Homomorphisms Are a Good Basis for Counting Small Subgraphs[*]

Radu Curticapean[†]
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary
radu.curticapean@gmail.com

Holger Dell
Saarland University and Cluster of Excellence (MMCI)
Saarbrücken, Germany
hdell@mmci.uni-saarland.de

Dániel Marx[‡]
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary
dmarx@cs.bme.hu

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.

We use the framework of graph motif parameters to obtain faster algorithms for counting subgraph copies of fixed graphs H in host graphs G. More precisely, 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.

CCS CONCEPTS

• Mathematics of computing → Graph algorithms; **• Theory of computation → Problems, reductions and completeness**; **Fixed parameter tractability**;

KEYWORDS

counting subgraphs, homomorphisms, fixed-parameter tractability, exponential time hypothesis

ACM Reference format:
Radu Curticapean, Holger Dell, and Dániel Marx. 2017. Homomorphisms Are a Good Basis for Counting Small Subgraphs. In *Proceedings of 49th Annual ACM SIGACT Symposium on the Theory of Computing, Montreal, Canada, June 2017 (STOC'17),* 14 pages.
DOI: 10.1145/3055399.3055502

1 INTRODUCTION

Deciding the existence of subgraph patterns H in input graphs G constitutes the classical *subgraph isomorphism problem* [13, 49], 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 [50].

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 [42], and there is a large body of work dedicated to the algorithmic discovery of network motifs [2, 10, 24, 32, 33, 45–47, 51].

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 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

[*]Part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing, and at the Dagstuhl Seminar 17041 – "Randomization in Parameterized Complexity".
[†]Dr. Curticapean was supported by ERC grant PARAMTIGHT (No. 280152).
[‡]Dr. Marx was supported by ERC grant PARAMTIGHT (No. 280152).

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 [38, 39] 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.

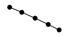
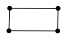
1.1 Counting Small Subgraphs

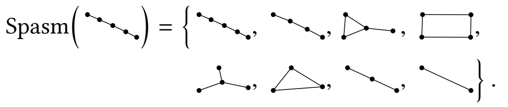
For any fixed k-vertex pattern graph H, we can count all subgraph copies of H in an n-vertex host graph G using brute-force for a running time of $O(n^k)$. While this running time is polynomial for any fixed H, it quickly becomes infeasible as k grows. Fortunately enough, non-trivial improvements on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

For instance, we can count triangles in the same time $O(n^\omega)$ that it takes to multiply two $(n \times n)$-matrices [27]. It is known that $\omega < 2.373$ holds [23, 52]. This approach can be generalized from triangles to k-cliques with $k \in \mathbb{N}$ [43], 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 [21, 34].

Secondly, 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 [4, 35]. 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 $n^{0.455k+O(1)}$.

Finally, 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 $n^{\mathrm{vc}(H)+O(1)}$ [54] (also cf. [15, 36]). 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


(1)

Our main algorithmic result can then be stated as follows:

Theorem 1.1. *Given a k-edge graph H and an n-vertex graph G, we can compute $\#\mathrm{Sub}(H \to G)$ 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 [54] 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 [48, 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.

Corollary 1.2. *Given a k-edge graph H and an n-vertex graph G, we can compute $\#\mathrm{Sub}(H \to G)$ 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 than its predecessors.

When H is the triangle , the algorithm from Theorem 1.1 matches the running time $O(n^3)$ of the exhaustive search method. To achieve a smaller constant in the exponent, we have to use matrix multiplication, since faster triangle detection is equivalent to faster Boolean matrix multiplication [53]. 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, then 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.*

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 [15] 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 [26], adapted to the counting setting [18], 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 [9] — 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 $\#\mathsf{Sub}(H \to G)$ 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.

The parameterized reduction in [15] 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" [40], 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 [40, 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 FPT ≠ #W[1]. In this setting, Theorem 1.4 takes on the following form:

Theorem 1.5 ([15]). *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 $|V(H)|$.*

The original proof of Theorem 1.5 relied on the #W[1]-completeness of counting k-matchings, which was nontrivial on its own [6, 14]. 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 there is no problem of the form $\#\mathsf{Sub}(\mathcal{H})$ that is "truly" FPT– 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 FPT ≠ #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 [12] $\#\mathsf{Sub}(\mathcal{H})$.

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 [28], 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 [11, 37].

1.2 Counting Small Homomorphisms

Similar classifications as Theorem 1.5 for counting subgraphs were previously known for counting homomorphisms [17] 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 $\#\mathsf{Hom}(H \to G)$ 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 interpreted as a dynamic programming algorithm: Given a table with the number of ℓ-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. [19]). *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.*

This proposition is the basis of our main algorithmic result for subgraphs (Theorem 1.1) and other graph motif parameters. 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 $\#\mathrm{Hom}(\mathcal{H})$.

Theorem 1.8 (Dalmau and Jonsson [17]). *Let $\mathcal{H}$ be a recursively enumerable class of graphs. If $\mathcal{H}$ has bounded treewidth, then the problem $\#\mathrm{Hom}(\mathcal{H})$ of counting homomorphisms from graphs in $\mathcal{H}$ into host graphs is polynomial-time solvable. Otherwise, it is #W[1]-complete when parameterized by $|V(H)|$.*

We also need the following hardness under #ETH, the proof of which is a simple corollary of [40, 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 $\#\mathrm{Hom}(\mathcal{H})$ for given 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 [25]: 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 [12, 29–31, 34, 41], 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 ([12]). *Let $\mathcal{H}$ be a recursively enumerable class of graphs. If $\mathcal{H}$ is finite, then the problem $\#\mathrm{Ind}(\mathcal{H})$ of counting induced subgraphs from $\mathcal{H}$ is polynomial-time solvable. Otherwise, it is #W[1]-complete when parameterized by $|V(H)|$.*

Jerrum and Meeks [29–31, 41] introduced the following generalization of the problems $\#\mathrm{Ind}(\mathcal{H})$ to fixed graph properties Φ: Given a graph G and $k\in\mathbb{N}$, the task is to compute the number of induced k-vertex subgraphs that have property Φ. Let us call this problem $\#\mathrm{IndProp}(\Phi)$. They identified some classes of properties Φ 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 Φ is a decidable graph property, then $\#\mathrm{IndProp}(\Phi)$ is fixed-parameter tractable or #W[1]-hard when parameterized by $|V(H)|$.*

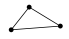
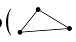
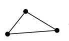
1.4 A Unified View: Graph Motif Parameters

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 [7, 38]. 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.

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.

Cases where homomorphisms from H to G are not automatically injective are more interesting. 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, $\#\mathrm{Hom}(\quad\to G) - \#\mathrm{Hom}(\quad\to G)$ 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 ρ of $V(H)$ in which each block is an independent set, and then form the *quotient graph* H/ρ obtained from H by merging each block of ρ 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 ρ_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 ρ_h is the finest partition, i.e., the partition where each block has size one.

The homomorphisms from H/ρ to G are precisely those homomorphisms from H to G that fail to be injective "at least as badly as ρ", that is, those homomorphisms f whose type ρ_f is a coarsening of ρ. As remarked by Lovász et al. [7, 38], one can then use Möbius inversion, a generalization of the inclusion–exclusion principle, to turn this observation into the "inverse" identity

$$\#\mathrm{Sub}(H \to G) = \sum_{\rho} \mu_\rho \cdot \#\mathrm{Hom}(H/\rho \to G)\,, \text{where} \qquad (2)$$

$$\mu_\rho = \frac{(-1)^{|V(H)|-|V(H/\rho)|} \cdot \prod_{B\in\rho}(|B|-1)!}{\#\mathrm{Aut}(H)}\,. \qquad (3)$$

The sum in (2) ranges over all partitions ρ 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/ρ to G. Each H/ρ is isomorphic to a graph in $\mathrm{Spasm}(H)$, and so by collecting terms for isomorphic graphs, (2) represents the number of 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 $\alpha_1, \dots, \alpha_t \in \mathbb{Q}$ and graphs $H_1, \dots, H_t$ such that, for all graphs G, we have

$$f(G) = \sum_{i=1}^{t} \alpha_i \cdot \#\mathrm{IndSub}(H_i \to G)\,. \qquad (4)$$

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 $\alpha_1, \dots, \alpha_t$ and $H_1, \dots, H_t$. 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 (4) 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 mostly neglect for our purposes.

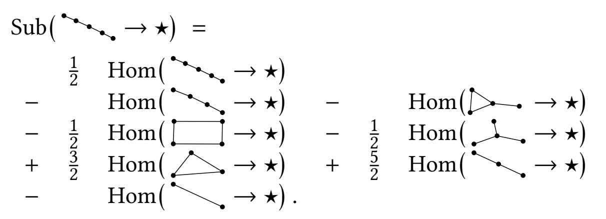
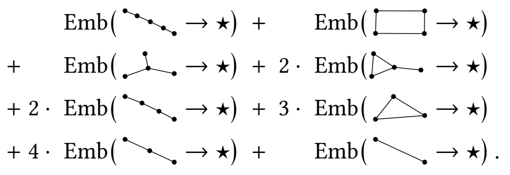


Figure 1: An example for (2), where H is the path with four edges. The number of subgraphs is represented as a linear combination of homomorphisms from graphs $F \in \mathrm{Spasm}(H)$. Each such F has treewidth at most two, so we can compute the homomorphism numbers in time $O(n^3)$ via Proposition 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.

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 $\alpha_1, \dots, \alpha_t \in \mathbb{Q}$ and $H_1, \dots, H_t$ be graphs such that $f(G) = \sum_i \alpha_i \cdot \#\mathrm{Hom}(H_i \to G)$ holds 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 the linear combination $f(G)$ 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:


Emb(→ ★) + Emb(→ ★)
+ Emb(→ ★) + 2 · Emb(→ ★)
+ 2 · Emb(→ ★) + 3 · Emb(→ ★)
+ 4 · Emb(→ ★) + Emb(→ ★) .

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 $\mathrm{Hom}(\quad\to \star)$, 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 graph motif problem $\#\mathsf{Ind}(\mathcal{A})$ is then given a pattern-coefficient list from $\mathcal{A}$ and a graph G, and is supposed to compute the linear combination (4). 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* (4) *re-expressed as linear combinations of homomorphisms contain nonzero coefficients only for graphs of treewidth at most t, the problem $\#\mathsf{Ind}(\mathcal{A})$ can be computed in time $f(\alpha) \cdot n^{t+1}$. Otherwise, the problem is #W[1]-hard parameterized by $|\alpha|$ and does not have $f(\alpha) \cdot n^{o(t/\log t)}$ time algorithms under #ETH.*

With respect to fixed-parameter tractability vs. #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 (so we are 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 ([15, 17, 41]). *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 #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 ρ 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 #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, $1, \ldots, k$ 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: those that are polynomial-time solvable, and those that are not polynomial-time solvable under the complexity assumption 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.

1.6 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]. Due to space constraints, the colored subgraph problems are deferred to the full version.

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 [16, 22, 44] 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 $f(k) \cdot \mathrm{poly}(n)$, where $n = |x|$ and $k = \kappa(x)$.

A *parameterized Turing reduction* is a Turing reduction from a problem (Π, κ) to a problem (Π', κ') such that the reduction runs in $f(\kappa(x)) \cdot \mathrm{poly}(|x|)$ 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 [26] asserts that satisfiability of 3-CNF formulas cannot be decided substantially faster than by trying all possible assignments. The counting version of this hypothesis [18] 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. [9] 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. 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, β), where T is a tree and β 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, β) 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.

3 THE SPACE OF GRAPH MOTIF PARAMETERS

We develop our interpretation of the general setup of Lovász [39]. 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 α 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 α as row vectors, this definition can also be phrased as requiring $f = \alpha \cdot \mathrm{IndSub}$ for α of finite support. Here, for two vectors $\alpha, \beta \in \mathbf{Q}^{\mathcal{G}^*}$, we define the scalar product (α, β) 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 α 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 α.

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, all 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

$$\mathrm{Sub}(H,G) = \sum_{F \in \mathcal{G}^*} \mathrm{Ext}(H,F) \cdot \mathrm{IndSub}(F,G). \tag{5}$$

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 (5) takes on the concise form

$$\mathrm{Sub} = \mathrm{Ext} \cdot \mathrm{IndSub}. \tag{6}$$

Since Sub, Ext, and IndSub are upper triangular matrices whose diagonal entries are all 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: We can express every function $\mathrm{IndSub}(H,\star)$ as a 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 (5) can be interpreted as a zeta transform over the subset lattice [7, eq. (13) and (14)], so we can perform Möbius inversion to prove that

$$\mathrm{Ext}^{-1}(H,F) = (-1)^{|E(F)|-|E(H)|} \cdot \mathrm{Ext}(H,F) \tag{7}$$

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

$$\mathrm{Hom}(H,G) = \sum_{F \in \mathcal{G}^*} \mathrm{Surj}(H,F) \cdot \mathrm{Sub}(F,G). \tag{8}$$

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

$$\mathrm{Hom} = \mathrm{Surj} \cdot \mathrm{Sub}, \tag{9}$$

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 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 this end, 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:

$$\mathrm{Spasm}(H) = \left\{ F \in \mathcal{G}^* : \ \mathrm{Surj}(H, F) > 0 \right\}. \tag{10}$$

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/ρ 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/ρ, and we turn parallel edges into simple edges.

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

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

$$\mathrm{Surj}(H, F) = \#\mathrm{Aut}(F) \cdot \sum_{\rho \in \mathrm{Part}(H)} [H/\rho \simeq F], \tag{11}$$

For two partitions $\rho, \rho' \in \mathrm{Part}(H)$, we write $\rho \geq \rho'$ if ρ is coarser than ρ', that is, if every block of ρ' is contained in a block of ρ. This partial order gives rise to the *partition lattice* $(\mathrm{Part}(H), \geq)$ whose minimal element $\perp$ is the finest partition, i.e., the partition whose blocks all have size one. Now (8) 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 $\mathrm{Hom}(H, G) = \hat{f}(\perp)$. By Möbius inversion, we get (see also [7, eq. (15)]):

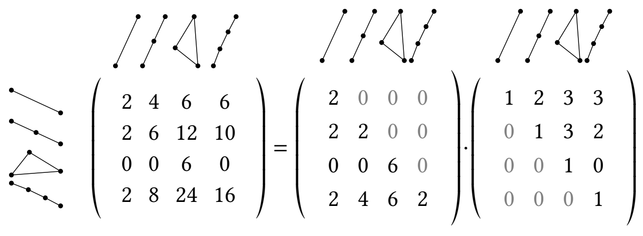
$$f(\rho) = \sum_{\rho' \geq \rho} (-1)^{|\rho| - |\rho'|} \cdot \Big(\prod_{B \in \rho'} (\lambda(\rho, \rho', B) - 1)! \Big) \cdot \hat{f}(\rho'),$$

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

$$\mathrm{Surj}^{-1}(H, F) = \frac{(-1)^{|V(H)| - |V(F)|}}{\#\mathrm{Aut}(H)} \cdot \sum_{\substack{\rho \in \mathrm{Part}(H) \\ H/\rho \simeq F}} \prod_{B \in \rho} (|B| - 1)!. \tag{12}$$

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.



$$\begin{pmatrix} 2 & 4 & 6 & 6 \\ 2 & 6 & 12 & 10 \\ 0 & 0 & 6 & 0 \\ 2 & 8 & 24 & 16 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 2 & 4 & 6 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 & 3 \\ 0 & 1 & 3 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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 , , , .

Lemma 3.3 (Proposition 5.43 in [39]). *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* 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 (8). 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 Surj_S and Sub_S are triangular matrices with non-zero diagonal entries, they are both invertible and so is 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.*
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 α, encoded as a list of pairs (F, α_F) for $F \in \mathrm{supp}(\alpha)$. Let $|\alpha|$ be the description length of α, 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 α 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 α.*

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

We could choose other representations for f, such as coefficient vectors α with $f = \alpha \cdot \mathrm{Sub}$ or $f = \alpha \cdot \mathrm{IndSub}$. However, switching between these representations only adds an overhead of $g(\alpha)$ in the running time, 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 (α, 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 [39, (5.30)]:

$$\mathrm{Hom}(F, G \times X) = \mathrm{Hom}(F, G) \cdot \mathrm{Hom}(F, X)\,. \qquad (13)$$

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 (13) to obtain

$$\sum_H \alpha_H \cdot \mathrm{Hom}(H, G) \cdot \mathrm{Hom}(H, X) = (\alpha \cdot \mathrm{Hom})(G \times X)\,. \qquad (14)$$

For each graph X, the reduction can compute the right side of this linear equation using the oracle, and it can determine the numbers α_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 Hom_S is invertible. Moreover, we have $\alpha \cdot \mathrm{Hom} = \alpha_S \cdot \mathrm{Hom}_S$. We rewrite (14) 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 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 #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 #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 #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 #Hom($\mathcal{A}$) is #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 #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 #Hom($\mathcal{F}$) to #Hom($\mathcal{A}$). Since $\mathcal{F}$ has unbounded treewidth, Theorem 1.8 implies that #Hom($\mathcal{F}$) is #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 #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 #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 (α, 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 α 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 #Hom($\mathcal{F}$) to #Hom($\mathcal{A}$), which proves that #Hom($\mathcal{A}$) is #W[1]-hard when parameterized by $|\alpha|$. Finally, we have $\mathrm{tw}(\alpha) = \mathrm{tw}(F)$, so if #Hom($\mathcal{A}$) can be computed in time $f(\alpha)\cdot n^{o(\mathrm{tw}(\alpha)/\log\mathrm{tw}(\alpha))}$, then #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 #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

$$C(f) = \inf\{\, c \in \mathbb{R} \,:\, f \text{ can be computed in time } O(n^c) \,\}\,. \tag{15}$$

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

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

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

$$C(f) = \max_{F\in\mathrm{supp}(\alpha)} C(\mathrm{Hom}(F,\star))\,, \tag{16}$$

Proposition 1.9 implies that $C(\mathrm{Hom}(F,\star)) \le \mathrm{tw}(F)+1$ holds. Under the assumption FPT ≠ #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 ρ 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 (8) and running an algorithm for counting homomorphisms. More concretely, recall that

$$\mathrm{Sub}(H,G) = \sum_F \mathrm{Surj}^{-1}(H,F)\cdot\mathrm{Hom}(F,G)\,. \tag{17}$$

We know from (12) that $\mathrm{Surj}^{-1}(H,F) \ne 0$ 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 $O(n^c)$. When using the basic treewidth-based algorithm from Proposition 1.6, this running time $O(n^c)$ is governed by the maximum treewidth among the graphs in $\mathrm{Spasm}(H)$.

Theorem 1.1 (restated). There is a deterministic 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. We obtain $\#\mathrm{Sub}(H\to G)$ by evaluating the right side of (17) 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 (17) 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 pattern graphs H: We mentioned in Fact 3.4 that any k-edge graph H only contains graphs with at most k edges in its spasm. Using this fact, we can 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 $\mathrm{tw}(F)\le ck+o(k)$ 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 ξ 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 [20, 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 [48, Corollary 21] prove that $\mathrm{tw}^*(k)\le\frac{13}{75}k+o(k)$ holds. Collecting these two results, we get the following bounds on ξ.

Theorem 4.2 ([20, 48]). *We have $\frac{1}{36}\le\xi\le\frac{13}{75}$.*

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

Corollary 1.2 (restated). There is a deterministic algorithm that is given H and G to compute $\#\mathrm{Sub}(H\to G)$. The algorithm runs in time $f(H)\cdot|V(G)|^{\xi\cdot k+o(k)}$, where $k=|E(H)|$ and $\xi<0.174$ 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, in the full version, we use fast matrix multiplication to count homomorphisms from patterns H of treewidth at most two, thus proving Theorem 1.3.

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 (17), 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, and 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 $\#\mathrm{Sub}(\mathcal{A})$ to compute*

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

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 $\#\mathrm{W}[1]$-hard parameterized by the description length of α, 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 #ETH fails.

Proof. Recall that $\mathrm{Sub} = \mathrm{Surj}^{-1}\,\mathrm{Hom}$ holds by (9), 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 α^H where $\alpha^H_F = [H = F]$ holds for some graph $H \in \mathcal{H}$. That is, α^H contains H with coefficient 1, and no other graphs. Clearly $\#\mathrm{Sub}(\mathcal{A})$ is equivalent to $\#\mathrm{Sub}(\mathcal{H})$.

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 β^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 (12), 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 [15], but that does follow from Theorem 5.2. The following statement was proved recently using a more complicated method.

Corollary 5.3 ([8]). *Given k and G, counting all trees with k vertices in G is $\#\mathrm{W}[1]$-hard when parameterized by 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 α 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 β. To see this, we expand the matrix-vector product:

$$\beta_{F_k} = (\alpha \cdot \mathrm{Surj}^{-1})_{F_k} = \sum_{T \in \mathcal{T}_k} \mathrm{Surj}^{-1}(T, F_k)\,. \tag{18}$$

Recall from (12) 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 (18) 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 α 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 [12] consider the restriction of computing #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 #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 $\mathrm{tw}(\beta) \le t$, then the problem #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 #W[1]-hard parameterized by the description length of α, 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 #ETH fails.

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

On a related note, Jerrum and Meeks [29–31, 41] introduced the following generalization of counting induced subgraphs: Let Φ be some graph property. The problem #IndProp(Φ) is, given a graph G and an integer k, to compute the number of induced k-vertex subgraphs that have property Φ. 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 Φ, Theorem 1.13 immediately implies a complexity dichotomy for these problems.

Corollary 1.11 (restated). Let Φ be any decidable graph property. The problem #IndProp(Φ) 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 #W[1]-hard when parameterized by k.

Finally, let us sketch how to recover the hardness result of Chen, Thurley, and Weyer [12] for counting induced subgraphs from a fixed class $\mathcal{H}$ as a special case of Corollary 1.11: When representing #IndSub$(H \to G)$ for a k-vertex graph H as a linear combination of subgraph numbers #Sub$(H' \to G)$ via (5), 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 (7). When further writing each term #Sub$(H' \to G)$ as a linear combination of homomorphisms #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 #Hom$(K_k \to G)$ in the representation of #IndSub$(H \to G)$ is non-zero. Thus, if $\mathcal{H}$ contains infinitely many graphs, we have unbounded cliques in the homomorphism representation, and the problem of counting induced subgraphs from $\mathcal{H}$ is #W[1]-hard by Corollary 1.11.

6 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 [48] and [20].

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.
[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.
[3] Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.
[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.
[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.
[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.
[7] 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.
[8] 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).
[9] 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.
[10] 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.
[11] Yijia Chen, Martin Grohe, and Bingkai Lin. The hardness of embedding grids and walls. *arXiv preprint arXiv:1703.06423*, 2017.
[12] 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.
[13] 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.
[14] 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.
[15] 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.
[16] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
[17] Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theoretical Computer Science*, 329(1):315–323, 2004.
[18] 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.
[19] 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.
[20] Zdeněk Dvořák and Sergey Norin. Strongly sublinear separators and polynomial expansion. *SIAM Journal on Discrete Mathematics*, 30(2):1095–1101, 2016.

[21] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004.

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

[23] 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.

[24] 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.

[25] 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.

[26] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

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

[28] 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.

[29] Mark Jerrum and Kitty Meeks. The parameterised complexity of counting even and odd induced subgraphs. *CoRR*, abs/1410.3375, 2014.

[30] 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.

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

[32] 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.

[33] 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.

[34] Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. *Information Processing Letters*, 74(3):115–121, 2000.

[35] 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.

[36] 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.

[37] 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.

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

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

[40] Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010.

[41] Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems. *Discrete Applied Mathematics*, 198:170–194, 2016.

[42] 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.

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

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

[50] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[51] Sebastian Wernicke. Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4), 2006.

[52] 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.

[53] 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.

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