact feature exactly one pattern (as in the situation of Theorem 1.5), then the linear combination re-expressed over homomorphisms uses graphs of unbounded treewidth if and only if the patterns in $\mathcal{A}$ have bounded vertex-cover number. We can hence recover Theorem 1.5 from Theorem 1.13.

## 1.5 Counting vertex-colored subgraphs

The techniques introduced above are sufficiently robust to handle generalizations to, e.g., the setting of vertex-colored subgraphs, where the vertices of $H$ and $G$ have colors and we count only subgraphs of $G$ with isomorphisms to $H$ that respect colors. A dichotomy for the special case of *colorful* patterns, where every vertex of the pattern $H$ has a different color, follows from earlier results by observing that embeddings and homomorphisms are the same for colorful patterns. For colorful patterns, bounded treewidth is the tractability criterion.

**Theorem 1.14 ([19, 17, 45]).** *Let $\mathcal{H}$ be a recursively enumerable class of colorful vertex-colored graphs. If $\mathcal{H}$ has bounded treewidth, then the problem $\#\mathsf{Sub}(\mathcal{H})$ of counting colorful subgraphs from $\mathcal{H}$ in vertex-colored host graphs is polynomial-time solvable. Otherwise, it is $\#\mathsf{W}[1]$-complete when parameterized by $|V(H)|$.*

Theorems 1.5 and 1.14 characterize the two extreme cases of counting colored subgraphs: the uncolored and the fully colorful cases. But there is an entire spectrum of colored problems in between these two extremes. What happens when we consider vertex-colored graphs with some colors appearing on more than one vertex? As we gradually move from colorful to uncolored graphs, where exactly is the point when a jump in complexity occurs? Answering such questions can be nontrivial even for simple patterns such as paths and matchings and can depend very much on how the colors appear on the pattern. Fortunately, by a basis change to (vertex-colored) homomorphisms via (2), we can answer such questions as easily as in the uncolored setting. The only technical change required is that we should consider only partitions $\rho$ that respect the coloring of $H$, that is, the vertices of $H$ that end up in the same block should have the same color. With these modifications, we can derive the following corollary from Theorem 1.13

**Theorem 1.15.** *Let $\mathcal{H}$ be a recursively enumerable class of vertex-colored patterns (or linear combinations thereof) and let $\mathcal{A}_{hom}$ be the class of linear combinations of homomorphisms derived from $\mathcal{H}$ by the identity (2) as discussed above. If there is a finite bound on the treewidth of graphs in $\mathcal{A}_{hom}$, then $\#\mathsf{Sub}(\mathcal{H})$ is FPT. Otherwise, the problem is $\#\mathsf{W}[1]$-hard when parameterized by $|V(H)|$.*

Theorem 1.15 raises a number of questions. First, being a corollary of Theorem 1.13, the tractability criterion is quite indirect, whereas we may want to have a more direct structural understanding of the FPT

cases. Secondly, Theorem 1.15 does not tell us whether the FPT cases are actually polynomial-time solvable or not. It is quite remarkable that in Theorems 1.5 and 1.14, all FPT cases are actually polynomial-time solvable, leaving no room for "true" FPT cases that are not polynomial-time solvable. It turns out however that, if we consider vertex-colored patterns in their full generality, then such pattern classes actually *do* appear. A prime example of this phenomenon is the case of *half-colorful matchings*, which are vertex-colored $k$-matchings such that one endpoint of each edge $e_i$ for $i \in \{1, \ldots, k\}$ is colored with 0, while the other is colored with $i$. Since counting perfect matchings in bipartite graphs is #P-hard, a trivial argument shows that counting half-colorful matchings is also #P-hard: if a bipartite graph with $n + n$ vertices is colored such that one part has color 0 and each vertex of the other part has a distinct color from 1 to $n$, then the number of half-colorful matchings of size $n$ is exactly the number of perfect matchings. On the other hand, it is not difficult to show that counting the number of half-colorful matchings of size $k$ in a graph colored with colors 0, 1, ..., $k$ is fixed-parameter tractable. It is essentially a dynamic programming exercise: for any subgraph $H' \subseteq H$ of the half-colorful matching and for any integer $i$, we want to compute the number of subgraphs of $G$ isomorphic to $H'$ that are allowed to use only the first $i$ vertices of color class 0.

We give a complete classification of the polynomial-time solvable cases of counting colored patterns from a class $\mathcal{H}$. For classes of patterns (but not linear combinations thereof), we refine the FPT cases of Theorem 1.15 into two classes: the polynomial-time solvable cases, and those that are not polynomial-time solvable, assuming the *Nonuniform Counting Exponential Time Hypothesis*. This shows that the existence of half-colorful matchings is the canonical reason why certain classes of patterns require dynamic programming and therefore the full power given by the definition of FPT: those cases are polynomial where the size of the largest half-colorful matching appearing as a subgraph is at most logarithmic in the size of the pattern.

### Organization of the paper

In §2, we provide basics on parameterized complexity and the graph-theoretical notions used in this paper. We formalize *graph motif parameters* in §3, and in §3.1 we show how to switch between different useful representations of graph motif parameters. In §3.2, we then address computational aspects of graph motif parameters. These results are first put to use in §4, where we count subgraph patterns by reduction to homomorphisms. In §5, we prove hardness results for linear combinations of subgraphs and induced subgraphs under #ETH and FPT $\neq$ #W[1]. Finally, we prove our results for counting vertex-colored subgraphs in §6.

## 2 Preliminaries

For a proposition $P$, we use the *Iverson bracket* $[P] \in \{0, 1\}$ to indicate whether $P$ is satisfied. For a potentially infinite matrix $M$, a *principal submatrix* $M_S$ is a submatrix of $M$ where the selected row and column index sets are the same set $S$.

### 2.1 Parameterized complexity theory

We refer to the textbooks [18, 24, 48] for background on parameterized complexity theory. Briefly, a *parameterized counting problem* is a function $\Pi : \{0, 1\}^* \to \mathbb{N}$ that is endowed with a *parameterization* $\kappa : \{0, 1\}^* \to \mathbb{N}$; it is *fixed-parameter tractable* (FPT) if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm to compute $\Pi(x)$ in time $f(k) \cdot \text{poly}(n)$, where $n = |x|$ and $k = \kappa(x)$.

A *parameterized Turing reduction* is a Turing reduction from a parameterized problem $(\Pi, \kappa)$ to a parameterized problem $(\Pi', \kappa')$ such that the reduction runs in $f(\kappa(x)) \cdot \text{poly}(|x|)$ time on instances $\Pi(x)$ and each oracle query $\Pi'(y)$ satisfies $\kappa'(y) \leq g(k)$. Here, both $f$ and $g$ are computable functions. A parameterized problem is #W[1]*-hard* if there is a parameterized Turing reduction from the problem of counting the $k$-cliques in a given graph; since it is believed that the latter does not have an FPT-algorithm, #W[1]-hardness is a strong indicator that a problem is not FPT.

The exponential time hypothesis (ETH) by Impagliazzo and Paturi [30] asserts that satisfiability of 3-CNF formulas cannot be decided substantially faster than by trying all possible assignments. The counting version of this hypothesis [20] states that there is a constant $c > 0$ such that no deterministic algorithm can compute #3-SAT in time $\exp(c \cdot n)$, where $n$ is the number of variables.

Chen et al. [10] proved that ETH implies the hypothesis that there is no $f(k) \cdot n^{o(k)}$-time algorithm to decide whether an $n$-vertex graph $G$ contains a $k$-clique. Their reduction is parsimonious, so #ETH rules out $f(k) \cdot n^{o(k)}$ time algorithms for counting $k$-cliques.

## 2.2 Graphs, subgraphs, and homomorphisms

Let $\mathcal{G}$ be the set of all labeled, finite, undirected, and simple graphs; in particular, these graphs contain neither loops nor parallel edges. That is, there is a suitable fixed and countably infinite universe $U$, and $\mathcal{G}$ contains all finite graphs $G$ with vertex set $V(G) \subseteq U$ and edge set $E(G) \subseteq \binom{V(G)}{2}$.

**Subgraphs.** If $G$ is a graph, a *subgraph* $F$ of $G$ is a graph with $V(F) \subseteq V(G)$ and $E(F) \subseteq E(G)$, and $F$ is an *induced subgraph* of $G$ if it is a subgraph with the additional property that, for all $uv \notin E(F)$ we have $uv \notin E(G)$. The set of subgraphs of $G$ that are isomorphic to $H$ is denoted with $\mathrm{Sub}(H \to G)$, and the set of induced subgraphs of $G$ that are isomorphic to $H$ is denoted with $\mathrm{IndSub}(H \to G)$.

**Homomorphisms and related notions.** If $H$ and $G$ are graphs, a *homomorphism* from $H$ to $G$ is a function $f : V(H) \to V(G)$ such that edges map to edges under $f$. That is, for all $\{u, v\} \in E(H)$, we have $\{f(u), f(v)\} \in E(G)$. The set of all homomorphisms from $H$ to $G$ is denoted with $\mathrm{Hom}(H \to G)$. *Embeddings* are injective homomorphisms, and we denote the corresponding set with $\mathrm{Emb}(H \to G)$. *Strong embeddings* are embeddings with the additional property that non-edges map to non-edges, that is, $\{f(u), f(v)\} \notin E(G)$ holds for all $\{u, v\} \notin E(G)$. We denote the set of strong embeddings with $\mathrm{StrEmb}(H \to G)$. A homomorphism $f \in \mathrm{Hom}(H \to G)$ is *surjective* if it hits all vertices and edges of $G$, that is, $f(V(H)) = V(G)$ and $f(E(H)) = E(G)$ hold. Note that this is a stronger requirement than $f$ being surjective on its codomain. An *isomorphism* from $H$ to $G$ is a strong embedding from $H$ to $G$ that is also surjective, and it is an *automorphism* if additionally $H = G$ holds. We write $H \simeq G$ if $H$ and $G$ are isomorphic.

**Colored graphs.** We also use vertex-colored graphs $G$, where each vertex has a color from a finite set $C$ of colors via a function $f : V(G) \to C$. We note that such a coloring is not necessarily proper, in the sense that any two adjacent vertices need to receive distinct colors. Each set $V_i(G) := f^{-1}(i)$ for $i \in C$ is a *color class* of $G$. A subgraph $H$ of $G$ is called (vertex-)*colorful* if $V(H)$ intersects each color class in exactly one vertex.

**Treewidth.** A *tree decomposition* of a graph $G$ is a pair $(T, \beta)$, where $T$ is a tree and $\beta$ is a mapping from $V(T)$ to $2^{V(G)}$ such that, for all vertices $v \in V(G)$, the set $\{\, t \in V(T) : v \in \beta(t) \,\}$ is nonempty and connected in $T$, and for all edges $e \in E(G)$, there is some node $t \in V(T)$ such that $e \subseteq \beta(t)$. The set $\beta(t)$ is the *bag at* $t$. The *width* of $(T, \beta)$ is the integer $\max\{\, |\beta(t)| - 1 : t \in V(T) \,\}$, and the *treewidth* $\mathrm{tw}(G)$ of a graph $G$ is the minimum possible width of any tree decomposition of $G$.

It will be convenient for us to view the tree $T$ as being directed away from the root, and we define the following mappings $\sigma, \gamma, \alpha : V(T) \to 2^{V(G)}$ for all $t \in V(T)$:

(the *separator at* $t$)
$$\sigma(t) = \begin{cases} \emptyset & \text{if } t \text{ is the root of } T, \\ \beta(t) \cap \beta(s) & \text{if } s \text{ is the parent of } t \text{ in } T, \end{cases} \tag{4}$$

(the *cone at* $t$)
$$\gamma(t) = \bigcup_{u \text{ is a descendant of } t} \beta(u), \tag{5}$$

(the *component at* $t$)
$$\alpha(t) = \gamma(t) \setminus \sigma(t). \tag{6}$$

The *adhesion* of $(T, \beta)$ is defined as $\max\{\, |\sigma(t)| : t \in V(T) \,\}$. The following conditions are easily checked:

(TD.1) $T$ is a directed tree.

(TD.2) For all $t \in V(T)$ we have $\alpha(t) \cap \sigma(t) = \emptyset$ and $N^G(\alpha(t)) \subseteq \sigma(t)$.

(TD.3) For all $t \in V(T)$ and $u \in N^T_+(t)$ we have $\alpha(u) \subseteq \alpha(t)$ and $\gamma(u) \subseteq \gamma(t)$.

(TD.4) For all $t \in V(T)$ and all distinct $u_1, u_2 \in N^T_+(t)$ we have $\gamma(u_1) \cap \gamma(u_2) = \sigma(u_1) \cap \sigma(u_2)$.

(TD.5) For the root $r$ of $T$ we have $\sigma(r) = \emptyset$ and $\alpha(r) = V(G)$.

Conversely, consider any triple $(T, \sigma, \alpha)$ where $T$ is a directed graph and $\sigma, \alpha$ are functions of type $V(T) \to 2^{V(G)}$. We can then define functions $\gamma, \beta : V(T) \to 2^{V(G)}$ such that, for all $t \in V(T)$, we have $\gamma(t) = \sigma(t) \cup \alpha(t)$ and $\beta(t) = \gamma(t) \setminus \bigcup_{u \in N_+^T(t)} \alpha(u)$. If (TD.1)–(TD.5) are satisfied, then it can be verified that $(T, \beta)$ is a tree decomposition (see [28] for a proof). Hence (TD.1)–(TD.5) yield an alternative definition of tree decompositions, which we may use as is convenient.

# 3 The space of graph motif parameters

We develop our interpretation of the general setup of Lovász [43]. To this end, it will be useful to consider unlabeled graphs: For concreteness, we say that a graph $H \in \mathcal{G}$ is *unlabeled* if it is *canonically labeled*, that is, if it is the lexicographically first graph that is isomorphic to $H$. Then the set $\mathcal{G}^*$ of unlabeled graphs is the subset of $\mathcal{G}$ that contains exactly the canonically labeled graphs. *Graph parameters* are functions $f : \mathcal{G} \to \mathbb{Q}$ that are invariant under isomorphisms, and we view them as functions $f : \mathcal{G}^* \to \mathbb{Q}$.

For all $H, G \in \mathcal{G}^*$, we define $\mathrm{IndSub}(H, G)$ as the number of (labeled) induced subgraphs of $G$ that are isomorphic to $H$. We can view this function as an infinite matrix with indices from $\mathcal{G}^* \times \mathcal{G}^*$ and entries from $\mathbf{N}$. The matrix indices are ordered according to some fixed total order on $\mathcal{G}^*$ that respects the total size $|V(F)| + |E(F)|$ of the graphs $F \in \mathcal{G}^*$. Among graphs of the same total size, ties may be broken arbitarily. If $H$ and $G$ are graphs such that $H$ has larger total size than $G$, then $H$ cannot be an induced subgraph of $G$, that is, $\mathrm{IndSub}(H, G) = 0$. We conclude that IndSub is an upper triangular matrix.

We define graph motif parameters as graph parameters that can be expressed as finite linear combinations of induced subgraph numbers. To obtain a clean formulation in terms of linear algebra, we represent these linear combinations as infinite vectors $\alpha \in \mathbb{Q}^{\mathcal{G}^*}$ of finite support. Here, the *support* $\mathrm{supp}(\alpha)$ of a vector $\alpha$ is the set of all graphs $F \in \mathcal{G}^*$ with $\alpha_F \neq 0$.

**Definition 3.1.** *A graph parameter $f : \mathcal{G}^* \to \mathbb{Q}$ is a* graph motif parameter *if there is a vector $\alpha \in \mathbb{Q}^{\mathcal{G}^*}$ with finite support such that $f(G) = \sum_{F \in \mathcal{G}^*} \alpha_F \cdot \mathrm{IndSub}(F, G)$ holds for all $G \in \mathcal{G}^*$.*

If we interpret $f$ and $\alpha$ as row vectors, this definition can also be phrased as requiring $f = \alpha \cdot \mathrm{IndSub}$ for $\alpha$ of finite support. Here, for two vectors $\alpha, \beta \in \mathbf{Q}^{\mathcal{G}^*}$, we define the scalar product $(\alpha, \beta)$ as $\sum_{F \in \mathcal{G}^*} \alpha_F \cdot \beta_F$ if this sum is defined.[1] The definition of the matrix-vector and matrix-matrix products is then as usual.

The set of all graph motif parameters, endowed with the operations of scalar multiplication and pointwise addition, forms an infinite-dimensional vector space. More specifically, it is the finitely supported row-span of the matrix IndSub. We remark that even if we drop the condition of $\alpha$ being finitely supported, the scalar product $\alpha \cdot \mathrm{IndSub}$ remains well-defined, since every column of IndSub has finite support (as a consequence of every graph $G$ having only finitely many induced subgraphs $H$). In fact, it can be verified that *every* graph parameter $f$ can be written as $\alpha \cdot \mathrm{IndSub}$ for some $\alpha$.

## 3.1 Relations between graph motif parameters

One may wonder why we chose induced subgraph numbers for our definition of graph motif parameters and not, say, the numbers of subgraphs or homomorphisms. It turns out that all of these choices lead to the same vector space: Subgraph and homomorphism numbers are graph motif parameters themselves, and indeed they also span the space of graph motif parameters.

Since some properties of graph motif parameters, such as their computational complexity, turn out to be easier to understand over the homomorphism basis, we show explicitly how to perform basis transformations. To this end, we first present the basis transformation between subgraphs and induced subgraphs, then we proceed with the basis transformation between subgraphs and homomorphisms.

**Subgraphs and induced subgraphs.** For graphs $H, G \in \mathcal{G}^*$, we first show how to express $\mathrm{Sub}(H, G)$ as a linear combination of numbers $\mathrm{IndSub}(F, G)$. To this end, note that every subgraph copy of $H$ in $G$ is contained in some induced subgraph $F$ of $G$ on $|V(H)|$ vertices. This induced subgraph $F$ is isomorphic to a supergraph of $H$, and we call these supergraphs $F$ *extensions*. More precisely, an *extension* of $H$ is a (labeled) supergraph $X$ of $H$ with $V(X) = V(H)$.

[1] In this paper, such scalar products degenerate into finite sums, since the support of at least one of the vectors will be finite.

Note that $H$ might have different extensions that are isomorphic. Thus, given a graph $F$, let $\mathrm{Ext}(H,F)$ be the number of extensions $X$ of $H$ that are isomorphic to $F$; equivalently, we have

$$\mathrm{Ext}(H,F) = [|V(H)| = |V(F)|] \cdot \mathrm{Sub}(H,F),$$

and we thus obtain

$$\mathrm{Sub}(H,G) = \sum_{F \in \mathcal{G}^*} \mathrm{Ext}(H,F) \cdot \mathrm{IndSub}(F,G). \tag{7}$$

Every graph $H$ admits only finitely many extensions, and so the function $\mathrm{Sub}(H,\star)$ is a graph motif parameter for every fixed $H$. In matrix notation, the identity (7) takes on the concise form

$$\mathrm{Sub} = \mathrm{Ext} \cdot \mathrm{IndSub}\,. \tag{8}$$

Since Sub, Ext, and IndSub are upper triangular matrices with diagonal entries equal to 1, every finite principal submatrix of any of these triangular matrices is invertible; indeed the entire matrix Ext has an inverse with $\mathrm{IndSub} = \mathrm{Ext}^{-1} \cdot \mathrm{Sub}$. This implies that Sub also *spans* the space of graph motif parameters: Every function $\mathrm{IndSub}(H,\star)$ is a finite linear combination of functions $\mathrm{Sub}(F,\star)$ with coefficients $\mathrm{Ext}^{-1}(H,F)$.

We remark that the values of the coefficients $\mathrm{Ext}^{-1}(H,F)$ are actually well understood: The identity (7) can be interpreted as a zeta transform over the subset lattice [8, eq. (13) and (14)], so we can perform Möbius inversion to prove that

$$\mathrm{Ext}^{-1}(H,F) = (-1)^{|E(F)|-|E(H)|} \cdot \mathrm{Ext}(H,F) \tag{9}$$

holds for all graphs $H$ and $F$. We will use this identity later to check that $\mathrm{Ext}^{-1}(H,F) \neq 0$ holds for specific pairs $(H,F)$ of graphs.

**Homomorphisms and subgraphs.** We wish to express $\mathrm{Hom}(H,G)$ as a finitely supported linear combination $\sum_F \alpha_F \mathrm{Sub}(F,G)$ of subgraph numbers. For a homomorphism $h$ from $H$ to $G$, let $I$ be the image of $h$, that is, the graph with vertex set $f(V(H))$ and edge set $f(E(H))$; we observe that $h$ is a surjective homomorphism from $H$ to $I$ and $I$ is a subgraph of $G$. That is, every homomorphism from $H$ to $G$ can be written as a surjective homomorphism into a subgraph $F$ of $G$. Writing $\mathrm{Surj}(H,F)$ for the number of surjective homomorphisms from $H$ to $F$, we have

$$\mathrm{Hom}(H,G) = \sum_{F \in \mathcal{G}^*} \mathrm{Surj}(H,F) \cdot \mathrm{Sub}(F,G). \tag{10}$$

Note that $\mathrm{Surj}(H,F) = 0$ holds if $H$ is smaller than $F$ in total size. Thus, analogously to the case of subgraphs, for each fixed $H$, we have $\mathrm{Surj}(H,F) \neq 0$ only for finitely many $F \in \mathcal{G}^*$. Therefore $\mathrm{Hom}(H,\star)$ is indeed a graph motif parameter for every fixed $H$. In matrix notation, we have

$$\mathrm{Hom} = \mathrm{Surj} \cdot \mathrm{Sub}, \tag{11}$$

where Surj is a lower triangular matrix. Moreover, the diagonal entries of Surj satisfy $\mathrm{Surj}(F,F) = \mathrm{Aut}(F) \neq 0$, and hence each finite principal submatrix is invertible. In fact the entire matrix has an inverse $\mathrm{Surj}^{-1}$ satisfying $\mathrm{Sub} = \mathrm{Surj}^{-1} \cdot \mathrm{Hom}$, and so Hom spans the space of graph motif parameters as well.

The inverse of Surj can be understood in terms of a Möbius inversion on a partition lattice. To see this, let us first consider the support of the vector $\mathrm{Surj}(H,\star)$, that is, the set of all unlabeled graphs that are homomorphic images of $H$. This set will play an important role throughout this paper, and we call it the *spasm* of $H$:

$$\mathrm{Spasm}(H) = \{F \in \mathcal{G}^* : \mathrm{Surj}(H,F) > 0\}. \tag{12}$$

In a more graph-theoretical interpretation, the elements in the spasm of $H$ can also be understood as the unlabeled representatives of all graphs that can be obtained from $H$ by merging independent sets. We make this more formal in the following definition. For each $H \in \mathcal{G}$, let $\mathrm{Part}(H)$ be the set of all partitions of $V(H)$, where a partition is a set of disjoint non-empty subsets $B \subseteq V(H)$ whose union equals $V(H)$.

**Definition 3.2.** *For a graph $H \in \mathcal{G}$ and a partition $\rho \in \mathrm{Part}(H)$, the* quotient *$H/\rho$ is the graph obtained by identifying, for each block $B \in \rho$, the vertices in $B$ to a single vertex. This process may create loops or parallel edges; we keep loops intact in $H/\rho$, and we turn parallel edges into simple edges.*

For $F \in \mathcal{G}^*$, we have $F \in \mathrm{Spasm}(H)$ if and only if there is a partition $\rho \in \mathrm{Part}(H)$ with $F \simeq H/\rho$. Note that graphs $F \in \mathcal{G}^*$ does not have loops, since we explicitly restricted the graphs in $\mathcal{G}^*$ to be simple. Consequently, $F \simeq H/\rho$ can only hold if all blocks of $\rho$ are independent sets of $H$, that is, if $\rho$ represents a proper vertex-coloring of $H$.

Every surjective homomorphism from $H$ to $F$ can be interpreted as a pair $(\rho, \pi)$ where $H/\rho \simeq F$ and $\pi \in \mathrm{Aut}(F)$. Hence we have

$$\mathrm{Surj}(H,F) = \#\mathrm{Aut}(F) \cdot \sum_{\rho \in \mathrm{Part}(H)} [H/\rho \simeq F]\,, \tag{13}$$

For two partitions $\rho, \rho' \in \mathrm{Part}(H)$, we write $\rho \geq \rho'$ if $\rho$ is coarser than $\rho'$, that is, if every block of $\rho'$ is contained in a block of $\rho$. This partial order gives rise to the *partition lattice* $(\mathrm{Part}(H), \geq)$ whose minimal element $\bot$ is the finest partition, i.e., the partition whose blocks all have size one. Now (10) can be viewed as a zeta-transformation on the partition lattice: Let $H$ and $G$ be fixed graphs. Let $f(\rho) = \#\mathrm{Emb}(H/\rho \to G)$. Then consider its *upwards zeta-transform* on the partition lattice, i.e., the function $\hat{f}$ defined by

$$\hat{f}(\rho) = \sum_{\rho' \geq \rho} f(\rho')\,.$$

We observe that $\mathrm{Hom}(H,G) = \hat{f}(\bot)$. By Möbius inversion, we get (see also [8, eq. (15)]):

$$f(\rho) = \sum_{\rho' \geq \rho} (-1)^{|\rho|-|\rho'|} \cdot \Big( \prod_{B \in \rho'} (\lambda(\rho, \rho', B) - 1)! \Big) \cdot \hat{f}(\rho')\,,$$

where $\lambda(\rho, \rho', B)$ is the number of blocks $C \in \rho$ with $C \subseteq B$. We set $\rho = \bot$ and collect terms $\rho'$ that lead to isomorphic graphs $H/\rho'$. Note that, for a given graph isomorphism type, all terms leading to this type are non-zero and have the same sign. We obtain

$$\mathrm{Surj}^{-1}(H,F) = \frac{(-1)^{|V(H)|-|V(F)|}}{\#\mathrm{Aut}(H)} \cdot \sum_{\substack{\rho \in \mathrm{Part}(H) \\ H/\rho \simeq F}} \prod_{B \in \rho} (|B|-1)!\,. \tag{14}$$

In particular, this yields $\mathrm{Surj}^{-1}(H,F) \neq 0$ if and only $\mathrm{Surj}(H,F) \neq 0$; that is, $F \in \mathrm{Spasm}(H)$ is equivalent to $\mathrm{Surj}^{-1}(H,F) \neq 0$. This observation will be crucial in the proof of our hardness result.

While we established before that Hom spans the space of graph motif parameters, it is not immediately clear that the homomorphism numbers form a *basis*, that is, that the rows of Hom are linearly independent. The following proposition on the invertibility of certain principal submatrices will be important for our hardness results, and it implies that the rows of Hom are linearly independent with respect to finite linear combinations.

**Lemma 3.3 (Proposition 5.43 in [43]).** *Let $S \subseteq \mathcal{G}^*$ be a finite set of graphs that is closed under surjective homomorphisms, that is, we have $\mathrm{Spasm}(H) \subseteq S$ for all $H \in S$. Then the principal submatrix $\mathrm{Hom}_S$ of Hom is invertible and satisfies $\mathrm{Hom}_S = \mathrm{Surj}_S \cdot \mathrm{Sub}_S$.*

*Proof.* Let $F, G \in S$ and consider the expansion of $\mathrm{Hom}(H,G)$ from (10). Since $S$ is closed under surjective homomorphisms, only terms with $F \in S$ contribute to the sum. Hence we have $\mathrm{Hom}_S = \mathrm{Surj}_S \cdot \mathrm{Sub}_S$. Since $\mathrm{Surj}_S$ and $\mathrm{Sub}_S$ both are triangular matrices with non-zero diagonal entries, they are invertible, and consequently so is their product $\mathrm{Hom}_S$. □

See Figure 2 for an example of Lemma 3.3. Let us also note three simple and useful properties that $\mathrm{Spasm}(H)$ inherits from $H$.

**Fact 3.4.** *For all graphs $H$, the following properties hold:*

1. *Every graph $F \in \mathrm{Spasm}(H)$ has at most $|V(H)|$ vertices and at most $|E(H)|$ edges.*
2. *If $H$ has a vertex-cover of size $b \in \mathbb{N}$, then every graph $F \in \mathrm{Spasm}(H)$ has a vertex-cover of size $b$.*

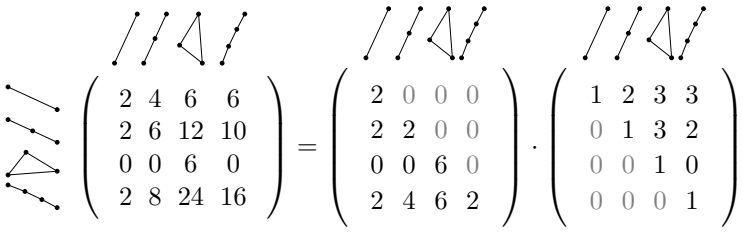


Figure 2: The matrix identity $\mathrm{Hom}_S = \mathrm{Surj}_S \cdot \mathrm{Sub}_S$, where $S$ is the spasm of the path , consisting of the four graphs , , , .

3. *If $H$ contains a matching with $k \in \mathbb{N}$ edges, then every graph with $k$ edges (and no isolated vertices) can be found as a minor of some graph in* $\mathrm{Spasm}(H)$.

*Proof.* Only the third claim merits some explanation. If $M_k$ is the (not necessarily induced) $k$-matching in $H$ and $F$ is the $k$-edge graph we want to find, then we determine an arbitrary surjective homomorphism $g : V(M_k) \to V(F)$, which hits every edge of $F$. We are allowed to contract edges (due to the minor operation) or consolidate non-edges of $H$ (due to the quotient operation) to build $F$. To this end, we simply identify all vertices of $g^{-1}(i)$, for each $i \in V(F)$, and delete all vertices in $V(H) \setminus V(M_k)$. □

**Embeddings and strong embeddings.** For completeness, we define matrices for embeddings and strong embeddings. For $H, G \in \mathcal{G}^*$, let $\mathrm{Emb}(H,G)$, $\mathrm{StrEmb}(H,G)$, and $\mathrm{Iso}(H,G)$ be the number of embeddings, strong embeddings, and isomorphisms from $H$ to $G$, respectively. Clearly Iso is a diagonal matrix with $\mathrm{Iso}(F,F) = \mathrm{Aut}(F)$. We have $\mathrm{Emb} = \mathrm{Iso} \cdot \mathrm{Sub}$ and $\mathrm{StrEmb} = \mathrm{Iso} \cdot \mathrm{IndSub}$.

## 3.2 The complexity of graph motif parameters

Several computational problems can be associated with graph motif parameters, but perhaps the most natural one is the *evaluation problem*: Given as input a graph motif parameter $f : \mathcal{G}^* \to \mathbb{Q}$ and a graph $G \in \mathcal{G}^*$, compute the value $f(G)$.

This problem requires a suitable representation of the input $f$, and while we could choose any basis to represent $f$, the homomorphism basis turns out to be particularly useful for algorithmic purposes. That is, in this subsection, we represent graph motif parameters $f$ as vector-matrix products $f = \alpha \cdot \mathrm{Hom}$ for finitely supported row vectors $\alpha \in \mathbb{Q}^{\mathcal{G}^*}$. The input is then the coefficient vector $\alpha$, encoded as a list of pairs $(F, \alpha_F)$ for $F \in \mathrm{supp}(\alpha)$. Let $|\alpha|$ be the description length of $\alpha$, and let $\mathrm{tw}(\alpha)$ be the maximum treewidth $\mathrm{tw}(F)$ among all graphs $F \in \mathrm{supp}(\alpha)$. The following lemma is immediate:

**Lemma 3.5 (Algorithm).** *There is a deterministic algorithm that is given $\alpha$ and $G$ to compute $(\alpha \cdot \mathrm{Hom})(G)$ in time $g(\alpha) + \mathrm{poly}(|\alpha|) \cdot |V(G)|^{\mathrm{tw}(\alpha)+1}$ for some computable function $g$ depending only on $\alpha$.*

*Proof.* For each $F \in \mathrm{supp}(\alpha)$, run the algorithm from Proposition 1.6 to compute $\mathrm{Hom}(F,G)$ in time $\exp(O(k)) + \mathrm{poly}(k) \cdot n^{\mathrm{tw}(F)+1}$, where $k = |V(F)|$ and $n = |V(G)|$. Then output $\sum_F \alpha_F \cdot \mathrm{Hom}(F,G)$. □

We could choose other representations for $f$, such as coefficient vectors $\alpha$ with $f = \alpha \cdot \mathrm{Sub}$ or $f = \alpha \cdot \mathrm{IndSub}$. Switching between these representations only adds an overhead of $g(\alpha)$ in the running time, for some function $g$, as can be seen from §3.1.

It is clear that the generic evaluation problem for graph motif parameters is #W[1]-hard, since it subsumes counting $k$-cliques as a special case. The following reduction shows that evaluating $f$ with $f = \alpha \cdot \mathrm{Hom}$ is at least as hard as every individual homomorphism problem $\mathrm{Hom}(F, \star)$ for $F \in \mathrm{supp}(\alpha)$. That is, if a linear combination of homomorphisms contains a "hard" pattern graph, then the entire linear combination is "hard".

**Lemma 3.6 (Extracting summands).** *There is a deterministic Turing reduction that is given a finitely supported vector $\alpha \in \mathbb{Q}^{\mathcal{G}^*}$, a graph $F \in \mathrm{supp}(\alpha)$, and a graph $G \in \mathcal{G}^*$ to compute the number* $\mathrm{Hom}(F,G)$

*with an oracle for the function* $(\alpha \cdot \mathrm{Hom})(\star)$. *The reduction runs in time* $g(\alpha) \cdot \mathrm{poly}(|V(G)|)$ *for some computable function* $g$, *makes at most* $g(\alpha)$ *queries to* $(\alpha \cdot \mathrm{Hom})(\star)$, *and each queried graph has at most* $\max_{H \in \mathrm{supp}(\alpha)} |V(H)| \cdot |V(G)|$ *vertices.*

*Proof.* On input $(\alpha, F, G)$, the reduction only makes queries of the form $(\alpha \cdot \mathrm{Hom})(G \times X)$ for graphs $X$, where $G \times X$ is the categorical product, that is, the graph with vertex set $V(G) \times V(X)$ such that $(v, x)$ and $(v', x')$ are adjacent in $G \times X$ if and only if $vv' \in E(G)$ and $xx' \in E(X)$. The following holds [43, (5.30)]:

$$\mathrm{Hom}(F, G \times X) = \mathrm{Hom}(F, G) \cdot \mathrm{Hom}(F, X). \tag{15}$$

Using this identity for various $X$, we aim at setting up a linear equation system that can be solved uniquely for $\mathrm{Hom}(F, G)$. We expand the sum $(\alpha \cdot \mathrm{Hom})(G \times X)$ and apply (15) to obtain

$$\sum_H \alpha_H \cdot \mathrm{Hom}(H, G) \cdot \mathrm{Hom}(H, X) = (\alpha \cdot \mathrm{Hom})(G \times X). \tag{16}$$

For each graph $X$, the reduction can compute the right side of this linear equation using the oracle, and it can determine the numbers $\alpha_H$ and $\mathrm{Hom}(H, X)$ in some time $f(\alpha)$. It remains to choose a suitable set $S$ of graphs $X$ so that the resulting system of linear equations can be uniquely solved for $\mathrm{Hom}(F, G)$.

Let $S = \bigcup_{H \in \mathrm{supp}(\alpha)} \mathrm{Spasm}(H)$ be the closure of $\mathrm{supp}(\alpha)$ under spasms. By Lemma 3.3, the matrix $\mathrm{Hom}_S$ is invertible. Moreover, we have $\alpha \cdot \mathrm{Hom} = \alpha_S \cdot \mathrm{Hom}_S$. We rewrite (16) as $\mathrm{Hom}_S \cdot x = b$ where $b \in \mathbb{Q}^S$ and $\mathrm{Hom}_S \in \mathbb{Q}^{S \times S}$ represent the known quantities with $b_X = (\alpha \cdot \mathrm{Hom})(G \times X)$ and $\mathrm{Hom}_S(H, X) = \mathrm{Hom}(H, X)$, and the vector $x \in \mathbb{Q}^S$ represents the indeterminates with $x_H = \alpha_H \cdot \mathrm{Hom}(H, G)$. We have $x = (\mathrm{Hom}_S)^{-1} \cdot b$, so we can solve uniquely for the indeterminates. In particular, we can compute $\mathrm{Hom}(F, G) = x_F / \alpha_F$, since $\alpha_F \neq 0$ holds by assumption.

The set $S$ and the matrices $\mathrm{Hom}_S$ and $(\mathrm{Hom}_S)^{-1}$ can be computed in time $g(\alpha)$ for some computable function $g$. For the vector $b$, we need to compute the product graphs and query the oracle. The number of queries is $|S|$ and thus bounded by $g(\alpha)$. Overall, the reduction takes time $g(\alpha) \cdot \mathrm{poly}(|V(G)|)$. □

Analogously to the problems $\#\mathsf{Sub}(\mathcal{H})$ in §1.1, we consider restricted classes of graph motif parameters (represented as linear combinations of homomorphism numbers) to obtain more expressive hardness results.

**Definition 3.7.** *Let* $\mathcal{A} \subseteq \mathbb{Q}^{\mathcal{G}^*}$ *be a set of finitely supported vectors. We let* $\#\mathsf{Hom}(\mathcal{A})$ *be the computational problem whose task is to compute* $(\alpha \cdot \mathrm{Hom})(G)$ *on input* $\alpha \in \mathcal{A}$ *and* $G \in \mathcal{G}^*$.

We apply Lemma 3.6 to establish the hard cases of $\#\mathsf{Hom}(\mathcal{A})$.

**Lemma 3.8 (Hardness).** *Let* $\mathcal{A} \subseteq \mathbb{Q}^{\mathcal{G}^*}$ *be a recursively enumerable class of finitely supported vectors. If* $\mathcal{A}$ *contains vectors of arbitrarily large treewidth* $\mathrm{tw}(\alpha)$, *then* $\#\mathsf{Hom}(\mathcal{A})$ *is* $\#\mathsf{W}[1]$*-hard when parameterized by* $|\alpha|$. *Moreover, the problem does not have* $g(\alpha) \cdot |V(G)|^{o(\mathrm{tw}(\alpha)/\log \mathrm{tw}(\alpha))}$ *time algorithms if* $\#\mathsf{ETH}$ *holds.*

*Proof.* For each $\alpha \in \mathcal{A}$, we select a graph $F_\alpha \in \mathrm{supp}(\alpha)$ of maximum treewidth. Then the set $\mathcal{F}$ with $\mathcal{F} = \{ F_\alpha : \alpha \in \mathcal{A} \}$ has unbounded treewidth. To prove the hardness, we provide a parameterized Turing reduction from $\#\mathsf{Hom}(\mathcal{F})$ to $\#\mathsf{Hom}(\mathcal{A})$. Since $\mathcal{F}$ has unbounded treewidth, Theorem 1.8 implies that $\#\mathsf{Hom}(\mathcal{F})$ is $\#\mathsf{W}[1]$-hard when parameterized by $|V(F)|$ and Proposition 1.9 implies it cannot be computed in time $g(F) \cdot |V(G)|^{o(\mathrm{tw}(F)/\log \mathrm{tw}(F))}$ for any computable $g$ if $\#\mathsf{ETH}$ holds.

Let $(F, G)$ with $F \in \mathcal{F}$ be an input for the reduction, whose goal is to compute $\mathrm{Hom}(F, G)$ with oracle access to $\#\mathsf{Hom}(\mathcal{A})$. First the reduction computes some $\alpha \in \mathcal{A}$ with $F_\alpha = F$. This is possible, since $\mathcal{A}$ is recursively enumerable. The reduction then applies the algorithm from Lemma 3.6 on input $(\alpha, F, G)$. The algorithm runs in time $g(\alpha) \cdot \mathrm{poly}(|V(G)|)$ for a computable function $g$ and makes queries to the function $(\alpha \cdot \mathrm{Hom})(\star)$; its output is the desired number $\mathrm{Hom}(F, G)$.

For the running time of the reduction, note that finding $\alpha$ takes time $h(k)$ for some computable function $h$, where $k = |V(F)|$. The algorithm from Lemma 3.6 runs in some time $g(\alpha) \cdot \mathrm{poly}(n) \leq h(k) \cdot \mathrm{poly}(n)$. Hence we indeed obtain a parameterized reduction from $\#\mathsf{Hom}(\mathcal{F})$ to $\#\mathsf{Hom}(\mathcal{A})$, which proves that $\#\mathsf{Hom}(\mathcal{A})$ is $\#\mathsf{W}[1]$-hard when parameterized by $|\alpha|$. Finally, we have $\mathrm{tw}(\alpha) = \mathrm{tw}(F)$, so if $\#\mathsf{Hom}(\mathcal{A})$ can be computed in time $f(\alpha) \cdot n^{o(\mathrm{tw}(\alpha)/\log \mathrm{tw}(\alpha))}$, then $\#\mathsf{Hom}(\mathcal{F})$ can be solved in time $f(F) \cdot n^{o(\mathrm{tw}(F)/\log \mathrm{tw}(F))}$, which by Proposition 1.9 is impossible if $\#\mathsf{ETH}$ holds. □

From the perspective of fine-grained complexity, for every *fixed* graph motif parameter $f$, Lemma 3.5 yields an algorithm for evaluating $f$ on $n$-vertex graphs in some time $O(n^c)$. Here, $c$ is a constant that depends on the largest treewidth in the homomorphism representation of $f$. To express this connection more precisely, given a graph motif parameter $f$, we define the constant

$$C(f) = \inf\{\, c \in \mathbb{R} : \ f \text{ can be computed in time } O(n^c) \ \}\,. \tag{17}$$

In particular, we can consider this constant for the graph parameters $\mathrm{Hom}(F,\star)$ for fixed graphs $F$. Then the constant $C(\mathrm{Hom}(F,\star))$ is the smallest possible exponent required for computing $\mathrm{Hom}(F,G)$ on $n$-vertex graphs $G$. The proof of Lemma 3.5 implies

$$C(f) \leq \max_{F \in \mathrm{supp}(\alpha)} C(\mathrm{Hom}(F,\star))\,,$$

where $\alpha \in \mathbb{Q}^{\mathcal{G}^*}$ is the representation of $f$ over the homomorphism basis, that is, the vector $\alpha$ with $f = \alpha \cdot \mathrm{Hom}$. Lemma 3.6 implies the corresponding lower bound, so in fact we have

$$C(f) = \max_{F \in \mathrm{supp}(\alpha)} C(\mathrm{Hom}(F,\star))\,, \tag{18}$$

Proposition 1.9 implies that $C(\mathrm{Hom}(F,\star)) \leq \mathrm{tw}(F) + 1$ holds. Under the assumption $\mathsf{FPT} \neq \#\mathsf{W}[1]$, Theorem 1.8 implies that $C(\mathrm{Hom}(F,\star))$ cannot be bounded by a universal constant, and under the stronger assumption #ETH, Proposition 1.9 implies that $C(\mathrm{Hom}(F,\star))$ is not bounded by $o(\mathrm{tw}(F)/\log \mathrm{tw}(F))$, even when $F$ is restricted to be from any fixed family $\mathcal{F}$ of graphs of unbounded treewidth. The $k$-clique hypothesis is that the current fastest algorithm for $k$-clique is optimal [1], which can be formalized as $C(\mathrm{Hom}(K_k,\star)) = \omega k/3$. These facts suggest that the representation of a graph motif parameter $f$ in the homomorphism basis and your favorite complexity hypotheses are all that is needed to understand the complexity of $f$ (concerning the exponent of $n$).

**Remark 3.9.** *Lovász's framework is easily adapted to the setting of vertex-colored pattern and host graphs. Here, we only wish to count homomorphisms (or embeddings, or strong embeddings) from $H$ into $G$ that map vertices of $H$ to vertices of $G$ of the same colors. In the definition of the spasm of $H$, one is then only allowed to identify non-adjacent vertices of the same color, that is, the allowed partitions $\rho$ are those where each block is a monochromatic independent set. The algorithm (Lemma 3.5) and hardness result (Lemma 3.8) can also be adapted to the setting of vertex-colored homomorphisms without modifications. We excluded these variants from the main text to simplify the presentation.*

# 4 Algorithms for counting subgraphs

In this section, we obtain algorithms for counting subgraphs and embeddings by expressing these quantities over the homomorphism basis via (10) and running an algorithm for counting homomorphisms. More concretely, recall that

$$\mathrm{Sub}(H,G) = \sum_F \mathrm{Surj}^{-1}(H,F) \cdot \mathrm{Hom}(F,G)\,. \tag{19}$$

We know from (14) that $\mathrm{Surj}^{-1}(H,F) \neq 0$ is equivalent to $F \in \mathrm{Spasm}(H)$. Hence for fixed patterns $H$, if we can compute the homomorphism numbers $\mathrm{Hom}(F,G)$ for all $F \in \mathrm{Spasm}(H)$ in time $O(n^c)$ on $n$-vertex graphs $G$, then we can compute $\mathrm{Sub}(H,G)$ in time $O(n^c)$. With the basic treewidth-based algorithm from Proposition 1.6, this running time $O(n^c)$ is governed by the maximum treewidth among $\mathrm{Spasm}(H)$.

**Theorem 1.1, restated.** *There is an algorithm that is given a $k$-edge graph $H$ and an $n$-vertex graph $G$ to compute $\#\mathrm{Sub}(H \to G)$ in time $k^{O(k)} n^{t+1}$, where $t$ is the maximum treewidth among all graphs in $\mathrm{Spasm}(H)$.*

*Proof.* Use (19) and evaluate the right-hand side in the straightforward manner: First compute the spasm of $H$ and all coefficients $\mathrm{Surj}^{-1}(H,F)$ for $F \in \mathrm{Spasm}(H)$ by first computing $\mathrm{Surj}_{\mathrm{Spasm}(H)}$ via brute-force and then inverting this triangular matrix. Then, for each $F \in \mathrm{Spasm}(H)$, compute $\#\mathrm{Hom}(F \to G)$ via Proposition 1.6. Finally, we compute the sum on the right side of (19) to obtain $\#\mathrm{Sub}(H \to G)$.

For the running time claim, note that the size of $\mathrm{Spasm}(H)$ is bounded by the number of partitions of the set $V(H)$, which in turn can be bounded crudely by $k^{O(k)}$. Each term $\#\mathrm{Hom}(F \to G)$ can be computed in time $\exp(O(k)) \cdot n^{t+1}$ by Proposition 1.6. □

Theorem 1.1 is particularly useful for sparse patterns $H$: By Fact 3.4, any $k$-edge graph $H$ only contains graphs with at most $k$ edges in its spasm. We can thus exploit known bounds on the treewidth of sparse graphs to bound the running time guaranteed by Theorem 1.1: If $F$ has $k$ edges, then $\mathrm{tw}(F) \leq ck + o(k)$ is known to hold for fairly small constants $c < 1$. Furthermore, there are $k$-edge graphs $F$ with $\mathrm{tw}(F) = \Theta(k)$. Since the best known upper and lower bounds on the treewidth of $k$-edge graphs are not tight, it will be useful to dedicate a universal constant $\xi$ to the linear coefficient in the treewidth bound.

**Definition 4.1.** *For $k \in \mathbb{N}$, let $\mathrm{tw}^*(k)$ be defined as the maximum treewidth among all graphs with $k$ edges. We define the constant $\xi \in \mathbb{R}$ as the limit superior*

$$\xi = \limsup_{k \to \infty} \frac{\mathrm{tw}^*(k)}{k}\,.$$

Using the existence of good 3-regular expanders, Dvořák and Norin [22, Corollary 7] find a family of 3-regular graphs on $k$ edges and $n = \frac{2}{3}k$ vertices with treewidth at least $\frac{1}{24}n - 1 = \frac{1}{36}k - 1$ for all large enough $k$. On the other hand, Scott and Sorkin [52, Corollary 21] prove that $\mathrm{tw}^*(k) \leq \frac{13}{75}k + o(k)$ holds. Collecting these two results, we get the following bounds on $\xi$.

**Theorem 4.2 ([22, 52]).** *We have $\frac{1}{36} \leq \xi \leq \frac{13}{75}$.*

Since $\xi \leq \frac{13}{75} < 0.174$, this immediately implies upper bounds on the running times obtained from Theorem 1.1.

**Corollary 1.2, restated.** *There is an algorithm that is given $H$ and $G$ to compute $\#\mathrm{Sub}(H \to G)$ in time $f(H) \cdot |V(G)|^{\xi k + o(k)}$, where $k = |E(H)|$. Here, $\xi < 0.174$ is the constant from Definition 4.1.*

Instead of relying on Proposition 1.6 to count homomorphisms in Theorem 1.1, we can use more sophisticated methods where available. For instance, we can use fast matrix multiplication to count homomorphisms from patterns $H$ of treewidth at most two, thus proving Theorem 1.3.

**Theorem 1.3, restated.** *If $H$ has treewidth at most 2, we can determine $\#\mathrm{Hom}(H \to G)$ in time $\mathrm{poly}(|V(H)|) \cdot |V(G)|^{\omega}$, where $\omega < 2.373$ is the matrix multiplication constant and $f$ is some computable function.*

*Proof.* Let $H$ be a graph and $(T, \beta)$ be a tree decomposition of width at most 2. We can compute an optimal tree decomposition in time $\mathrm{poly}(|V(H)|)$, for example via Bodlaender's $\exp(O(\mathrm{tw}(H)^3)) \cdot |V(H)|$ time algorithm [7]. It will simplify notation if we assume the decomposition to have the following properties, which can be achieved by easy modifications:

1. The root bag $r$ has size 2 and every other bag has size exactly 3.
2. The root bag $r$ has a unique child $r'$.
3. For every $t \in V(T) \setminus \{r\}$, we have $|\sigma(t)| = 2$.

Let us sketch how these properties can be achieved. If $\beta(t_1) \subseteq \beta(t_2)$ holds for two adjacent nodes $t_1, t_2 \in V(T)$, then the two nodes can be merged. If $|\beta(t)| < 3$ and $t$ has a neighbor $t'$ in $T$ with $\beta(t') \not\subseteq \beta(t)$, then the size of $\beta(t)$ can be increased by adding an element of $\beta(t') \setminus \beta(t)$ to it. After applying these two rules exhaustively, every bag has size exactly 3. If two adjacent bags $\beta(t_1) = \{a, b, c\}$ and $\beta(t_2) = \{c, d, e\}$ intersect only in one element $c$, then we can insert a new bag $\{b, c, d\}$ between them. Similarly, if $\beta(t_1) = \{a, b, c\}$ and $\beta(t_2) = \{d, e, f\}$ are disjoint, then we insert two new bags $\{b, c, d\}$ and $\{c, d, e\}$ between them. This way, we achieve property 3 above. Finally, we take any bag $\beta(r') = \{a, b, c\}$, attach a new bag $\beta(r) = \{a, b\}$ to it, and make $r$ the root of the tree.

We may assume that $V(H) = \{1, \ldots, |V(H)|\}$, that is, the vertices are represented by integers from 1 to $|V(H)|$. We may further assume that the numbering is consistent with the structure of the tree decomposition. More precisely, we assume that the following condition holds for all $t, t' \in V(T)$ and $u, u' \in V(H)$ with $u \in \beta(t)$ and $u' \in \beta(t')$: If $t$ is an ancestor of $t'$ in the tree $T$, the vertex $u$ does not occur in the bag of any ancestor of $t$, and $u'$ does not occur in the bag of any ancestor of $t$, then the numbering reflects this fact in that $u < u'$ holds.

For any $t \in V(H) \setminus \{r\}$, we write $u_1(t), u_2(t), u_3(t) \in \beta(t)$ for the three elements of the bag at $t$, chosen in such a way that $u_1(t) < u_2(t) < u_3(t)$ holds. By the assumption on the numbering, we have $\sigma(t) = \{u_1(t), u_2(t)\}$ since the two vertices in $\sigma(t)$ are have their topmost bag above $t$ while the topmost bag of the vertex in $\beta(t) \setminus \sigma(t)$ is at $t$.

For any $t \in V(H) \setminus \{r\}$ and $v_1, v_2 \in V(G)$, let $h_t(v_1, v_2)$ be the number of homomorphisms from $\mathrm{Hom}(H[\gamma(t)] \to G)$ that map $u_1(t)$ to $v_1$ and $u_2(t)$ to $v_2$. By summing over all $v_1$ and $v_2$, we obtain the number of all homomorphisms from the cone at $t$, that is, we have

$$\#\mathrm{Hom}(H[\gamma(t)] \to G) = \sum_{v_1, v_2 \in V(G)} h_t(v_1, v_2)\,.$$

As $H[\gamma(r')] = H$ holds, the function $h_{r'}$ at the root $r'$ of the tree decomposition can be thus used to determine the quantity $\#\mathrm{Hom}(H \to G)$ that we wish to compute. With random access to the function table of $h_{r'}$, we can evaluate the sum in $O(|V(G)|^2)$ arithmetic operations.

We compute the function tables $h_t$ in a bottom-up fashion. At the leaves $t$ of $T$, we compute $h_t$ in constant time using brute force. Now suppose that $t$ is a non-leaf vertex of $T$ and that the function table of $h_{t'}$ has already been computed for every child $t'$ of $t$. For each $ij \in \{12, 13, 23\}$, let $C_{ij}$ be the set of children $t'$ of $t$ that satisfy $\sigma(t') = \{u_i(t), u_j(t)\}$. Note that $C_{12} \,\dot\cup\, C_{13} \,\dot\cup\, C_{23}$ is the set of all children of $t$. The following identity holds:

$$h_t(v_1, v_2) = \sum_{v_3 \in V(G)} I_{t,v_1,v_2,v_3} \cdot \prod_{t' \in C_{12}} h_{t'}(v_1, v_2) \cdot \prod_{t' \in C_{13}} h_{t'}(v_1, v_3) \cdot \prod_{t' \in C_{23}} h_{t'}(v_2, v_3)\,, \tag{20}$$

where $I_{t,v_1,v_2,v_3} \in \{0, 1\}$ indicates whether the mapping $\beta(t) \to \{v_1, v_2, v_3\}$ that we want to fix here is a homomorphism in $\mathrm{Hom}(H[\beta(t)] \to G)$. The identity in (20) should be read as follows: In order to count the number of homomorphisms from $\gamma(t)$ that fix $u_1(t)$ to $v_1$ and $u_2(t)$ to $v_2$, we first sum over all possible values $v_3$ that $u_3(t)$ might map to and count those homomorphisms from $\gamma(t)$ that have all three values and thus all function values of $\beta(t)$ fixed to $v_1$, $v_2$, and $v_3$, respectively. In order to count the latter, we discard with the factor $I_{t,v_1,v_2,v_3}$ those fixings that violate the homomorphism property locally. Finally, we observe that $\gamma(t'_1) \cap \gamma(t'_2) = \sigma(t'_1) = \{u_i(t), u_j(t)\}$ holds for any two distinct children $t'_1, t'_2 \in C_{ij}$. Hence, after fixing the values for the homomorphism on $\beta(t)$, the extensions to the cones $\gamma(t')$ are independent for different children $t'$, and so the total number of extensions is the product of the $h_{t'}$.

Using the identity (20) naively to perform the computation at each bag would result in an overall running time of roughly $|V(G)|^3$. Instead, we want to use matrix multiplication. We define three functions $a_{12}, a_{13}, a_{23} : V(G) \times V(G) \to \mathbb{N}$ which we later interpret as matrices whose indices range over $V(G)$. For each $v, v' \in V(G)$, let

$$a_{ij}(v, v') = \big[u_i u_j \in E(H) \text{ implies } vv' \in E(G)\big] \cdot \prod_{t' \in C_{ij}} h_{t'}(v, v')\,, \tag{21}$$

where $[P] \in \{0, 1\}$ for a proposition $P$ is equal to 1 if and only if $P$ is true. Then (20) implies

$$h_t(v_1, v_2) = a_{12}(v_1, v_2) \cdot \sum_{v_3 \in V(G)} \Big(a_{13}(v_1, v_3) \cdot a_{23}(v_2, v_3)\Big) \tag{22}$$

Let $A_{13}$ be a $|V(G)| \times |V(G)|$ matrix where the rows and columns are indexed by $|V(G)|$ and the value in row $v_1$ and column $v_3$ is $a_{13}(v_1, v_3)$. Similarly, let $A_{23}$ be a $|V(G)| \times |V(G)|$ matrix where the rows and columns are indexed by $|V(G)|$ and the value in row $v_3$ and column $v_2$ is $a_{23}(v_2, v_3)$. The sum in (22) is exactly the value of the matrix product $A_{13}A_{23}$ in row $v_1$ and column $v_2$. Thus $h_t(v_1, v_2)$ can be computed by constructing the matrices $A_{13}$ and $A_{23}$, performing a matrix multiplication, and multiplying the entries with the values $a_{12}(v_1, v_2)$. The running time is dominated by the matrix multiplication, which yields a total running time of $f(H) \cdot |V(G)|^\omega$. Note that the entries of the matrices are nonnegative integers not greater than $|V(G)|^{|V(H)|}$, hence the arithmetic operations are on $O(|V(H)| \log |V(G)|)$ bit integers; arithmetic operations on such integers are in time $f(|V(H)|)$ in the standard word-RAM model with $\log|V(G)|$-size words. □

Theorem 1.3, our algorithm for counting subgraphs whose spasm has maximum treewidth 2, follows by replacing the homomorphism counting subroutine in the proof of Theorem 1.2 with the one above.

# 5 Complexity of linear combination problems

## 5.1 Linear combinations of subgraphs

We prove the dichotomy stated in Theorems 1.4 and 1.5. Recall that due to the basis transformation (19), we can express linear combinations of subgraph numbers as equivalent linear combinations of homomorphism numbers. By Lemma 3.8, the most difficult homomorphism number in this linear combination governs the complexity of the problem. Since the hardness criterion for homomorphisms is treewidth, and expressing subgraph numbers in the homomorphism basis yields terms for all graphs in the spasm, we first make the following observation.

**Fact 5.1.** *Let $H$ be a graph. If $H$ has a maximum matching of size $k$, the maximum treewidth among all graphs in* $\mathrm{Spasm}(H)$ *is* $\Theta(k)$.

*Proof.* Let $k$ be the size of a maximum matching of $H$. Then the vertex-cover number of $H$ is at least $k$ and at most $2k$. By the second item of Fact 3.4, the vertex-cover number and thus the treewidth of graphs in the spasm of $H$ is then also at most $2k$. For the lower bound, we use the third item of Claim 3.4: Since $H$ contains a matching of size $k$, every $k$-edge graph occurs as a minor of some graph in $\mathrm{Spasm}(H)$. By Theorem 4.2, there exist $k$-edge graphs $F$ with treewidth $\mathrm{tw}^*(k) \geq \Omega(k)$. Moreover, taking minors does not increase the treewidth, so there is a graph in $\mathrm{Spasm}(H)$ with treewidth at least $\Omega(k)$. □

Let $\mathcal{A} \subseteq \mathbb{Q}^{\mathcal{G}^*}$ be a family of finitely supported vectors. Recall the matrices Sub and Surj from Section 3.1. We define the set $\mathcal{A} \cdot \mathrm{Surj}^{-1}$ as the set of all vectors $\alpha \cdot \mathrm{Surj}^{-1}$ for $\alpha \in \mathcal{A}$. That is, for each $\alpha \in \mathcal{A}$, the graph motif parameter $\alpha \cdot \mathrm{Sub}$ appears in the set $\mathcal{A} \cdot \mathrm{Surj}^{-1}$ via its representation in the homomorphism basis.

**Theorem 5.2.** *Let $\mathcal{A} \subseteq \mathbb{Q}^{\mathcal{G}^*}$ be a recursively enumerable family of finitely supported vectors. If there is a constant $t \in \mathbb{N}$ such that all vectors $\beta \in \mathcal{A} \cdot \mathrm{Surj}^{-1}$ have treewidth $\mathrm{tw}(\beta) \leq t$, then the problem $\#\mathsf{Sub}(\mathcal{A})$ to compute the quantity*

$$\sum_{H \in \mathcal{G}^*} \alpha_H \cdot \#\mathrm{Sub}(H \to G)\,,$$

*on input $\alpha \in \mathcal{A}$ and an $n$-vertex graph $G$, admits an algorithm with running time $g(\alpha) \cdot n^{t+1}$. Otherwise, it is $\#\mathsf{W}[1]$-hard parameterized by the description length of $\alpha$, and it cannot be computed in time $g(\alpha) \cdot n^{o(t/\log t)}$ for $t = \mathrm{tw}(\alpha \cdot \mathrm{Surj}^{-1})$ unless $\#\mathsf{ETH}$ fails.*

*Proof.* Recall that $\mathrm{Sub} = \mathrm{Surj}^{-1}\,\mathrm{Hom}$ holds by (11), so we have $\alpha\,\mathrm{Sub} = \alpha\,\mathrm{Surj}^{-1}\,\mathrm{Hom}$ for all $\alpha \in \mathcal{A}$. This implies the algorithmic claim via Lemma 3.5 and the hardness claims via Lemma 3.8. □

Since any family of subgraph patterns $\mathcal{H}$ can be represented as a family $\mathcal{A}$ of linear combinations in which each vector has support one, we obtain Theorem 1.4 and Theorem 1.5 as a special case. The quantitative lower bound regarding the vertex-cover number follows from the relationship between the vertex-cover number and the largest treewidth in the spasm via Fact 5.1.

*Proof of Theorems 1.4 and 1.5.* Let $\mathcal{H}$ be a graph family of unbounded vertex-cover number. Its closure $\bigcup_{H \in \mathcal{H}} \mathrm{Spasm}(H)$ has unbounded treewidth by Fact 5.1. We want to apply Theorem 5.2. Let $\mathcal{A}$ be the set of all $\alpha^H$ where $\alpha^H_F = [H = F]$ holds for some graph $H \in \mathcal{H}$. That is, $\alpha^H$ contains $H$ with coefficient 1, and no other graphs. Clearly $\#\mathsf{Sub}(\mathcal{A})$ is equivalent to $\#\mathsf{Sub}(\mathcal{H})$.

Now consider the set $\mathcal{B}$ with $\mathcal{B} = \mathcal{A} \cdot \mathrm{Surj}^{-1}$. We need to prove that the graph class $\bigcup_{\beta \in \mathcal{B}} \mathrm{supp}(\beta)$ has unbounded treewidth. We do so by showing that this class is in fact equal to $\bigcup_{H \in \mathcal{H}} \mathrm{Spasm}(H)$. By definition, for each $H \in \mathcal{H}$, the set $\mathcal{B}$ contains a vector $\beta^H$ with $\beta^H = \alpha^H \cdot \mathrm{Surj}^{-1}$. Expanding this vector-matrix product, we get

$$\beta^H_F = \sum_{J \in \mathcal{G}^*} \alpha^H_J \cdot \mathrm{Surj}^{-1}(J, F)\,.$$

Since $\alpha^H_J = [H = J]$, we have $\beta^H_F = \mathrm{Surj}^{-1}(H, F)$. By (14), the latter is non-zero if and only if $F \in \mathrm{Spasm}(H)$, so the claim follows. □

We present an example that does not directly follow from [17], but that does follow from Theorem 5.2. The following statement was proved recently using a more complicated method.

**Corollary 5.3 ([9]).** *Given $k$ and $G$, counting all trees with $k$ vertices in $G$ is $\#\mathsf{W}[1]$-hard on parameter $k$.*

*Proof.* The number of $k$-vertex trees can be seen as a linear combination of subgraph numbers; for each fixed $k$, we set $\alpha_F = 1$ for all unlabeled graphs $F \in \mathcal{G}^*$ such that $F$ is a $k$-vertex tree, and $\alpha_F = 0$ otherwise. Let $\mathcal{A}$ be the family of all such $\alpha$ over all $k \in \mathbb{N}$. Since the class of all trees has unbounded vertex cover number, Fact 5.1 shows that the union of spasms of $k$-vertex trees has unbounded treewidth as $k$ grows.

Write $\mathcal{T}_k$ for the set of all $k$-vertex trees. For each $k \in \mathbb{N}$, pick a graph $F_k \in \mathrm{Spasm}(\mathcal{T}_k)$ such that the sequence $F_1, F_2, \ldots$ has unbounded treewidth. In order to apply Theorem 5.2, we need to prove that some vector $\beta \in \mathcal{A} \cdot \mathrm{Surj}^{-1}$ indeed has $F_k$ in its support: To this end, let $\alpha \in \mathcal{A}$ be the vector corresponding to all $k$-vertex trees and let $\beta = \alpha \cdot \mathrm{Surj}^{-1}$. We claim that $F_k$ is contained in the support of $\beta$. To see this, we expand the matrix-vector product:

$$\beta_{F_k} = (\alpha \cdot \mathrm{Surj}^{-1})_{F_k} = \sum_{T \in \mathcal{T}_k} \mathrm{Surj}^{-1}(T, F_k)\,. \tag{23}$$

Recall from (14) that $\mathrm{Surj}^{-1}(T, F_k) \neq 0$ if and only if $F_k \in \mathrm{Spasm}(T)$. The same equation implies that the sign of $\mathrm{Surj}^{-1}(T, F_k)$ is equal to $(-1)^{|V(T)|-|V(F_k)|}$ for all $T \in \mathcal{T}_k$ with $F_k \in \mathrm{Spasm}(T)$. Since $|V(T)| = k$ holds for all $T \in \mathcal{T}_k$, the sign is in fact $(-1)^{k-|V(F_k)|}$. Therefore, all terms in the sum of (23) have the same sign and at least one term is non-zero; thus $\beta_{F_k} \neq 0$ holds as claimed. □

The preceding corollary also holds for counting $k$-vertex forests, and can be generalized to families $\mathcal{A}$ where each linear combination $\alpha$ contains only graphs with the same number of vertices, and where $\mathcal{A}$ contains graphs of unbounded vertex-cover number in their support. In fact, the graphs in $\mathrm{supp}(\alpha)$ do not even need to have the same number of vertices, but the same *parity* of number of vertices suffices. The proof is analogous to that of Corollary 5.3.

## 5.2 Linear combinations of induced subgraphs

Chen, Thurley, and Weyer [13] consider the restriction of computing $\#\mathrm{StrEmb}(H \to G)$, parameterized by $k = |V(H)|$, when the graphs $H$ are chosen from some class $\mathcal{H}$. They prove that the problem is $\#\mathsf{W}[1]$-hard if $\mathcal{H}$ is infinite, otherwise it is polynomial-time computable. Recall that counting strong embeddings is equivalent to counting induced subgraphs. In full analogy to Theorem 5.2, we can generalize their result to a classification of linear combinations of induced subgraph numbers. For the following statement, recall the matrices Ext and Surj from Section 3.1.

**Theorem 1.13, restated.** *Let $\mathcal{A} \subseteq \mathbb{Q}^{\mathcal{G}^*}$ be a recursively enumerable family of finitely supported vectors. If there is a constant $t \in \mathbb{N}$ such that all vectors $\beta \in \mathcal{A} \cdot \mathrm{Ext}^{-1} \cdot \mathrm{Surj}^{-1}$ have treewidth $\mathrm{tw}(\beta) \leq t$, then the problem $\#\mathsf{Ind}(\mathcal{A})$ to compute*

$$\sum_{H \in \mathcal{G}^*} \alpha_H \cdot \#\mathrm{IndSub}(H \to G)\,,$$

*on input $\alpha \in \mathcal{A}$ and an $n$-vertex graph $G$, admits an algorithm with running time $g(\alpha) \cdot n^{t+1}$. Otherwise, it is $\#\mathsf{W}[1]$-hard parameterized by the description length of $\alpha$, and it cannot be solved in time $g(\alpha) \cdot n^{o(t/\log t)}$ for $t = \mathrm{tw}(\alpha \cdot \mathrm{Ext}^{-1} \cdot \mathrm{Surj}^{-1})$ unless $\#\mathsf{ETH}$ fails.*

The proof is analogous to that of Theorem 5.2, only the basis change matrix needs to be chosen as $\mathrm{Ext}^{-1} \cdot \mathrm{Surj}^{-1}$ rather than $\mathrm{Surj}^{-1}$.

On a related note, Jerrum and Meeks [33, 34, 35, 45] introduced the following generalization of counting induced subgraphs: Let $\Phi$ be some graph property. The problem $\#\mathsf{IndProp}(\Phi)$ is, given a graph $G$ and an integer $k$, to compute the number of induced $k$-vertex subgraphs that have property $\Phi$. Let us write this number as $I_{\Phi,k}(G)$. Since $I_{\Phi,k}(G)$ can be expressed as a sum $\sum_H \mathrm{IndSub}(H, G)$ over all $k$-vertex graphs $H$ that satisfy $\Phi$, Theorem 1.13 immediately implies a complexity dichotomy for these problems.

**Corollary 1.11, restated.** *Let $\Phi$ be any decidable graph property. The problem $\#\mathsf{IndProp}(\Phi)$ is fixed-parameter tractable if all $I_{\Phi,k}$ can be represented as linear combinations of homomorphisms from graphs of bounded treewidth. Otherwise, the problem is $\#\mathsf{W}[1]$-hard when parameterized by $k$.*

Finally, let us sketch how to recover the hardness result of Chen, Thurley, and Weyer [13] for counting induced subgraphs from a fixed class $\mathcal{H}$ as a special case of Corollary 1.11: When representing $\#\mathrm{IndSub}(H \to G)$ for a $k$-vertex graph $H$ as a linear combination of subgraph numbers $\#\mathrm{Sub}(H' \to G)$ via (7), this linear combination has a non-zero coefficient $\mathrm{Ext}^{-1}(H, H')$ for the clique $H' = K_k$. Indeed, the $k$-clique extends every graph on $k$ vertices, so we have $\mathrm{Ext}(H, H') \neq 0$, which in turn implies $\mathrm{Ext}^{-1}(H, H') \neq 0$ by (9). When further writing each term $\#\mathrm{Sub}(H' \to G)$ as a linear combination of homomorphisms $\#\mathrm{Hom}(H'' \to G)$ for $H'' \in \mathrm{Spasm}(H')$, we get exactly one term for $H'' = H' = K_k$, since $K_k$ only occurs in its own spasm, and we have $\mathrm{Surj}^{-1}(H', H'') = 1/\#\mathrm{Aut}(H')$. Hence the coefficient of $\#\mathrm{Hom}(K_k \to G)$ in the representation of $\#\mathrm{IndSub}(H \to G)$ is non-zero. Thus, if $\mathcal{H}$ is infinite, we have unbounded cliques in the homomorphism representation, and the problem of counting induced subgraphs from $\mathcal{H}$ is $\#\mathsf{W}[1]$-hard by Corollary 1.11.

# 6 Counting colored subgraphs in polynomial time

In this section, our goal is to determine which classes $\mathcal{H}$ of vertex-colored graphs make $\#\mathsf{Sub}(\mathcal{H})$ polynomial-time solvable. From Theorem 1.15, we know that if the graphs in $\mathrm{Spasm}(H)$ have treewidth bounded by some constant $c$ for every $H \in \mathcal{H}$, then $\#\mathsf{Sub}(\mathcal{H})$ is FPT and it is $\#\mathsf{W}[1]$-hard otherwise. However, this does not tell us which of the FPT cases are actually polynomial-time solvable. Answering this question seems to require different techniques than what we have seen in the previous sections: in particular, the hardness proofs of Section 5.1 based on the basis transformation to homomorphisms gives reductions with superpolynomial running time and hence cannot be used to rule out polynomial-time algorithms in the cases when the problem is FPT.

To understand the limits of polynomial-time algorithms, we give two different structural characterizations that are equivalent to the condition that $\mathrm{Spasm}(H)$ has bounded treewidth for a colored graph $H$. We need the following definitions:

- $\widehat{H}$ is the simple graph obtained from $H$ by consolidating each color class into a single vertex. That is, vertex $v_i$ of $\widehat{H}$ represents color class $i$ of $H$, and $v_i$ and $v_j$ of $\widehat{H}$ are adjacent if there is an edge between color class $i$ and color class $j$ in $H$.
- $H^\bullet$ is the supergraph of $H$ obtained by making each color class a clique.
- $H^{\bullet\setminus i}$ is the supergraph of $H$ obtained by making each color class *except class $i$* a clique.
- A *$c$-flower centered at color class $i$ in $H$* is a vertex-disjoint collection of $c$ paths of length at least 1 in $H^{\bullet\setminus i}$ such that each path has both of its endpoints in class $i$. (Note that it is possible that path in the collection has length 1, that is, consists of only a single edge inside color class $i$.)

We show that $\mathrm{Spasm}(H)$ having bounded treewidth can be expressed by saying that $\widehat{H}$ has bounded treewidth and there are no large flowers centered at any color class. For the second characterization, we need a novel form of tree decompositions specifically tailored for our colored counting applications.

**Definition 6.1.** *Let $H$ be a colored graph, let $(T, \beta)$ be a tree decomposition of $H^\bullet$, and let $g : V(T) \to 2^{V(H)}$ be a function with $\sigma(t) \subseteq g(t) \subseteq \beta(t)$ for every $t \in V(T)$. The triple $(T, \beta, g)$ is a* guarded cutvertex decomposition *if the following two properties hold:*

1. *for every $t \in V(T)$, set $g(t)$ is a vertex cover of $H[\beta(t)]$.*
2. *for every $t \in V(T)$ and every child $t'$ of $t$, we have $|\sigma(t') \setminus g(t)| \leq 1$.*

*The* guard size *of $(T, \beta, g)$ is defined as $\max_{t \in V(T)} |g(t)|$. We say that the* bags have at most $c$ colors *if $\beta(t)$ for each $v \in V(T)$ contains vertices of at most $c$ different colors.*

Note that this is a decomposition of $H^\bullet$ (where each color class is a clique), but guard set $g(t)$ is a vertex cover of $H[\beta(t)]$ only (where the color classes can be sparse). It turns out the existence of precisely this kind of decompositions with guard sets of bounded size is a sufficient and necessary condition for $\mathrm{Spasm}(H)$ to have bounded treewidth.

**Theorem 6.2.** *Let $\mathcal{H}$ be a class of colored graphs. The following statements are equivalent:*

1. *There is a constant $c$ such that for every $H \in \mathcal{H}$, we have that every graph in $\mathrm{Spasm}(H)$ has treewidth at most $c$.*
2. *There is a constant $c$ such that for every $H \in \mathcal{H}$, we have that $\widehat{H}$ has treewidth at most $c$ and there is no $c$-flower centered at any class of $H$.*
3. *There is a constant $c$ such that for every $H \in \mathcal{H}$, there is a guarded cutvertex decomposition of $H$ with guard size at most $c$ and the bags having at most $c$ colors.*

As we are interested in polynomial-time algorithms, the existence statements in Theorem 6.2 are not sufficient for us: we need an algorithmic statement showing that guarded cutvertex decompositions can be found efficiently. The following theorem gives an algorithmic version of the (2) $\Rightarrow$ (3) implication of Theorem 6.2.

**Theorem 6.3.** *Let $H$ be a colored graph such that*

- *$\widehat{H}$ has treewidth at most $w$, and*
- *$H$ has no $c$-flower centered at any of the color classes.*

*Then we can compute a guarded cutvertex decomposition $(T, \beta, g)$ of $H$ with guard size $O((w+c)^2 w^2)$ such that the bags have at most $O((c+w)w)$ colors.*

We present a dynamic programming algorithm solving $\mathrm{Sub}(H \to G)$, given a guarded cutvertex decomposition of $H$ with bounded guard size. This can be used to give another proof that $\#\mathsf{Sub}(H)$ is FPT when $\mathrm{Spasm}(H)$ has bounded treewidth for every $H \in \mathcal{H}$ (but we have already established that in Theorem 1.15). More importantly, this dynamic programming algorithm can be used to solve all the polynomial-time solvable cases of $\#\mathsf{Sub}(H)$. The running time of the algorithm has a factor that has exponential dependence on the number of certain types of vertices in the guarded decomposition. If the number of these vertices is at most logarithmic in the size of $H$, then this exponential factor is polynomially bounded in the size of $H$ and the running time is polynomial.

We need the following definition for the formal statement of the running time of the algorithm. In a guarded cutvertex decomposition of $H$, the vertices of $\beta(t) \setminus g(t)$ are of two types: either there is some child $t'$ of $t$ such that $v$ is adjacent to $\alpha(t')$ or not. Moreover, the requirement $|\sigma(t') \setminus g(t)| \leq 1$ ensures that at most one vertex of $\beta(t) \setminus g(t)$ can be adjacent to $\alpha(t')$. Formally, for every $t \in V(T)$ and $v \in \beta(t) \setminus g(t)$, we define a set $h_t(v) \subseteq V(T)$ the following way: a child $t'$ of $t$ is in $h_t(v)$ if $v$ is adjacent to $\alpha(t')$. That is, $h_t(v)$ contains those children of $t$ that are "hanging" on the vertex $v$ (and on some of the guards $g(t)$). Let $\lambda(t) \subseteq \beta(t) \setminus g(t)$ contain those vertices $v$ for which $h_t(v) \neq \emptyset$.

**Lemma 6.4.** *Let $H$ be a colored graph and $(T, \beta, g)$ be a tight guarded cutvertex decomposition of $H$ with guard size at most $c$ and at most $c$ colors in each bag. Suppose that $|\lambda(t)| \leq d$ for every $t \in V(T)$. Then $\mathrm{Emb}(H \to G)$ can be computed in time $2^{O(d)} \cdot |V(G)|^{2^{O(c)}}$.*

Finally, we need to show that if the algorithm of Lemma 6.4 cannot be used to give a polynomial-time algorithm for $\#\mathsf{Sub}(\mathcal{H})$, then the problem is unlikely to be polynomial-time solvable. We observe that $\lambda(t)$ being large at some node $t$ implies the existence of a large half-colorful matching and then such matchings can be used in a reduction from counting $k$-matchings to $\#\mathsf{Sub}(\mathcal{H})$. If half-colorful matchings of super-logarithmic size appear in the graphs of $\mathcal{H}$, then this reduction can be used to count perfect matchings in $2^{o(n)}$ time.

## 6.1 Obtaining a tree decomposition

The goal of this subsection is to prove Theorems 6.2 and 6.3. After stating useful facts about guarded cutvertex decompositions, we discuss two technical tools: we formally state a minor but tedious transformation of the tree decomposition and recall some known results about finding $A$-paths. Then we prove Theorem 6.3, which shows the (2) $\Rightarrow$ (3) implication of Theorem 6.2. Then we complete Theorem 6.2 with the implications (3) $\Rightarrow$ (1) and (1) $\Rightarrow$ (2).

### 6.1.1 Useful facts

We make a few observations on the structure of guarded decompositions.

**Lemma 6.5.** *Let $H$ be a colored graph and let $(T, \beta, g)$ be a guarded cutvertex decomposition of $H$. If $v$ is a vertex in $\beta(t) \setminus (g(t) \cup \lambda(t))$ for some $t \in V(T)$, then*

1. *$v$ does not appear in $\beta(t')$ for any $t' \in V(T)$ with $t \neq t'$.*
2. *$\beta(t)$ contains every vertex with the same color as $v$.*
3. *$N_H(v) \subseteq g(t)$.*

*Proof.* Suppose $v$ appears in $\beta(t')$ for some $t' \neq t$. If $t'$ is a descendant of $t$, then $v$ has to appear in $\beta(t'')$ for some child $t''$ of $t$, which means that $v$ is in $\sigma(t'')$ and hence $v \in \lambda(t)$ follows, a contradiction. If $t'$ is not a descendant of $t$, then $v$ has to appear in $\sigma(t) \subseteq g(t)$, again a contradiction. Thus $v$ cannot appear in $\beta(t')$ for any $t' \neq t$, proving statement 1. Furthermore, this implies that every neighbor of $v$ in $H^\bullet$ has to appear in $\beta(t)$. As every vertex with the same color as $v$ is a neighbor of $v$ in $H^\bullet$, statement 2 follows. Since $H$ is a subgraph of $H^\bullet$, it also follows that $N_H(v)$ is in $\beta(t)$. The fact that $g(t)$ is a vertex cover of $H[\beta(t)]$ further implies that every neighbor of $v$ is in $g(t)$, proving statement 3. □

It is not obvious that if $H$ has a guarded cutvertex decomposition with bounded guard size, then $H$ has bounded treewidth: the definition does not say anything about the size of the bags. However, it is not difficult to show that this is indeed the case, although we need to modify the decomposition to make the size of the bags bounded.

**Lemma 6.6.** *If $H$ has a guarded cutvertex decomposition with guard size at most $c$, then $H$ has treewidth at most $c$.*

*Proof.* Let $(T, \beta, g)$ be a guarded cutvertex decomposition of $H$ with guard size at most $c$. We obtain a tree decomposition $(T', \beta')$ of $H$ the following way. For every node $t \in V(T)$, we introduce $t$ into $T'$ and set $\beta'(t) = g(t)$. Then for every $v \in \beta(t) \setminus g(t)$, we introduce a child $t_v$ of $t$ into $T'$ and set $\beta'(t_v) = g(t) \cup \{v\}$. If a child $w$ of $t$ in $T$ has $\sigma(w) \setminus g(t) = \{v\}$ (by definition of the guarded cutvertex decomposition, there can be at most one vertex in this set), then we attach $w$ to be a child of $t_v$ in $T'$, instead of being a child of $t$. It is not difficult to verify that the new decomposition satisfies the properties of a tree decomposition. □

The following lemma gives a quick sanity check: a guarded cutvertex decomposition rules out the possibility of a large matching in any color class or between any two color class. This is expected, as such matchings would make the counting problem hard.

**Lemma 6.7.** *If $H$ has a guarded cutvertex decomposition $(T, \beta, g)$ with guard size at most $c$, then $H$ has no matching $x_1y_1, \ldots, x_{c+1}y_{c+1}$ where every $x_i$ is in class $C_x$ and every $y_i$ is in class $C_y$ (possibly $C_x = C_y$).*

*Proof.* As each color class is a clique in $H^\bullet$, there is a node $t_x$ (resp., $t_y$) with $C_x \subseteq \beta(t_x)$ (resp, $C_y \subseteq \beta(t_y)$). If $C_x = C_y$, then $x_1y_1, \ldots, x_{c+1}y_{c+1}$ is a matching in $H[\beta(t_x)]$, contradicting the assumption that $g(t)$ is a vertex cover of $H[\beta(t_x)]$. If $C_x \neq C_y$, then assume without loss of generality that $t_x$ is not an ancestor of $t_y$. Then $\sigma(t_x)$ separates $\beta(t_x) \setminus \sigma(t_x)$ and $\beta(t_y) \setminus \sigma(t_x)$. As $|\sigma(t_x)| \leq c$, there is an edge $x_iy_i$ such that $x_i, y_i \notin \sigma(t_x)$. However, this means that there can be no common bag where both $x_i$ and $y_i$ can appear. □

### 6.1.2 Massaging a tree decomposition

Let $(T, \beta)$ be a tree decomposition of a graph $G$. Let $t' \in V(T)$ be a child of $t \in V(T)$. There are certain easy modifications that allow us to make the tree decomposition nicer and more useful for dynamic programming.

**Definition 6.8.** *We say that tree decomposition $(T, \beta)$ of a graph $H$ is* tight *if for every $t \in V(T)$ and $v \in \sigma(t)$, vertex $v$ has at least one neighbor in $\alpha(t)$.*

First, if the decomposition is not tight, then $\sigma(t)$ could contain vertices not in $N^G(\alpha(t))$, but we may safely remove any such vertex from $\beta(t)$. Second, it is not necessarily true that $G[\alpha(t')]$ is connected: it may contain two or more connected components of $G - \beta(t)$. However, it is not difficult to modify the tree decomposition in such a way to ensure that $G[\alpha(t')]$ is connected: we need to introduce restricted copies of the subtree of $T$ rooted at $t'$, one for each component of $G - \beta(t)$ contained in $\alpha(t')$. We will need this transformation in the proof of Theorem 6.15, as we can prove the bound on $\sigma(t')$ only if $G[\alpha(t')]$ is connected. While these transformations are easy to perform, in the following lemma we state them in a very formal way. The reason is that in the proof of Theorem 6.3 we need to verify how certain properties of the original decomposition survive this transformation.

**Lemma 6.9.** *Let $(T^1, \beta^1)$ be a tree decomposition of a connected graph $G$ such that $\beta^1(r^1) \neq \emptyset$ for the root $r^1$ of $T^1$. We can compute in polynomial time another tree decomposition $(T^2, \beta^2)$ of $G$ and a mapping $f\colon V(T^2) \to V(T^1)$ with the following properties:*

(P.1) *if $r^2$ is the root of $T^2$, then $f(r^2) = r^1$ is the root of $T^1$ and $\beta^2(r^2) = \beta^1(r^1) \neq \emptyset$,*

(P.2) *$\beta^2(t) \subseteq \beta^1(f(t))$ for every $t \in V(T^2)$,*

(P.3) *$(\beta^1(f(t)) \setminus \beta^2(t)) \cap \gamma^2(t) = \emptyset$ for every $t \in V(T^2)$,*

(P.4) *$G[\alpha^2(t)]$ is connected for every $t \in V(T^2)$, and*

(P.5) *$\sigma^2(t) = N^G(\alpha^2(t))$ for every $t \in V(T^2)$.*

*Proof.* We give a recursive procedure for computing the required tree decomposition $(T^2, \beta^2)$. Let $r^1 \in V(T^1)$ be the root of $T^1$. By assumption, we have $\beta^1(r^1) \neq \emptyset$. Let $V_1, \ldots, V_t$ be the vertex sets of components of $G - \beta^1(r^1)$ and let $G_i = G[V_i \cup N^G(V_i)]$. If $t = 0$, then it is easy to see that the tree decomposition consisting only of the root bag $\beta^1(r_1)$ satisfies the requirements. Otherwise, for $1 \leq i \leq t$, the root $r^1$ has a child $r^1_i$ in $T^1$ such that the vertices of $V_i$ appear only in bags of the subtree $T^1_i$ of $T$ rooted at $r^1_i$ (note that $r^1_i = r^1_j$ is possible for $i \neq j$). This gives a tree decomposition $(T^1_i, \beta^1_i)$ of $G_i$, where we define $\beta^1_i(t) = \beta^1(t) \cap (V_i \cup N^G(V_i))$ for every $t \in V(T_i)$. Note that $\beta^1_i(r^1_i) \neq \emptyset$: if $\beta^1(r^1_i)$ contains no vertex of $V_i \cup N^G(V_i)$, then this would imply that $V_i \neq \emptyset$ and $\beta^1(r^1) \neq \emptyset$ are not in the same component of $G$, that is, $G$ is not connected. Let us recursively call the procedure on $(T^1_i, \beta^1_i)$ and let $(T^2_i, \beta^2_i)$ be the resulting tree decomposition of $G_i$ with root $r^2_i \in V(T^2_i)$ and let $f_i : V(T^2_i) \to V(T^1_i)$ be the corresponding function.

We construct tree decomposition $(T^2, \beta^2)$ the following way. The tree $T^2$ is constructed by taking a disjoint copy of every $T^2_i$, introducing a new root $r^2$, and connecting $r^2$ with every $r^2_i$. The function $\beta^2$ is defined the obvious way: we let $\beta^2(r^2) = \beta(r^1)$ and $\beta^2(t) = \beta^2_i(t)$ for every $t \in V(T^2_i)$. To verify that $(T^2, \beta^2)$ is a tree decomposition, we need to check for every vertex $v$ that the bags containing $v$ form a connected subtree of $T^2$. If $v \notin \beta^1(r^1)$, then $v$ appears only in the bags of $T^1_i$ for some $i$ and then the statement follows from the fact that $(T^2_i, \beta^2_i)$ is a tree decomposition. If $v \in \beta^1(r^1)$, then $v$ may appear in the bags of $T^1_i$ for more than one $i$. However, for each such $i$, we have that $v \in \beta^1_i(r^1_i)$ holds, and then the fact that $(T^2_i, \beta^2_i)$ and $f_i$ satisfy property (P.1) implies that $v$ appears in $\beta^2_i(r^2_i) = \beta^1_i(r^1_i)$.

Let us define $f(r^2) = r^1$ and let $f(t) = f_i(t)$ if $t \in V(T^2_i)$. We claim that tree decomposition $(T^2, \beta^2)$ and the function $f$ satisfy the requirements of the lemma. For the root $r^2$ of $T^2$, properties (P.1)– (P.3) hold by construction, as we have $\beta^2(r^2) = \beta(f(r^2)) = \beta(r^1) \neq \emptyset$. Property (P.4) follows from the assumption that $G$ is connected. We have $\sigma^2(r^2) = \emptyset$, satisfying property (P.5).

For a node $t \in V(T^2_i)$, we verify properties (P.2)–(P.5) as follows.

(P.2) $\beta^2(t) = \beta^2_i(t) \subseteq \beta^1_i(f_i(t)) \subseteq \beta^1(f_i(t)) = \beta^1(f(t))$, where the first subset relation uses the assumption that $(T^2_i, \beta^2_i)$ satisfies property (P.2) and the second subset relation follows from the definition $\beta^1_i$.

(P.3) We have $\beta^2(t) = \beta_i^2(t)$, $\gamma^2(t) = \gamma_i^2(t) \subseteq V_i \cup N^G(V_i)$ and $\beta_i^1(f_i(t)) = \beta_i^1(f(t)) = \beta^1(f(t)) \cap (V_i \cup N^G(V_i))$, implying $\beta_i^1(f_i(t)) \cap \gamma^2(t) = \beta^1(f(t)) \cap \gamma^2(t)$. Thus $(\beta^1(f(t)) \setminus \beta^2(t)) \cap \gamma^2(t) = (\beta_i^1(f_i(t)) \setminus \beta_i^2(t)) \cap \gamma_i^2(t) = \emptyset$, where the second equality uses that $(T_i^2, \beta_i^2)$ satisfies property (P.3).

(P.4) If $t = r_i^2$, then $\gamma^2(t) = V_i \cup N_G(V_i)$ and $\alpha^2(t) = V_i$, hence $G[\alpha^2(t)]$ is connected by the definition of $V_i$. If $t \in V(T_i^2) \setminus \{r_i^2\}$, then $\alpha^2(t) = \alpha_i^2(t)$, hence the fact that $(T_i^2, \beta_i^2)$ satisfies property (P.4) implies that $G[\alpha^2(t)] = G[\alpha_i^2(t)]$ is connected.

(P.5) If $t = r_i^2$, then $\sigma^2(t) = N_G(V_i)$. If $t \in V(T_i^2) \setminus \{r_i^2\}$, then $\sigma^2(t) = \sigma_i^2(t)$ and $\alpha^2(t) = \alpha_i^2(t)$, hence the fact that $(T_i^2, \beta_i^2)$ satisfies property (P.5) implies $\sigma^2(t) = \sigma_i^2(t) = N_{G_i}(\alpha_i^2(t)) = N_{G_i}(\alpha^2(t)) = N_G(\alpha^2(t))$.

To argue that this recursive procedure terminates, consider the measure $\zeta(T^1, \beta^1) = |V(T^1)| \cdot |V(G) \setminus \beta^1(r^1)|$, where $r^1$ is the root of $T^1$. If this measure is zero, then the recursion is terminated: $\beta^1(r^1)$ contains every vertex of $G$, hence $G - \beta^1(r^1)$ has zero components. The measure strictly decreases in each step, as the tree decomposition has strictly fewer nodes in each recursive call. Moreover, we claim that when processing tree decomposition $(T^1, \beta^1)$, the sum of the measures in the recursive calls is at most $\zeta(T^1, \beta^1)$. To see this, notice that $V(G_i) \setminus \beta^1(r_i^1)$ and $V(G_j) \setminus \beta^1(r_j^1)$ are disjoint for $i \neq j$: we have $V(G_i) \cap V(G_j) \subseteq \beta^1(r^1)$, but a vertex $v \in \beta^1(r^1)$ cannot appear in a bag of $T_i^1 \setminus \{r_i^1\}$ without appearing in $\beta^1(r_i^1)$ as well. Therefore, we can bound the number of leafs of the recursion tree by $|V(T^1)| \cdot |V(G) \setminus \beta^1(r^1)| \leq |V(T^1)| \cdot |V(G)|$. □

### 6.1.3 A-paths

If $A \subseteq V(G)$ is a set of vertices, then path $P$ is an *$A$-path* if both endpoints are in $A$ (we allow here that $P$ has other vertices in $A$, although this will not make much difference, as every $A$-path has a subpath whose internal vertices are not in $A$). The following classical result shows that either there are many vertex-disjoint $A$-paths, or they can be covered by a bounded number of vertices.

**Theorem 6.10.** *Given a graph $G$, a set $A \subseteq V(G)$ of vertices, and an integer $k$, in polynomial time we can find either*

1. *$k$ pairwise vertex-disjoint $A$-paths, or*
2. *a set $S \subseteq V(G)$ of at most $2k-2$ vertices such that $G - S$ has no $A$-path.*

If we want to cover the $A$-paths using only vertices in $A$, then we may need significantly more vertices: for example, this is the case if there is a vertex $v \notin A$ adjacent to every vertex of $A$. However, what we can show is that selecting vertices outside $A$ can be avoided unless they are highly connected to $A$ in the following sense. We say that a vertex $v$ is *$\ell$-attached* to a set $A$ of vertices if there is a set of $\ell$ paths of length at least 1 connecting $v$ and $A$ such that they share only the vertex $v$, but otherwise disjoint. Note that the definition is slightly delicate if $v \in A$: then we need $\ell$ paths connecting $v$ to $\ell$ other vertices of $A$. It can be tested in polynomial time if a vertex $v$ is $\ell$-attached using simple flow/disjoint paths algorithms.

**Theorem 6.11.** *Given a connected graph $G$, a set $A \subseteq V(G)$ of vertices, and integers $k, \ell$ with $\ell \geq 2k$, in polynomial time we can find either*

1. *$k$ pairwise vertex-disjoint $A$-paths , or*
2. *a set $A^* \subseteq A$ of at most $(2k-2)\ell$ vertices and a set $S^* \subseteq V(G)$ of at most $2k-2$ vertices such that $G - (S^* \cup A^*)$ has no $A$-path and $S^*$ is precisely the set of vertices that are $\ell$-attached to $A$.*

*Proof.* Let us first invoke the algorithm of Theorem 6.10. If it returns $k$ pairwise vertex-disjoint $A$-paths, then we are done. Otherwise, let $S$ be a set of at most $2k-2$ vertices hitting every $A$-path. We say that a vertex $v \in S$ *sees* vertex $t \in A$ if $v \neq t$ and there is a $v - t$ path that intersect $S$ only in $v$ (it is possible that $v = t$). We claim that if $v$ sees at least $\ell + 1$ vertices of $A$, then $v$ is $\ell$-attached to $A$. If $v$ sees at least $\ell + 1$ vertices of $A$, then at least $\ell$ of them are distinct from $v$. Suppose that $v$ sees vertices $t_1, \ldots, t_\ell$ of $A \setminus \{v\}$, that is, for every $1 \leq i \leq \ell$, there is a $v - t_i$ path $P_i$ of length at least 1 that intersects $S$ only in $v$. If these paths share only the vertex $v$, then $v$ is $\ell$-attached to $A$. Otherwise, suppose that $P_i$ and $P_j$ share a vertex

different from $v$. Then $(V(P_i) \cup V(P_j)) \setminus \{v\}$ contains no vertex of $S$ and induces a connected graph in $G$ connecting two distinct vertices $t_i$ and $t_j$ of $A$, contradicting the assumption that $S$ hits every $A$-path.

Let $S^* \subseteq S$ be the set of vertices that are $\ell$-attached to $A$. By the argument in the previous paragraph, every in $|S \setminus S^*|$ see at most $\ell$ vertices of $A$. Let $A^* \subseteq A$ be the set of vertices seen by $S \setminus S^*$. We clearly have $|S^*| \leq 2k-2$ and $|A^*| \leq |S \setminus S^*|\ell \leq (2k-2)\ell$.

We claim that $S^* \cup A^*$ hits every $A$-path $P$. Let $t_1, t_2 \in A$ be the endpoints of $P$; we may assume that $P$ contains no other vertex of $A$. Let $v$ be the first vertex of $P$ in $S$ when going from $t_1$ to $t_2$ (it is possible that $v = t_1$ or $v = t_2$). This means that $v$ sees $t_1$. If $v$ sees at least $\ell + 1$ vertices of $A$ (that is, at least $\ell$ vertices other than $t_1$), then, as we have seen above, $v$ is $\ell$-attached to $A$, hence $v \in S^*$ and we are done. Otherwise, $t_1$ is one of the at most $\ell$ vertices seen by $v \in S \setminus S^*$, hence $t_1 \in A^*$ and we are done again.

Finally, we need to show that $S^*$ is precisely the set of vertices $\ell$-attached to $A$. Every vertex is in $S^*$ is $\ell$-attached to $A$ by construction, hence we need to show only the converse: if $v$ is $\ell$-attached to $A$, then $v \in S^*$. Suppose that $P_1, \ldots, P_\ell$ are $v - A$ paths of length at least 1, sharing only vertex $v$. If $v \in S$, then $v$ was introduced into $S^*$. If $v \notin S$, then $\ell \geq 2k \geq |S| + 2$ implies that there are two paths $P_i$ and $P_j$ disjoint from $S$. But the these two paths show that there is an $A$-path in $G - S$, a contradiction. □

### 6.1.4 Constructing the decomposition

The following lemma contains the main technical part of the section, where we exploit the lack of $c$-flowers to constructing a certain kind of decomposition.

**Lemma 6.12.** *Let $H$ be a connected colored graph such that*

- *$\widehat{H}$ has treewidth at most $w$, and*
- *$H$ has no $c$-flower centered at any of the color classes.*

*Then we can compute in polynomial time a tree decomposition $(T^1, \beta^1)$ of $H^\bullet$ and a guard set $g^1(t)$ with $g^1(t) \subseteq \beta^1(t)$ of size $O((c+w)^2 w^2)$ for every $t \in V(T^1)$ such that the following holds:*

1. *for every $t \in V(T^1)$, $g^1(t)$ is a vertex cover of $H[\beta^1(t)]$.*
2. *for every $t \in V(T^1)$ and every component $C$ of $H^\bullet[\gamma^1(t)] - \beta^1(t)$, it holds that $|N^{H^\bullet}(C) \setminus g^1(t)| \leq 1$.*

*Furthermore, every bag $\beta^1(t)$ contains vertices from $O((c+w)w)$ colors.*

*Proof.* It will be convenient to extend the definition of $H^{\bullet \setminus i}$ the following way: for a set $S$ of colors, $H^{\bullet \setminus S}$ is the graph $H$ with every color class $C_i$ with $i \neq S$ made a clique. Consider first a tree decomposition $(\widehat{T}, \widehat{\beta})$ of $\widehat{H}$ of width $w$, that is, $|\widehat{\beta}(t)| \leq w+1$ for every $t \in V(\widehat{T})$. We can then define a tree decomposition $(T^0, \beta^0)$ of $H^\bullet$ in a natural way: let $T^0 = \widehat{T}$ and let $\beta^0(t) = \bigcup_{i \in \widehat{\beta}(t)} C_i$. Note that this decomposition has the property that every bag $\beta^0(t)$ contains vertices from at most $w+1$ color classes.

Let $k := (2c+w)(w+1)$.

**Claim 6.13.** *For every $t \in V(T^0)$, graph $H^{\bullet \setminus \widehat{\beta}(t)}$ has no $k$ pairwise vertex-disjoint $\beta^0(t)$-paths.*

*Proof.* Suppose that there is a collection of $k$ pairwise vertex-disjoint $\beta^0(t)$-paths in $H^{\bullet \setminus \widehat{\beta}(t)}$. Each endpoint of such a path is in class $C_i$ for some $i \in \widehat{\beta}(t)$ and hence there has to be an $i \in \widehat{\beta}(t)$ such that at least $2c+w$ of these paths have at least one endpoint in $C_i$. If both endpoints of a path are in $C_i$, then it is a $C_i$-path in $H^{\bullet \setminus \widehat{\beta}(t)}$, and hence also in its supergraph $H^{\bullet \setminus i}$. If there are two paths such that each of them has an endpoint in $C_i$ and an endpoint in $C_j$ for some $j \in \widehat{\beta}(t) \setminus \{i\}$, then they together form a $C_i$-path in $H^{\bullet \setminus i}$ (as $C_j$ is a clique in $H^{\bullet \setminus i}$). For every $j \in \widehat{\beta}(t) \setminus \{i\}$, if the collection contains an odd number of paths with one endpoint in $C_i$ and the other endpoint in $C_j$, then let us throw away one such path; this way we are throwing away at most $w$ paths and hence at least $2c$ paths with at least one endpoint in $C_i$ remain. The paths with both endpoints in $C_i$ are $C_i$-paths and the paths with an endpoint in $C_i$ and an endpoint in $C_j$ can be matched up to form $C_i$-paths. Therefore, the $2c$ remaining paths form at least $c$ vertex-disjoint $C_i$-paths in $H^{\bullet \setminus i}$, contradicting the assumption that there is no $c$-flower centered at class $C_i$. □

We define a new tree decomposition $(T^1, \beta^1)$ of $H^\bullet$ the following way. Let $T^1 = T^0 = \widehat{T}$. Let us use the algorithm of Theorem 6.11 on the graph $H^{\bullet\setminus\widehat{\beta}(t)}$, set $\beta^0(t)$, and integers $k = (2c+w)(w+1)$ and $\ell = 2k$. By Claim 6.13, the algorithm cannot return a set of $k$ pairwise vertex-disjoint $\beta^0(t)$-paths, hence we obtain a set $A_t^* \subseteq \beta^0(t)$ of size at most $(2k-2)\ell$ and the set $S_t^*$ of all the at most $2k$ vertices $\ell$-attached to $\beta^0(t)$ in $H^{\bullet\setminus\widehat{\beta}(t)}$. We define $\beta^1(t) = \beta^0(t) \cup (S_t^* \cap \gamma^0(t))$ and $g^1(t) = (S_t^* \cap \gamma^0(t)) \cup A_t^*$; note that we have $|g^1(t)| \leq 2k + (2k-2)\ell = O(k^2) = (c+w)^2w^2$, as required. Moreover, as we add at most $2k$ vertices to $\beta^0(t)$, it follows that $\beta^1(t)$ contains vertices from at most $w + 2k + 1 = O((c+w)w)$ colors classes.

We need to prove that $(T^1, \beta^1)$ is a tree decomposition satisfying the requirements. To show that each vertex corresponds to a connected subtree in $(T^1, \beta^1)$, we need the following claim.

**Claim 6.14.** *If $t'$ is a child of $t$ in $T^1$ and $v \in \gamma^0(t')$ is $\ell$-attached to $\beta^0(t)$ in $H^{\bullet\setminus\widehat{\beta}(t)}$, then $v \in \beta^1(t')$.*

*Proof.* If $v \in \beta^0(t')$, then $v \in \beta^1(t')$ and we are done. Let us suppose that $v \notin \beta^0(t')$. Let $P_0, \ldots, P_\ell$ be $v - \beta^0(t)$ paths in $H^{\bullet\setminus\widehat{\beta}(t)}$ that share only $v$. Each path $P_j$ contains a vertex of $\beta^0(t')$; let $u_j$ be the vertex of $P_j$ that is in $\beta^0(t')$ and closest to $v$ on $P_j$. Let $P'_j$ be the subpath of $P_j$ from $v$ to $u_j$. Note that path $P'_j$ is of length at least one: $v \notin \beta^0(t')$ by assumption, hence $v \neq u_j$. We claim that $P'_j$ is a path not only in $H^{\bullet\setminus\widehat{\beta}(t)}$, but also in $H^{\bullet\setminus\widehat{\beta}(t')}$. If an edge appears in $H^{\bullet\setminus\widehat{\beta}(t)}$, but not in $H^{\bullet\setminus\widehat{\beta}(t')}$, then it has to connect two vertices of $C_i$ for some $i \in \widehat{\beta}(t')$. However, every vertex of $C_i$ is in $\beta^0(t')$ and $P'_j$ contains a only single vertex of $\beta^0(t')$, hence it cannot use such an edge. Thus the paths $P'_1, \ldots, P'_\ell$ remain paths in $H^{\bullet\setminus\widehat{\beta}(t')}$ and they show that $v$ is $\ell$-attached to $\beta^0(t')$ in $H^{\bullet\setminus\widehat{\beta}(t')}$. This means that $v$ (which is in $\gamma^0(t')$ by assumption) appeared in the set $S_{t'}^*$ and hence it was introduced into $\beta^1(t')$. □

**Claim 6.15.** *$(T^1, \beta^1)$ is a tree decomposition of $H^\bullet$.*

*Proof.* We need to show that the subset $V_v = \{t \in V(T^1) \mid v \in \beta^1(t)\}$ induces a connected subtree of $T^1$ for every $v \in V(H)$. Suppose that $v \in \beta^1(t) \setminus \beta^0(t)$, that is, $v$ is a vertex newly introduced into $\beta^1(t)$. By the definition of $\beta^1(t)$, this is only possible if $v \in \gamma^0(t)$ and $v$ is $\ell$-attached to $\beta^0(t)$ in $H^{\bullet\setminus\widehat{\beta}(t)}$. As $v \notin \beta^0(t)$, this also means that $v \in \gamma^0(t')$ for some child $t'$ of $t$. Then by Claim 6.14, $v \in \beta^1(t')$ also holds. Thus we have shown that $v \in \beta^1(t) \setminus \beta^0(t)$ implies that $v \in \beta^1(t')$ for a child $t'$ of $t$. This implies that $V_v$ is connected. □

Now let us verify that $(T^1, \beta^1)$ satisfies the two properties required by the lemma. Let $x, y \in \beta^1(t)$ be two adjacent vertices in $H$ for some $t \in i \in \widehat{\beta}(t)$ and suppose that none of them is in $g^1(t)$. Then $x, y \in \beta^0(t)$ and $xy$ is an edge of $H^{\bullet\setminus\widehat{\beta}(t)}$, forming a $\beta^0(t)$-path of length 1 in $H^{\bullet\setminus\widehat{\beta}(t)}$. However, $g^1(t) = S_t^* \cup A_t^*$ covers every $\beta^0(t)$-path in $H^{\bullet\setminus\widehat{\beta}(t)}$, a contradiction.

For the second property, let $C$ be a connected component of $G[\gamma^1(t) \setminus \beta^1(t)]$. A $(T^1, \beta^1)$ is a tree decomposition, we have $C \subseteq \gamma^1(t')$ for some child $t'$ of $t$. Let $x, y \in N_{H^\bullet}[C] \setminus g^1(t)$ be two distinct vertices. Observe that $x, y \in \beta^1(t) \setminus g^1(t) \subseteq \beta^0(t)$ and $C$ is disjoint from $\beta^1(t) \supseteq g^1(t)$. Thus there is an $x - y$ path $P$ in $H^\bullet$ with internal vertices in $C$ (and having length at least 2). We claim that $P$ is a path also in $H^{\bullet\setminus\widehat{\beta}(t)}$. It is not possible that $P$ contains an edge $xy$ of $H^\bullet$ that does not appear in $H^{\bullet\setminus\widehat{\beta}(t)}$: if $x, y \in C_i$ for some $i \in \widehat{\beta}(t)$, then $x, y \in \beta^1(t)$ and we have that $C$ is disjoint from $\beta^1(t)$. Thus $P$ is a $\beta^0(t)$-path in $H^{\bullet\setminus\widehat{\beta}(t)}$, but every such path is covered by $g^1(t) = A_i^* \cup S_i^* \subseteq \beta^1(t)$, which is disjoint from $P$, a contradiction. □

We are now ready to prove Theorem 6.3. The guarded cutvertex decomposition we need can be obtained by invoking Lemma 6.9 on the tree decomposition $(T^1, \beta^1)$ produced by Lemma 6.12.

*Proof (of Theorem 6.3).* Let us compute first the tree decomposition $(T^1, \beta^1)$ of $H^\bullet$ given by Lemma 6.12 and then use Lemma 6.9 to obtain the tree decomposition $(T^2, \beta^2)$ and the mapping $f : V(T^2) \to V(T^1)$. For every $t \in V(T^2)$, we define $g^2(t) = g^1(f(t)) \cap \beta^2(t)$. We claim that $(T^2, \beta^2)$ and $g^2$ satisfy the required properties:

1. for every $t \in V(T^2)$, $g^2(t)$ is a vertex cover of $H[\beta^2(t)]$.
2. for every $t \in V(T^2)$ and every child $t'$ of $t$, we have $|\sigma^2(t') \setminus g^2(t)| \leq 1$.

For the first property, Lemma 6.12 implies that $g^1(t)$ is a vertex cover of $H[\beta^1(t)]$ for every $t \in V(T)$. Thus for every $t \in V(T^2)$, the subset $g^1(f(t)) \subseteq \beta^1(f(t))$ is a vertex cover of $H[\beta^1(f(t))]$. As $\beta^2(t) \subseteq \beta^1(f(t))$ by (P.2), it follows that $g^2(t) = g^1(f(t)) \cap \beta^2(t)$ is a vertex cover of $H[\beta^1(f(t)) \cap \beta^2(t)] = H[\beta^2(t)]$, as required

For the second property, consider a child $t'$ of $t$. By (P.4), we have that $C := H^\bullet[\alpha^2(t')]$ is connected, that is, $C$ is a connected component of $H^\bullet - \beta^2(t)$ with $V(C) \subseteq \gamma^2(t)$. Property (P.2) and the definition of $\gamma$ imply $\gamma^2(t) \subseteq \gamma^1(f(t))$, hence $V(C) \subseteq \gamma^1(f(t))$ also holds. By (P.3), we have that $\beta^1(f(t)) \setminus \beta^2(t)$ is disjoint from $V(C)$, hence not only $\beta^2(t)$, but its superset $\beta^1(t)$ is also disjoint from $V(C)$. Therefore, $C$ is also a connected component of $H^\bullet - \beta^1(f(t))$ and since it is in $\gamma^1(f(t))$, it is a connected component of $H^\bullet[\gamma^1(f(t))] - \beta^1(f(t))$. Now Lemma 6.12 implies that $|N_{H^\bullet}(C) \setminus g^1(f(t))| \leq 1$. By (P.5), we have $\sigma^2(t') = N_{H^\bullet}(C)$. As $\sigma^2(t') \subseteq \beta^2(t)$ by definition and $g^1(f(t)) \setminus g^2(t)$ is disjoint from $\beta^2(t)$ by the way we defined $g^2(t)$, we have

$$|\sigma^2(t') \setminus g^2(t)| = |\sigma^2(t') \setminus g^1(f(t))| = |N_{H^\bullet}(C) \setminus g^1(f(t)| \leq 1,$$

what we had to show for the second property. □

### 6.1.5 Proof of Theorem 6.2

We prove Theorem 6.2 by showing three implications. The most substantial one is the (2) ⇒ (3) implication, which was already shown by Theorem 6.3. The other two implications, (3) ⇒ (1) and (1) ⇒ (2), are much more straightforward.

*Proof (of Theorem 6.2).* (1) ⇒ (2). We prove the contrapositive: suppose that (2) is false and let us prove that (1) is false as well. If there is a graph $H \in \mathcal{H}$ such that $\widehat{H}$ has treewidth $c$, then (as $\widehat{H} \in \text{Spasm}(H)$), there is obviously a graph in $\text{Spasm}(H)$ with treewidth at least $c$. Suppose that there is a $\binom{c}{2}$-flower centered at color class $C_i$ of some $H \in \mathcal{H}$. Then by grouping the $c(c-1)$ endpoints of these paths into $c$ blocks of size $c-1$ in appropriate way and consolidating each group, we obtain a graph in $\text{Spasm}(H)$ where there are $c$ vertices in class $C_i$ with internally vertex disjoint paths between any two of them. Such a graph contains a $c$-clique minor and has treewidth at least $c-1$. Thus if (2) is not true for any $c$, then $\text{Spasm}(H)$ contains graphs with arbitrary large treewidth, making (1) also false.

(2) ⇒ (3). This is proved by Theorem 6.3.

(3) ⇒ (1). Let $H \in \mathcal{H}$ be a colored graph and let $H' \in \text{Spasm}(H)$. First we claim that if $H$ has a guarded cutvertex decomposition $(T, \beta, g)$ with guard size $c$, then there is such a decomposition $(T, \beta', g')$ for $H'$ as well. Let $f : V(H) \to V(H')$ be a mapping representing the partition used to obtain $H'$, that is, every vertex $v$ of $H$ with $f(v) = u$ was consolidated into $u$. It is not very difficult to show that setting $\beta'(t) = \{f(v) \mid v \in \beta(t)\}$ and $g'(t) = \{f(v) \mid v \in g(t)\}$ yields such a decomposition. The crucial fact to verify is that $X'_u = \{t \in V(T) \mid u \in \beta'(t)\}$ is connected in the tree $T$ for every $u \in V(H')$. Observe that $u$ appears in $\beta'(t)$ if and only if some vertex of $f^{-1}(u)$ appears in $\beta(t)$, thus $X'_u$ is the union of the subtrees $X_v = \{t \in V(T) \mid v \in \beta(t)\}$ for $v \in f^{-1}(u)$. As every vertex of $f^{-1}(u)$ has the same color and $(T, \beta)$ is a tree decomposition of $H^\bullet$, there is a node $t \in V(T)$ that is contained in every subtree $X_v$ corresponding to a vertex of $v \in f^{-1}(u)$, hence their union is also connected. It is easy to verify that $g'(t)$ is a vertex cover of $H'[\beta'(t)]$ for any $t \in V(T)$. Thus for every $H' \in \text{Spasm}(H)$, we have that $H'$ has a guarded cutvertex decomposition with guard size at most $c$ and then Lemma 6.6 implies that every graph in $\text{Spasm}(H)$ has treewidth at most $c$. □

## 6.2 Half-colorful matchings

In this section, we look at half-colorful matchings, which are the main reason why certain FPT cases cannot be solved in polynomial time. First, if we have guraded cutvertex decomposition where $\lambda(t)$ is large for some node $t$, then this can be used to extract a large half-colorful matching.

**Lemma 6.16.** *Let $(T, \beta, g)$ be a guarded cutvertex decomposition of a colored graph $H$ with guard size at most $c$ and the bags having at most $c$ colors. If $\lambda(t) \geq k$ for some $t \in V(T)$, then $H$ contains a half-colorful matching of size $k/c^2 - 1$.*

*Proof.* Let $x_1, \ldots, x_k$ be vertices of $\lambda(t)$. By definition, $x_i$ has a neighbor $y_i$ that is in $\alpha(t_i)$ for some child $t_i$ of $t$ and all the $t_i$'s are distinct. The $x_i$'s are in $\beta(t)$, hence they come from at most $c$ different color classes.

Thus there are at least $k/c$ of them from the same color class, say, $C_0$. By Lemma 6.7, each color can appear at most $c$ times on the $y_i$'s. This means that we can select $k/c^2 - 1$ of the $y_i$'s such that they are from distinct color classes different from $C_0$. This gives a half-colorful matching of the required size. □

The following lemma shows that a half-colorful matching can be cleaned in certain way, assuming there are no large flowers in the graph (in which case we know that he problem is hard anyway).

**Lemma 6.17.** *Let $H$ be a colored graph where there is no $c$-flower centered at any color class for some $c \geq 1$. If there is a half-colorful $k$-matching centered at color class $C_0$, then there is a set $S \subseteq C_0$ with $|S| \leq 3c$, vertices $x_1, \ldots, x_{k-3c}$ in $C_0$, and $k - 3c$ color classes $C_{i_1}, \ldots, C_{i_{k-3c}}$ such that $x_j$ is the unique neighbor of $C_{i_j}$ in $C_0 \setminus S$.*

*Proof.* Let us consider all triples $(x, y, C_i)$ where $x, y \in C_0$ are two distinct vertices and $C_i$ is a color class different from $C_0$ such that $x, y \in N(C_i)$. These triples can be considered as 3-element subsets of a universe $U$ consisting of the vertices of $C_0$ and elements representing the color classes different from $C_0$. Let us select a maximum collection of pairwise disjoint triples. We claim that if there are $m$ pairwise disjoint triples, then there is an $m$-flower centered at $C_0$ in $H$. Indeed, the triple $(x, y, C_i)$ means that there is a path from $x$ to $y$ in $H^{\bullet \setminus 0}$ that has either one internal vertex in $C_i$ (if $x$ and $y$ have common neighbor in $C_i$) or two internal vertices in $C_i$ (if $x$ and $y$ have distinct neighbors in $C_i$). Thus by the assumption of the lemma, the maximum collection contains less than $c$ triples. Let $S$ be the set of all vertices of $C_0$ appearing in the $m$ selected triples and let $\mathcal{C}$ contain all the color classes appearing in these triples; we have $|S| + |\mathcal{C}| \leq 3m < 3c$.

Suppose that $x_1y_1, \ldots, x_ky_k$ is a half-colorful matching where every $x_i$ is in $C_0$. Let us ignore those edges $x_iy_i$ of the matching where $x_i \in S$ or the color class of $b_i$ is in $\mathcal{C}$. This way, we ignore at most $|S| + |\mathcal{C}| \leq 3c$ edges, hence at least $k - 3c$ of them remain. Assume without loss of generality that $x_1y_1, \ldots, x_{k-3c}y_{k-3c}$ are among the remaining edges and let $C_{i_j}$ be the color class of $y_j$. It is clear that $x_j$ is a neighbor of the color class $C_{i_j}$ in $C_0 \setminus S$. Moreover, if there it has another neighbor $x'_j \in C_0 \setminus S$, then $(x_j, x'_j, C_{i_j})$ is a triple where $x_j, x'_j \notin S$ and $C_{i_j} \notin \mathcal{C}$. However, this means that the triple would be disjoint from each of the selected $m$ triples, contradicting the maximality of the selection. Thus $x_1, \ldots, x_{k-3c}$ satisfies the requirements of the lemma. □

Finally, we show how a half-colorful matching with the properties given in Lemma 6.17 can be exploited in a reduction from counting perfect matchings.

**Lemma 6.18.** *Let $H$ be a colored graph and suppose that we are given a set $S \subseteq C_0$ with $|S| \leq c$, vertices $x_1, \ldots, x_k$ in $C_0$, and $k$ color classes $C_1, \ldots, C_k$ such that $x_j$ is the unique neighbor of $C_j$ in $C_0 \setminus S$. Then given a $k + k$ vertex bipartite graph $G$ and oracle access to $\mathrm{Sub}(H \to \star)$, we can count the number of perfect matchings in $G$ in time $f(c)|V(H)|^{O(1)}$.*

*Proof.* We sketch the proof under the simplifying assumption that there is no color class outside $C_0, \ldots, C_k$, there are no edges inside color classes or between $C_i$ and $C_j$ with $i, j > 0$, and there are no isolated vertices. It is easy to extend the proof to the general case.

Let $H_0$ be the graph $H$ with all the edges incident to $C_0 \setminus S = \{x_1, \ldots, x_k\}$ removed. Suppose that $a_1, \ldots, a_k, b_1, \ldots, b_k$ are the vertices of $G$. We define the graph $H_G$ by starting with $H_0$, and for every edge $a_ib_j$ of $G$, we connect $N(a_i) \subseteq C_i$ with $b_j$.

The vertices of $C_0$ in $H$ are of two types: let $C_0^1$ contain those vertices that are adjacent to exactly one color class $C_i$ and let $C_0^2$ contain the remaining vertices, which are adjacent to more than one class. Clearly, we have $C_0^2 \subseteq S$ and in particular $|C_0^2| \leq c$. For every $v \in C_0^2$, let us choose two edges that connect it with two different color classes; let $F$ be the set of these $2|C_0^2| \leq 2c$ edges.

With oracle access to $\mathrm{Sub}(H \to \star)$, we can use standard inclusion-exclusion techniques to count the number of subgraphs of $H_G$ that are isomorphic to $H$ and moreover contains every vertex of $S$ and every edge of $F$. We claim that the number of such subgraphs is exactly the number of perfect matchings in $G$.

Consider a subgraph $H^* \subseteq H_G$ isomorphic to $H$ and let $f : V(H) \to V(H_G)$ be a corresponding embedding. Because every edge of $F$ appears in the subgraph, it is clear that $f$ has to map vertices of $C_0^2$ in $H$ to vertices of $C_0^2$ in $H_G$, because no other vertex in $H$ can be adjacent to two different color classes. It follows that it is also true that $f$ maps vertices of $C_0^1$ in $H$ to vertices of $C_0^1$ in $H_G$. Let $C_0^{1,i}$ be the subset of $C_0^1$ containng those vertices of $C_0^1$ that is adjacent only to color class $C_i$. Then exactly $|C_0^{1,i}| - 1$ of these vertices appear in

$S$. As every vertex of $S$ appears in the subgraph $H^*$, these vertices of $H_G$ are adjacent only to color class $C_i$, and there are no isolated vertices in $H$, it follows that $f$ should map vertices of $C_0^{1,i} \cap S$ of $H$ to vertices of $C_0^{1,i} \cap S$ of $H_G$ and maps exactly one vertex of $C_0^{1,i}$ to $C_0 \setminus S$ in $H_G$. Thus the subgraph $H^*$ describes a matching of $H_G$: the edges of $H^*$ connect every $x_i$ to a distinct color class $C_j$. Also, it is not hard to see that $H^*$ actually uses every edge betwen $x_i$ and this color class $C_j$ and $H^*$ uses every edge incident to $S$. This means that the number of edge sets of $H_G$ that form subgraphs isomorphic to $H$ is exactly the number of perfect matchings in $G$. □

## 6.3 Algorithm

The goal of this section is to prove the algorithmic result in Lemma 6.4. Given a colored graph $H$, we say that two vertices $u, v \in V(H)$ are *similar* if they have the same color and they have the same open neighborhood, that is, $N_H(u) = N_H(v)$. Note that if $u$ and $v$ are similar, then this means in particular that they are not adjacent. Clearly, similarity is an equivalence relation. We say that a partition $\Pi$ of $V(G)$ *respects similarity* if each class of $\Pi$ consists of vertices similar to each other (but it is possible that vertices in different classes are also similar).

We say that a colored graph $G$ is *ordered* if it is equipped with a total order $<$ on its vertices, for example, the vertex set is $[n]$ for some integer $n \geq 1$. Let $H$ and $G$ be two ordered graphs and let $\Pi$ be a partition of $V(H)$. We say that a subgraph embedding $\phi : V(H) \to V(G)$ is $\Pi$-*ordered* if whenever $u, v \in V(H)$ are in the same class and $u < v$ holds, then we have $\phi(u) < \phi(v)$. We denote by $\Pi$-OrdEmb$(H \to G)$ the number of $\Pi$-ordered embeddings from $H$ to $G$.

Observe that if $u$ and $v$ are similar vertices of $H$ and $\phi$ is a subgraph embedding from $H$ to $G$, then the values of $\phi(u)$ and $\phi(v)$ can be exchanged and the resulting mapping is still a valid subgraph embedding from $H$ to $G$. In fact, every subgraph embedding from $H$ to $G$ can be obtained from a $\Pi$-ordered subgraph embedding by permuting the values of $\phi$ inside each class of $\Pi$. Thus, as the following lemma states, the number of embeddings can be recovered easily from the number of $\Pi$-ordered subgraph embeddings.

**Lemma 6.19.** *Let $H$ and $G$ be two ordered graphs and let $\Pi = (P_1, \ldots, P_p)$ be a partition of $V(H)$ respecting similarity. Then we have*

$$\mathrm{Emb}(H \to G) = \Pi\text{-OrdEmb}(H \to G) \cdot \prod_{i=1}^{p} (|P_i|!).$$

For an ordered graph $G$ and subset $S \subseteq V(G)$, we say that $S$ is $\Pi$-*prefix* if whenever $u, v \in V(H)$ are in the same class of $\Pi$ with $u < v$ and $v \in S$ holds, then $u \in S$ holds as well. If $\phi : V(H) \to V(G)$ is a $\Pi$-ordered subgraph embedding and $i$ is some vertex of $G$, then the set $S_{\leq i} = \{v \in V(H) \mid \phi(v) \leq i\}$ of vertices that are mapped to vertices of $G$ not greater than $i$ is a $\Pi$-prefix set.

*Proof (of Lemma 6.4).* It will be convenient to assume that that the tree decomposition is tight: every vertex of $\sigma(t)$ has a neighbor in $\alpha(t)$. As discussed in Section 6.1.2, this can be achieved by easy transformations and these transformations do not increase the size of the sets $\lambda(t)$. Then $t' \in h_t(v)$ is equivalent to $\sigma(t') = g(t) \cup \{v\}$ (if the decomposition is not tight, then $\sigma(t') = g(t) \cup \{v\}$ is possible even if $v$ is not adjacent to any vertex of $\alpha(t')$ and hence $t' \notin h_t(v)$).

Let us define a partition $\Pi$ of $V(H)$ the following way. For every $t \in V(T)$, we partition $\beta(t) \setminus (g(t) \cup \lambda(t))$ according to similarity and let each such class be a class of $\Pi$ (by Lemma 6.5(1), these classes are disjoint). If a vertex does not appear in any of these classes, then let it appear in a singleton class of $\Pi$. Observe that $\Pi$ respects similarity. We present an algorithm for computing $\Pi$-OrdEmb$(H \to G)$. Then Lemma 6.19 can be used to compute $\mathrm{Emb}(H \to G)$.

The algorithm uses two layers of dynamic programming: the *outer* dynamic programming procedure uses the standard method of considering the nodes of the tree decomposition in a bottom-up order, and additionally there is an *inner* dynamic programming procedure at each node $t$, which restricts, for increasing values of $i$, that the vertices of $\beta(t) \setminus g(t)$ can be mapped only to the first $i$ vertices.

**The outer dynamic programming.** For a node $t \in V(T)$, let $H_t = H[\gamma(t)]$. For a set $S \subseteq V(H)$, let $\mathcal{F}_S$ be the set of all injective mappings from $S$ to $V(G)$. The goal of the outer dynamic programming

procedure is to compute, for every node $t \in V(T)$ and function $f \in \mathcal{F}_{\sigma(t)}$, the size of the set

$$\mathcal{W}_{t,f} = \{\phi \in \Pi\text{-OrdEmb}(H_t \to G) : \phi \text{ restricted to } \sigma(t) \text{ is } f\}.$$

If $r$ is the root of $H$, then $H_r = H$ and $\sigma(r) = \emptyset$ holds, hence $\Pi$-OrdEmb$(H_t \to G) = \mathcal{W}_{t,f}$, where $f$ is the unique empty function $\emptyset \to V(H)$. There is a slight abuse of notation here: in $\Pi$-OrdEmb$(H_t \to G)$, we would need to use the restriction of $\Pi$ to $V_t$ instead of $\Pi$. However, observe that every class of $\Pi$ is either contained in $V_t$ or disjoint from $V_t$, so this does not create any ambiguity.

We compute the values in a bottom up way: when computing $\#\mathcal{W}_{t,f}$, we assume that $\#\mathcal{W}_{t',f'}$ is already available for every child $t'$ of $t$ and every $f \in \mathcal{F}_{\sigma(t')}$. To compute $\#\mathcal{W}_{t,f}$, we solve a slightly more restricted problem: we fix the value of the embedding not only on $\sigma(t)$, but on $g(t) \supseteq \sigma(t)$. For every $t \in V(T)$ and $\overline{f} \in \mathcal{F}_{g(t)}$, we define

$$\overline{\mathcal{W}}_{t,\overline{f}} = \{\phi \in \Pi\text{-OrdEmb}(H_t \to G) : \phi \text{ restricted to } g(t) \text{ is } \overline{f}\}.$$

Clearly, we have $\#\mathcal{W}_{t,f} = \sum_{\overline{f} \in \mathcal{F}_{g(t)}, \overline{f}_{|\sigma(t)} = f} \#\overline{\mathcal{W}}_{t,\overline{f}}$, thus computing the values $\#\overline{\mathcal{W}}_{t,\overline{f}}$ is sufficient to compute $\#\mathcal{W}_{t,f}$.

**The inner dynamic programming.** The goal of the inner dynamic programming is to compute $\#\overline{\mathcal{W}}_{t,\overline{f}}$ for a given $t \in V(T)$ and $\overline{f} \in \mathcal{F}_{g(t)}$, assuming that the values $\#\mathcal{W}_{t',f'}$ are available for every child $t'$ of $t$ and $f' \in \mathcal{F}_{\sigma(t')}$. To describe the subproblems of the inner dynamic programming, we need some further definitions.

For any subset $g(t) \subseteq S \subseteq \beta(t)$, we define

$$V_{t,S} := S \cup \bigcup_{v \in S \setminus g(t)} \bigcup_{t' \in h_t(v)} \alpha(t') = S \cup \bigcup_{t' \text{ is a child of } t,\ \sigma(t') \subseteq S} \alpha(t')$$

and $H_{t,S} := H[V_{t,S}]$. That is, $V_{t,S}$ contains the subset $S$ of $\beta(t)$, and those branches of the tree decomposition that are rooted at some child $t'$ of $t$ that is hanging at some vertex in $S$, or more formally, $\sigma(t') \subseteq S$. Here we use the assumption $|\sigma(t') \setminus g(t)| \leq 1$, which implies that $\sigma(t') \subseteq S$ is equivalent to $t' \in h_t(v)$ for some $v \in S$.

For every $t \in V(T)$, we define $\mathcal{S}_t$ to contain every subset $S$ with $g(t) \subseteq S \subseteq \beta(t)$ that is a $\Pi$-prefix subset of $V(H)$. From the way we defined $\Pi$, it is clear that $\beta(t)$ itself is a $\Pi$-prefix subset of $V(H)$, hence $\beta(t) \in \mathcal{S}_t$. It is not difficult to bound the size of $\mathcal{S}_t$.

**Claim 6.20.** *For every $t \in V(T)$, we have $|\mathcal{S}_t| = 2^d \cdot |V(G)|^{c \cdot 2^c}$ and the collection $\mathcal{S}_t$ can be constructed in time $2^{O(d)}|V(G)|^{2^{O(c)}}$.*

*Proof.* The partition $\Pi$ classifies the vertices of $\beta(t) \setminus (g(t) \cup \lambda(t))$ according to the similarity relation. By Lemma 6.5(3), these vertices have at most $2^{|g(t)|} \leq 2^c$ possible neighborhoods and they have $c$ possible colors, thus $\Pi$ partitions $\beta(t) \setminus (g(t) \cup \lambda(t))$ into at most $c \cdot 2^c$ classes $P_1, \ldots, P_s$. If a $\Pi$-prefix set contains exactly $j$ vertices of one such class $P_i$, then we know that it contains exactly the first $j$ vertices of $P_i$. Thus each set in $\mathcal{S}_t$ can be completely specified by describing its intersection with $\lambda(t)$ and specifying the size of its intersection with each $P_i$. As $|\lambda(t)| \leq d$ by assumption, this gives $2^d \cdot n^{c \cdot 2^c}$ different possibilities. □

We are now ready to define the subproblems of the inner dynamic programming. For every $t \in V(T)$, $\overline{f} \in \mathcal{F}_{g(t)}$, $S \in \mathcal{S}_t$, and $0 \leq i \leq |V(G)|$, we define

$$\#\overline{\mathcal{W}}_{t,\overline{f},S,i} = \{\phi \in \Pi\text{-OrdEmb}(H_{t,S} \to G) : \phi \text{ restricted to } g(t) \text{ is } \overline{f} \text{ and } \phi(v) \leq i \text{ for every } v \in \beta(t) \setminus g(t)\}.$$

That is, we need to map only $H_{t,S}$, but we have a restriction on where the vertices of $\beta(t) \setminus g(t)$ can be mapped. As we have observed that $\beta(t) \in \mathcal{S}_t$ holds, the value $\#\overline{\mathcal{W}}_{t,\overline{f},\beta(t),|V(G)|}$ is defined, and it is equal to $\#\overline{\mathcal{W}}_{t,\overline{f}}$.

**Solving the subproblems.** The inner dynamic programming proceeds by solving the subproblems by increasing value of $i$: when computing $\#\overline{\mathcal{W}}_{t,\overline{f},S,i}$, we assume that that values $\#\overline{\mathcal{W}}_{t,\overline{f},S',i'}$ are available for every $i' < i$ and $S' \in \mathcal{S}_t$.

For $i = 0$, determining $\#\overline{\mathcal{W}}_{t,\overline{f},S,i}$ is trivial: its value is 1 if $S = g(t)$ and it is 0 if $S \supset g(t)$ (as then an embedding would need to map a vertex of $S \setminus g(t)$ to the first $i = 0$ vertices). For $i \geq 1$, consider vertex $i$ of $G$. If an embedding of $\overline{\mathcal{W}}_{t,\overline{f},S,i}$ maps no vertex to $i$, or maps a vertex not in $\beta(t) \setminus g(t)$ to $i$, then this embedding already appears in $\overline{\mathcal{W}}_{t,\overline{f},S,i-1}$. In particular, if $i \in \overline{f}(g(t))$, then every mapping of $\overline{\mathcal{W}}_{t,\overline{f},S,i}$ maps a vertex of $g(t)$ to $i$, hence we have $\#\overline{\mathcal{W}}_{t,\overline{f},S,i} = \#\overline{\mathcal{W}}_{t,\overline{f},S,i-1}$. Thus in the following, we can assume that $i \notin \overline{f}(g(t))$. Let us fix a vertex $v \in \beta(t) \setminus g(t)$ and count the number of embeddings $\phi \in \overline{\mathcal{W}}_{t,\overline{f},S,i}$ that map $v$ to $i$; summing these values for every $v \in \beta(t) \setminus g(t)$ and adding the number $\#\overline{\mathcal{W}}_{t,\overline{f},S,i-1}$ of embeddings that do not map any vertex of $\beta(t) \setminus g(t)$ to $i$ gives exactly the required value $\#\overline{\mathcal{W}}_{t,\overline{f},S,i}$. We consider two cases depending on whether $v$ is in $\lambda(t)$ or not.

*Case 1.* $v \notin \lambda(t)$. There are three obvious conditions that are necessary for the existence of embeddings in $\overline{\mathcal{W}}_{t,\overline{f},S,i}$ that map $v$ to $i$.

- Vertex $i$ needs to have the same color as $v$.
- For every neighbor $u$ of $v$ in $H$ (note that $u$ has to be in $g(t)$ by Lemma 6.5(3)), vertex $\overline{f}(u)$ should be a neighbor of $i$.
- The set $S' := S \setminus \{v\}$ should be $\Pi$-prefix: as $S$ is $\Pi$-prefix, the only way for $S'$ to lose this property is if there is a $u \in S'$ with $v < u$ that is in the same class of $\Pi$ as $v$, which in particular means that $u \in \beta(t) \setminus (g(t) \cup \lambda(t))$. But then the $\Pi$-ordered embedding $\phi \in \overline{\mathcal{W}}_{t,\overline{f},S,i}$ should map $u \in \beta(t) \setminus g(t)$ to a vertex greater than $i$, which is not possible by the definition of $\overline{\mathcal{W}}_{t,\overline{f},S,i}$.

We claim that if $v$ satisfies these three conditions, then any embedding of $\overline{\mathcal{W}}_{t,\overline{f},S',i-1}$ can be extended to an embedding of $\overline{\mathcal{W}}_{t,\overline{f},S,i}$ by mapping $v$ to $i$. The only subtle point here is that we need to argue that the embeddings in $\overline{\mathcal{W}}_{t,\overline{f},S',i-1}$ cannot already use $i$: this is because every vertex of $H$ with same color as $i$ and $v$ appears in $\beta(t)$ (Lemma 6.5(2)), we assumed that $\overline{f}$ maps no vertex of $g(t)$ to $i$, and an embedding in $\overline{\mathcal{W}}_{t,\overline{f},S',i-1}$ cannot map a vertex of $\beta(t) \setminus g(t)$ to $i$ by definition. Thus we can conclude that if the three conditions hold, the number of embeddings in $\overline{\mathcal{W}}_{t,\overline{f},S,i}$ that map $v$ to $i$ is exactly $\#\overline{\mathcal{W}}_{t,\overline{f},S',i-1}$.

*Case 2.* $v \in \lambda(t)$. Again, we have two obvious conditions:

- Vertex $i$ needs to have the same color as $v$.
- For very neighbor $u$ of $v$ in $H$ with $u \in g(t)$ (note that now $v$ can have neighbors not in $g(t)$), vertex $\overline{f}(u)$ should be a neighbor of $i$.

The set $S' := S \setminus \{v\}$ is always $\Pi$-prefix: $v \in \lambda(t)$ implies that $v$ is in a singleton class of $\Pi$. Again, we can argue that no embedding in $\overline{\mathcal{W}}_{t,\overline{f},S',i-1}$ can use vertex $i$ of $G$. This means that any embedding $\psi : V_{t,S'} \to V(G)$ appearing in $\overline{\mathcal{W}}_{t,\overline{f},S,i}$ can be extended to an embedding $\psi^+ : V_{t,S'} \cup \{v\} \to V(G)$ by setting $\psi^+(v) = i$. However, $V_{t,S}$ contains more than just $V_{t,S'} \cup \{v\}$: it contains $\alpha(t')$ for every $t' \in h_t(v)$. For any $t' \in h_t(v)$, let $f_{t'}$ be the restriction of $\psi^+$ to $\sigma(t')$. As $\sigma(t') \subseteq g(t) \cup \{v\}$ (by definition of $t' \in h_t(v)$), the function $f_{t'}$ depends only on $\overline{f}$ and on $i$. We have already computed the value $\#\mathcal{W}_{t',f_{t'}}$, which is the number of extensions of $f_{t'}$ to a embedding of $H_t$ to $G$. The crucial observation is that an embedding $\psi_{t'} \in \mathcal{W}_{t',f_{t'}}$ cannot conflict with $\psi^+$ in the sense that it is not possible that there is a vertex $w \in V_{t,S'} \cup \{v\}$ and a vertex $w' \in \alpha(t')$ with $\psi^+(w) = \psi_{t'}(w')$. This would be possible only if $w$ and $w'$ have the same color, that is, they are adjacent in $H^\bullet$. But then $w' \in \alpha(t')$ would imply $w \in \gamma(t')$, which is only possible if $w \in \sigma(t')$. Mappings $\psi^+$ and $\psi_{t'}$ agree on $\sigma(t')$ (as $f_{t'}$ is the restriction of $\psi^+$ to $\sigma(t')$, while $\psi_{t'}$ extends $f_{t'}$), hence $w \in \sigma(t')$ cannot conflict with $w'$. Thus any $\psi_{t'} \in \mathcal{W}_{t',f_{t'}}$ can be used to extend $\psi^+$ to an embedding from $H_{t,S'} \cup \{v\} \cup \alpha(t')$. Moreover, by the same reasoning, if $t', t'' \in h_t(v)$ are two distinct children of $t$, then two embeddings $\psi_{t'} \in \mathcal{W}_{t',f_{t'}}$ and $\psi_{t''} \in \mathcal{W}_{t'',f_{t''}}$ cannot conflict either: a vertex of $\alpha(t')$ cannot have the same color as a vertex of $\alpha(t'')$. Therefore, if we pick any combination of embeddings $\psi_{t'} \in \mathcal{W}_{t',f_{t'}}$ for each $t' \in h_t(v)$, then together they can be used to extend $\psi^+$ to an embedding of $H_{t,S}$. We can conclude that if the two conditions hold, then the number of embeddings in $\overline{\mathcal{W}}_{t,\overline{f},S,i}$ that map $v$ to $i$ is exactly $\#\overline{\mathcal{W}}_{t,\overline{f},S,i-1}$ times the product of the value $\#\mathcal{W}_{t',f_{t'}}$ for every $t' \in h_t(v)$.

We have shown that $\#\overline{\mathcal{W}}_{t,\overline{f},S,i}$ can be computed in polynomial time, assuming we have already computed $\#\overline{\mathcal{W}}_{t,\overline{f},S',i-1}$ for every $S' \in \mathcal{S}_t$, and $\#\mathcal{W}_{t',f'}$ for every child $t'$ of $t$ and mapping $f' \in \mathcal{F}_{\sigma(t')}$. For a given $t$ and $\overline{f}$, there are $2^d \cdot |V(G)|^{2^{O(c)}}$ values $\#\overline{\mathcal{W}}_{t,\overline{f}}$ compute (Claim 6.20) and solving these subproblems allows us to compute $\#\overline{\mathcal{W}}_{t,\overline{f}}$, hence we can conclude that $\#\overline{\mathcal{W}}_{t,\overline{f}}$ can be computed in time $2^d \cdot |V(G)|^{2^{O(c)}}$. For a given $t \in V(T)$, the number of possibilities for $\overline{f} \in \mathcal{F}_{g(t)}$ is at most $|V(G)|^{g(t)} \leq |V(G)|^c$, which means that the at most $|V(H)| \cdot |V(G)|^c$ subproblems $\#\overline{\mathcal{W}}_{t,\overline{f}}$ can be all solved in total time $2^d \cdot |V(G)|^{2^{O(c)}}$. □

### 6.4 Putting it together

Finally, we are ready to prove the main result of the section, classifying which of the FPT cases of counting colored patterns is polynomial-time solvable. The result is under assuming the Nonuniform Counting Exponential Time Hypothesis, which states that there is an $\epsilon > 0$ such that there is no infinite collection of algorithms $\{A_n \mid n \in N\}$ for some infinite set $N \subseteq \mathbb{Z}^+$ such that algorihm $A_n$ solves $n$-variable #3-SAT in time $2^{\epsilon n}$. By known reductions, we can replace #3-SAT with counting perfect matchings in an $n+n$ vertex bipartite graph [16].

**Theorem 6.21.** *Let $\mathcal{H}$ be a class of colored graph where, for every $H \in \mathcal{H}$, the graphs in* $\mathrm{Spasm}(H)$ *have treewidth at most $c$.*

1. *If there is a constant $h$ such that the largest half-colorful matching in every $H \in \mathcal{H}$ is at most $h \log |V(H)|$, then $\#\mathsf{Sub}(\mathcal{H})$ is polynomial-time solvable.*
2. *Otherwise, $\#\mathsf{Sub}(\mathcal{H})$ is not polynomial-time solvable, unless the Nonuniform Counting Exponential Time Hypothesis.*

*Proof.* Suppose that the first statement holds. Consider an instance of $\#\mathsf{Sub}(()\mathcal{H})$ with inputs $H$ and $G$. By Theorem 6.2, there is a constant $c'$ such that $\widehat{H}$ has treewidth at most $c'$ and there is no $c'$-flower centered at any color class of $H$. Thus by Theorem 6.3, there is a constant $c''$ such that we can obtain in polynomial time a guarded cutvertex decomposition with guard size $c''$ and at most $c''$ colors in each bag. If $\lambda(t) = k$ for some node $t$ of the decomposition, then Lemma 6.16 implies that there is a half-colorful matching of size $k/(c'')^2 - 1$ in $H$. Thus the assumption that there is no such matching larger than $h \log |V(H)|$ implies that $\lambda(t) = O(h(c'')^2 \log |V(H)|)$. It follows that the running time of the algorithm of Lemma 6.4 is $|V(H)|^{h(c'')^2} |V(H)|^{2^{O(c'')}}$, which is polynomial time for fixed constants $h$ and $c''$.

For the second statement, suppose that there is no such $h$. From the assumption that $\mathrm{Spasm}(H)$ has treewidth at most $c$ and Theorem 6.2, there is a constant $c'$ such that there is no $c'$-flower in $H$. Let us fix any $\epsilon > 0$. By assumption, there are infinitely many graphs $H \in \mathcal{H}$ where the size of the largest half-colorful matching is at least $(1/\epsilon) \log |V(H)|$. For every $n \geq 1$, if there is such a graph where the size of the largest half-colorful matching is exactly $n + 3c'$, then let us fix such a graph $H_n$. Note that $n \geq (1/\epsilon) \log |V(H)|$ means $|V(H)| \leq 2^{\epsilon n}$. Lemma 6.17 shows that there is a half-colorful matching of size $n$ satisfying certain conditions and then Lemma 6.18 can be used to reduce counting perfect matchings in an $n+n$ vertex bipartite graph to $\#\mathsf{Sub}(\mathcal{H})$ and use the assumed polynomial-time algorithm. This way, $A_n$ solves the problem of counting perfect matchings in time $(2^{\epsilon n})^{O(1)}$. As we can construct such an infinite sequence of algorithms for any $\epsilon > 0$, our complexity assumption fails. □

## 7 Open Problems

We have defined the space of graph motif parameters and explored three useful bases thereof, namely, Hom, Sub, and IndSub. These bases capture well-studied classes of counting problems, and we could use basis changes to transfer results between these classes. Are there other computationally interesting bases? Moreover, are there other interesting subspaces of the space of all graph parameters other than the graph motif parameters?

## Acknowledgments

Thanks a lot to Édouard Bonnet for pointing out [52] and [22].

## References

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 98–117, 2015. doi:10.1109/FOCS.2015.16.

[2] Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and S Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):i241–i249, 2008. doi:10.1093/bioinformatics/btn163.

[3] Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BFb0049422.

[4] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting paths and packings in halves. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA)*, pages 578–586, 2009. doi:10.1007/978-3-642-04128-0_52.

[5] Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. Counting thin subgraphs via packings faster than meet-in-the-middle time. In *Proceedings of the 25th Annual Symposium on Discrete Algorithms (SODA)*, pages 594–603, 2014. doi:10.1137/1.9781611973402.45.

[6] Markus Bläser and Radu Curticapean. Weighted counting of k-matchings is #W[1]-hard. In *Proceedings of the 7th International Symposium on Parameterized and Exact Computation (IPEC)*, pages 171–181, 2012. doi:10.1007/978-3-642-33293-7_17.

[7] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

[8] Christian Borgs, Jennifer Chayes, László Lovász, Vera T Sós, and Katalin Vesztergombi. Counting graph homomorphisms. In *Topics in discrete mathematics*, pages 315–371. Springer, 2006. doi:10.1007/3-540-33700-8_18.

[9] Cornelius Brand and Marc Roth. Parameterized counting of trees, forests and matroid bases. In *Proceedings of the 12th International Computer Science Symposium in Russia (CSR)*, 2017 (to appear). URL: https://arxiv.org/abs/1611.01823.

[10] Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.

[11] Jin Chen, Wynne Hsu, Mong Li Lee, and See-Kiong Ng. Nemofinder: Dissecting genome-wide protein-protein interactions with meso-scale network motifs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 106–115. ACM, 2006. doi:10.1145/1150402.1150418.

[12] Yijia Chen, Martin Grohe, and Bingkai Lin. The hardness of embedding grids and walls. *arXiv preprint arXiv:1703.06423*, 2017. URL: http://arxiv.org/abs/1703.06423.

[13] Yijia Chen, Marc Thurley, and Mark Weyer. Understanding the complexity of induced subgraph isomorphisms. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 587–596, 2008. doi:10.1007/978-3-540-70575-8_48.

[14] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual Symposium on Theory of Computing (STOC)*, pages 151–158, 1971. doi:10.1145/800157.805047.

[15] Radu Curticapean. Counting matchings of size k is W[1]-hard. In *Proceedings of the 40th International Conference on Automata, Languages, and Programming (ICALP)*, pages 352–363, 2013. doi:10.1007/978-3-642-39206-1_30.

[16] Radu Curticapean. Block interpolation: A framework for tight exponential-time counting complexity. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 380–392, 2015. doi:10.1007/978-3-662-47672-7_31.

[17] Radu Curticapean and Dániel Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 130–139, 2014. doi:10.1109/FOCS.2014.22.

[18] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

[19] Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theoretical Computer Science*, 329(1):315–323, 2004. doi:10.1016/j.tcs.2004.08.008.

[20] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Transactions on Algorithms*, 10(4):21:1–21:32, 2014. doi:10.1145/2635812.

[21] Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Counting H-colorings of partial k-trees. *Theoretical Computer Science*, 281(1-2):291–309, 2002. doi:10.1016/S0304-3975(02)00017-8.

[22] Zdeněk Dvořák and Sergey Norin. Strongly sublinear separators and polynomial expansion. *SIAM Journal on Discrete Mathematics*, 30(2):1095–1101, 2016. doi:10.1137/15M1017569.

[23] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004. doi:10.1016/j.tcs.2004.05.009.

[24] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

[25] Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012. doi:10.1007/s00493-012-2536-z.

[26] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

[27] Joshua A Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pages 92–106. Springer, 2007. doi:10.1007/978-3-540-71681-5_7.

[28] Martin Grohe. Descriptive complexity, canonisation, and definable graph structure theory. Manuscript. URL: http://www.lii.rwth-aachen.de/de/mitarbeiter/13-mitarbeiter/professoren/39-book-descriptive-complexity.html.

[29] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1):1:1–1:24, 2007. doi:10.1145/1206035.1206036.

[30] Russell Impagliazzo and Ramamohan Paturi. On the complexity of $k$-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

[31] Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978. doi:10.1137/0207033.

[32] Bart M. P. Jansen and Dániel Marx. Characterizing the easy-to-find subgraphs from the viewpoint of polynomial-time algorithms, kernels, and Turing kernels. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 616–629, 2015. doi:10.1137/1.9781611973730.42.

[33] Mark Jerrum and Kitty Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *Journal of Computer and System Sciences*, 81(4):702–716, 2015. doi:10.1016/j.jcss.2014.11.015.

[34] Mark Jerrum and Kitty Meeks. Some hard families of parameterized counting problems. *ACM Transactions on Computation Theory*, 7(3):11, 2015. doi:10.1145/2786017.

[35] Mark Jerrum and Kitty Meeks. The parameterised complexity of counting even and odd induced subgraphs. *Combinatorica*, pages 1–26, 2016. doi:10.1007/s00493-016-3338-5.

[36] Zahra Razaghi Moghadam Kashani, Hayedeh Ahrabian, Elahe Elahi, Abbas Nowzari-Dalini, Elnaz Saberi Ansari, Sahar Asadi, Shahin Mohammadi, Falk Schreiber, and Ali Masoudi-Nejad. Kavosh: A new algorithm for finding network motifs. *BMC bioinformatics*, 10(1):318, 2009. doi:10.1186/1471-2105-10-318.

[37] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004. doi:10.1093/bioinformatics/bth163.

[38] Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. *Information Processing Letters*, 74(3):115–121, 2000. doi:10.1016/S0020-0190(00)00047-8.

[39] Ioannis Koutis and Ryan Williams. LIMITS and applications of group algebras for parameterized problems. *ACM Transactions on Algorithms*, 12(3):31:1–31:18, 2016. doi:10.1145/2885499.

[40] Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Counting and detecting small subgraphs via equations. *SIAM Journal on Discrete Mathematics*, 27(2):892–909, 2013. doi:10.1137/110859798.

[41] Bingkai Lin. The parameterized complexity of $k$-biclique. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 605–615, 2015. doi:10.1137/1.9781611973730.41.

[42] László Lovász. Operations with structures. *Acta Mathematica Hungarica*, 18(3-4):321–328, 1967.

[43] László Lovász. *Large networks and graph limits*, volume 60. American Mathematical Society Providence, 2012.

[44] Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010. doi:10.4086/toc.2010.v006a005.

[45] Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems. *Discrete Applied Mathematics*, 198:170–194, 2016. doi:10.1016/j.dam.2015.06.019.

[46] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002. arXiv:http://science.sciencemag.org/content/298/5594/824.full.pdf, doi:10.1126/science.298.5594.824.

[47] Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

[48] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

[49] Saeed Omidi, Falk Schreiber, and Ali Masoudi-Nejad. MODA: An efficient algorithm for network motif discovery in biological networks. *Genes & genetic systems*, 84(5):385–395, 2009. doi:10.1266/ggs.84.385.

[50] Benjamin Schiller, Sven Jager, Kay Hamacher, and Thorsten Strufe. Stream - A stream-based algorithm for counting motifs in dynamic graphs. In *Proceedings of the 2nd International Conference on Algorithms for Computational Biology (AlCoB)*, pages 53–67, 2015. doi:10.1007/978-3-319-21233-3_5.

[51] Falk Schreiber and Henning Schwöbbermeyer. Frequency concepts and pattern detection for the analysis of motifs in networks. In *Transactions on computational systems biology III*, pages 89–104. Springer, 2005. doi:10.1007/11599128_7.

[52] Alexander D. Scott and Gregory B. Sorkin. Linear-programming design and analysis of fast algorithms for Max 2-CSP. *Discrete Optimization*, 4(3-4):260–287, 2007. doi:10.1016/j.disopt.2007.08.001.

[53] Julian R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976. doi:10.1145/321921.321925.

[54] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

[55] Sebastian Wernicke. Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4), 2006. doi:10.1109/TCBB.2006.51.

[56] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC)*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

[57] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 645–654, 2010. doi:10.1109/FOCS.2010.67.

[58] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM Journal on Computing*, 42(3):831–854, 2013. doi:10.1137/09076619X.

# A Appendix

In this section, we provide those proofs that we chose not to include in the main text.

*Proof of Proposition 1.6.* We begin by computing an optimal tree decomposition $(T, \beta)$ of $H$, for example via the $O(1.7549^{|V(F)|})$ time exact algorithm by Fomin and Villanger [25]. Without loss of generality, we assume it to be a *nice* tree decomposition, in which each node has at most two children; the leaves satisfy $\beta(v) = \emptyset$, the nodes with two children $w, w'$ satisfy $\beta(v) = \beta(w) = \beta(w')$ and called *join* nodes, the nodes with one child $w$ satisfy $\beta(v) = \beta(w) \cup \{x\}$ and called *introduce* nodes, or $\beta(v) \cup \{x\} = \beta(w)$ and called *forget* nodes.

Recall that $\alpha(v)$ be the union of all bags at or below vertex $v$ in the tree $T$. The goal of the dynamic programming algorithm is to build up the following information $I_v$ at each vertex $v$ of $T$: For each $h \in \mathrm{Hom}(H[\beta(v)] \to G)$, we let $I_v(h)$ be the number of homomorphisms $\bar{h} \in \mathrm{Hom}(H[\alpha(v)] \to G)$ such that $\bar{h}$ extends $h$. We observe some properties of $I_v$.

Let $v$ be a leaf of $T$. Then $\beta(v) = \emptyset$ and we define $I_v(h) = 1$ for the empty function $h$. For each leaf $v$, we can set up the dynamic programming table entry $I_v$ in constant time.

Let $v$ be an *introduce* vertex of $T$. Let $w$ be its unique child in the tree. Suppose the vertex $x \in V(G)$ is introduced at this node, that is, $\beta(w) \cup \{x\} = \beta(v)$. Let $h \in \mathrm{Hom}(H[\beta(v)] \to G)$, and let $h'$ be $h$ with $x$ removed from its domain. Then $I_v(h) = I_w(h')$ holds because $N_H(x) \cap \alpha(v) = N_H(x) \cap \beta(v)$. Thus at introduce vertices, we can compute $I_v$ from the table $I_w$ by going over all table entries $h$ (there are at most $|V(G)|^{|\beta(v)|} \leq |V(G)|^{\mathrm{tw}(H)+1}$ of them) and writing the value $I_w(h')$ to $I_v(h)$.

Let $v$ be a *forget* vertex of $T$. Let $w$ be its unique child in the tree. Suppose the vertex $x \in V(G)$ is forgotten at this node, that is, $\beta(w) \setminus \{x\} = \beta(v)$. Then the neighborhood of $x$ is contained in $\alpha(w)$. Let $h \in \mathrm{Hom}(H[\beta(v)] \to G)$. For each $z \in V(G)$, let $h_z : \beta(w) \to V(G)$ be the unique function that is consistent with $h$ and satisfies $h(x) = z$. Then we have

$$I_v(h) = \sum_{z \in V(G)} \Big[ h_z \in \mathrm{Hom}(H[\beta(w)] \to G) \Big] \cdot I_w(h_z)\,.$$

For each of the at most $|V(G)|^{|\beta(v)|} = |V(G)|^{|\beta(w)|-1} \leq |V(G)|^{\mathrm{tw}(H)}$ functions $h$, we compute $I_v(h)$ as follows: first we determine the set $Z$ of all $z$ for which $h_z$ is a homomorphisms, and then we sum up the corresponding entries $I_w(h_z)$. Let $N(x)$ be the open neighborhood of $x$ in $H$, and let $N'$ be the image of $N(x) \cap \beta(v)$ under $h$. Then the set $Z$ is the set of all vertices $z$ that are adjacent to all vertices in $N'$. Clearly $N'$ has at most $|\beta(v)| \leq \mathrm{tw}(H)$ elements. Thus $Z = \bigcap_{y \in N'} N_G(y)$ holds, and we can compute this intersection in time $\tilde{O}(\mathrm{tw}(H) \cdot |V(G)|)$. The running time for forget vertices is therefore $\tilde{O}(\mathrm{tw}(H) \cdot |V(G)|^{\mathrm{tw}(H)+1})$

Let $v$ be a *join* vertex of $T$, that is, it has exactly two children $w$ and $w'$ with $\beta(v) = \beta(w) = \beta(w')$. Let $h \in \mathrm{Hom}(H[\beta(v)] \to G)$. Then $I_v(h) = I_w(h) \cdot I_{w'}(h)$. This operation can be computed in time $|V(G)|^{|\beta(v)|} \leq |V(G)|^{\mathrm{tw}(H)+1}$. □

To prove Proposition 1.9, we state a vertex-colorful version of the subgraph counting problem. For two graphs $F$ and $G$, we say that $G$ is $F$-colored graph if there is a homomorphism $f \in \mathrm{Hom}(G \to F)$ from $G$ to $F$. Let $\mathrm{PartitionedSub}(F \to G)$ be the set of all subgraphs $H$ of $G$ that are isomorphic to $F$ and *vertex-colorful*, that is, these subgraphs $H$ satisfy $|f^{-1}(v) \cap V(H)| = 1$ for all $v \in V(F)$. In the problem $\#\mathsf{PartitionedSub}(\mathcal{F})$ for a fixed graph class $\mathcal{F}$, we are given a graph $F \in \mathcal{F}$ and an $F$-colored graph $G$, and we are asked to compute the number $\#\mathrm{PartitionedSub}(F \to G)$. A full dichotomy for this class of problems is known, and it establishes treewidth as the tractability criterion, together with near-tight lower bounds under $\#\mathsf{ETH}$.

**Theorem A.1 (Corollary 6.2 and 6.3 of [44]).** *Let $\mathcal{F}$ be a recursively enumerable family of graphs such that the treewidth of graphs in $\mathcal{F}$ is unbounded. If $\#\mathsf{ETH}$ is true, the problem $\#\mathsf{PartitionedSub}(\mathcal{F})$ cannot be solved in time $f(H) \cdot n^{o(\mathrm{tw}(H)/\log \mathrm{tw}(H))}$ for patterns $H \in \mathcal{F}$ of treewidth $\mathrm{tw}(H)$.*

We are ready to prove the following proposition.

**Proposition 1.9, restated.** *Let $\mathcal{F}$ be a recursively enumerable class of graphs of unbounded treewidth. If $\#\mathsf{ETH}$ holds, there is no $f(H) \cdot |V(G)|^{o(\mathrm{tw}(H)/\log \mathrm{tw}(H))}$ time algorithm to compute $\#\mathsf{Hom}(\mathcal{F})$ for given graphs $H \in \mathcal{F}$ and $G$.*

*Proof.* We reduce from $\#\mathsf{PartitionedSub}(\mathcal{F})$ to $\#\mathsf{Hom}(\mathcal{F})$. Given an instance $(H, G, f)$ with graphs $H \in \mathcal{F}$ and $G$ and a given homomorphism $f \in \mathrm{Hom}(G \to F)$, the reduction wants to compute $\mathrm{PartitionedSub}(H \to G)$. The reduction uses a straightforward inclusion–exclusion argument. For a set $A \subseteq V(H)$, let $G_{-A}$ be the graph obtained from $G$ by deleting all vertices $v \in V(G)$ with $f(v) \in A$. Then the set $\mathrm{Hom}(H \to G) \setminus \bigcup_{\emptyset \neq A \subseteq V(H)} \mathrm{Hom}(H \to G_{-A})$ contains all homomorphisms $h \in \mathrm{Hom}(H \to G)$ that are color-preserving, that is, they map each vertex $v \in V(H)$ to a vertex $h(v) \in V(G)$ with the property that $f(h(v)) = v$. In particular, these homomorphisms are injective. By the principle of inclusion and exclusion, their number is equal to

$$\sum_{\emptyset \neq A \subseteq V(H)} (-1)^{|A|} \cdot \#\mathrm{Hom}(H \to G_{-A}) \,.$$

Dividing by $\#\mathrm{Aut}(H)$ yields the number of vertex-colorful $H$-copies in $G$ as required. □