A Dichotomy Theorem for Linear Time Homomorphism Orbit Counting in Bounded Degeneracy Graphs

Daniel Paul-Pena
University of California, Santa Cruz, CA, USA

C. Seshadhri
University of California, Santa Cruz, CA, USA

Abstract

Counting the number of homomorphisms of a pattern graph H in a large input graph G is a fundamental problem in computer science. In many applications in databases, bioinformatics, and network science, we need more than just the total count. We wish to compute, for each vertex v of G, the number of H-homomorphisms that v participates in. This problem is referred to as *homomorphism orbit counting*, as it relates to the orbits of vertices of H under its automorphisms.

Given the need for fast algorithms for this problem, we study when near-linear time algorithms are possible. A natural restriction is to assume that the input graph G has bounded degeneracy, a commonly observed property in modern massive networks. Can we characterize the patterns H for which homomorphism orbit counting can be done in near-linear time?

We discover a dichotomy theorem that resolves this problem. For pattern H, let ℓ be the length of the longest induced path between any two vertices of the same orbit (under the automorphisms of H). If $\ell \leq 5$, then H-homomorphism orbit counting can be done in near-linear time for bounded degeneracy graphs. If $\ell > 5$, then (assuming fine-grained complexity conjectures) there is no near-linear time algorithm for this problem. We build on existing work on dichotomy theorems for counting the total H-homomorphism count. Surprisingly, there exist (and we characterize) patterns H for which the total homomorphism count can be computed in near-linear time, but the corresponding orbit counting problem cannot be done in near-linear time.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Homomorphism counting, Bounded degeneracy graphs, Fine-grained complexity, Orbit counting, Subgraph counting

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.54

Related Version *Full Version*: `https://arxiv.org/abs/2211.08605`

Funding Both authors are supported by NSF CCF-1740850, CCF-1839317, CCF-2402572, and DMS-2023495.

1 Introduction

Analyzing the occurrences of a small pattern graph H in a large input graph G is a central problem in computer science. The theoretical study has led to a rich and immensely deep theory [39, 20, 30, 24, 40, 2, 23, 45, 51, 17, 16]. The applications of graph pattern counts occur across numerous scientific areas, including logic, biology, statistical physics, database theory, social sciences, machine learning, and network science [34, 19, 22, 18, 27, 13, 29, 42, 59, 45, 25, 44]. (Refer to the tutorial [53] for more details on applications.)

A common formalism used for graph pattern counting is *homomorphism counting*. The pattern graph is denoted $H = (V(H), E(H))$ and is assumed to have constant size. The input graph is denoted $G = (V(G), E(G))$. Both graphs are simple and do not contain

self-loops. An H-homomorphism is a map $f : V(H) \to V(G)$ that preserves edges. Formally, $\forall (u, v) \in E(H)$, $(f(u), f(v)) \in E(G)$. Let $\mathrm{Hom}_H(G)$ denote the number of distinct H-homomorphisms in G.

Given the importance of graph homomorphism counts, the study of efficient algorithms for this problem is a subfield in itself [35, 3, 18, 27, 26, 24, 13, 23, 14, 51]. The simplest version of this problem is when H is a triangle, itself a problem that attracts much attention. Let $n = |V(G)|$ and $k = |V(H)|$. Computing $\mathrm{Hom}_H(G)$ is $\#W[1]$-hard when parameterized by k (even when H is a k-clique), so we do not expect $n^{o(k)}$ algorithms for general H [24]. Much of the algorithmic study of homomorphism counting is in understanding conditions on H and G when the trivial n^k running time bound can be beaten.

Our work is inspired by the challenges of modern applications of homomorphism counting, especially in network science. Typically, n is extremely large, and only near-linear time $(n \cdot \mathrm{poly}(\log n))$ algorithms are feasible. Inspired by a long history and recent theory on this topic, we focus on *bounded degeneracy* input graphs (we say bounded degeneracy graphs to refer to graphs belonging to classes of graphs with bounded degeneracy). This includes all non-trivial minor-closed graph families, such as planar graphs, bounded genus graphs, and bounded tree-width graphs. Many practical algorithms for large-scale graph pattern counting use algorithms for bounded degeneracy graphs [2, 38, 45, 43, 37, 44]. Real-world graphs typically have a small degeneracy, comparable to their average degree ([32, 37, 55, 5, 9], also Table 2 in [5]).

Secondly, many modern applications for homomorphism counting require more fine-grained statistics than just the global count $\mathrm{Hom}_H(G)$. The aim is to find, *for every vertex v of G*, the number of homomorphisms that v participates in. Seminal work in network analysis for bioinformatics plots the distributions of these per-vertex counts to compare graphs [36, 46]. Orbit counts can be used to generate features for vertices, sometimes called the graphlet kernel [54]. In the past few years, there have been many applications of these per-vertex counts [10, 59, 52, 57, 4, 58, 50, 60, 61].

Algorithms for this problem require considering the "roles" that v could play in a homomorphism. For example, in a 7-path (a path of length 6) there are 4 different roles: a vertex v could be in the middle, could be at the end, or at two other positions. These roles are colored in Fig. 1. The roles are called *orbits* (defined in the Section 3), and the problem of *H-homomorphism orbit counting* is as follows: for every orbit ψ in H and every vertex v in G, output the number of homomorphisms of H where v participates in the orbit ψ. This is the main question addressed by our work:

What are the pattern graphs H for which the H-homomorphism orbit counting problem is computable in near-linear time (when G has bounded degeneracy)?

Recent work of Bressan followed by Bera-Pashanasangi-Seshadhri introduced the question of homomorphism counting for bounded degeneracy graphs, from a fine-grained complexity perspective [14, 8]. A dichotomy theorem for near-linear time counting of $\mathrm{Hom}_H(G)$ was provided in subsequent work [6]. Assuming fine-grained complexity conjectures, $\mathrm{Hom}_H(G)$ can be computed in near-linear time iff the longest induced cycle of H has length at most 5. It is natural to ask whether these results extend to orbit counting.

1.1 Main Result

We begin with some preliminaries. The input graph $G = (V(G), E(G))$ has n vertices and m edges. A central notion in our work is that of graph degeneracy, also called the coloring number.

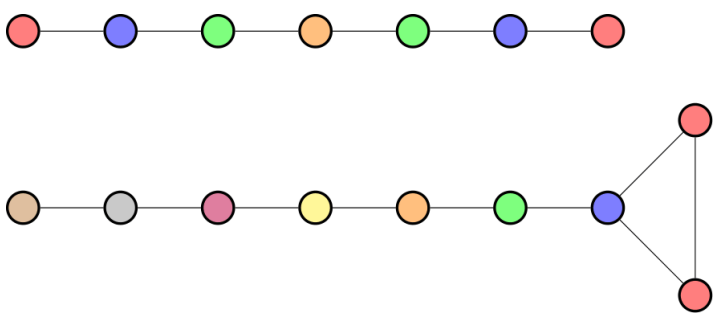

Figure 1 Examples of orbits and LIPCO values. Vertices in the same orbit have the same color. The top graph is the 7-path (a path of length 6). There is an induced path of length 6 between the red vertices, hence the LIPCO of this graph is 6. Theorem 5 implies that we can not compute OrbitHom in near-linear time.
The bottom graph adds a triangle at the end, breaking the symmetry, and the only vertices in the same orbit in that graph are the red ones. The LIPCO in this graph is now less than 6 so we can compute OrbitHom in near-linear time.

▶ **Definition 1.** *A graph G is κ-degenerate if the minimum degree in every subgraph of G is at most κ.*

The degeneracy of G is the minimum value of κ such that G is κ-degenerate.

A family of graphs has *bounded degeneracy* if the degeneracy is constant with respect to the graph size. Bounded degeneracy graph classes are extremely rich. For example, all non-trivial minor-closed families have bounded degeneracy. This includes bounded treewidth graphs. Preferential attachment graphs also have bounded degeneracy; real-world graphs have a small value of degeneracy (often in the 10s) with respect to their size (often in the hundreds of millions) [5].

We assume the pattern graph $H = (V(H), E(H))$ to have a constant number of vertices. (So we suppress any dependencies on purely the size of H.) Consider the group of automorphisms of H. The vertices of H can be partitioned into *orbits*, which consist of vertices that can be mapped to each other by some automorphism (defined formally in Definition 6). For example, in Fig. 1, the 7-path has four different orbits, where each orbit has the same color. The 7-path with a hanging triangle (in Fig. 1) has more orbits, since the pattern is no longer symmetric with respect to the "center" of the 7-path and hence the opposite "ends" of the 7-path cannot be mapped by a non-trivial automorphism.

The set of orbits of the pattern H is denoted $\Psi(H)$. Let $\Phi(H, G)$ be the set of homomorphisms from H to G ($\mathrm{Hom}_H(G) = |\Phi(H, G)|$). We now define our main problem.

▶ **Definition 2.** *Homomorphism Orbit Counts: For each orbit $\psi \in \Psi(H)$ and vertex $v \in V(G)$, define* $\mathrm{OrbitHom}_{H,\psi}(v)$ *to be the number of H-homomorphisms mapping a vertex of ψ to v. Formally,* $\mathrm{OrbitHom}_{H,\psi}(v) = |\{\phi \in \Phi(H, G) \colon \exists h \in \psi,\ \phi(h) = v\}|$.

The problem of H-homomorphism orbit counting *is to output the values* $\mathrm{OrbitHom}_{H,\psi}(v)$ *for all $v \in V(G), \psi \in \Psi(H)$. (Abusing notation,* $\mathrm{OrbitHom}_H(G)$ *refers to the list/vector of all of these values.)*

Note that for a given H, the size of the output is $n|\Psi(H)|$ (recall $n = |V(G)|$). For example, when H is the 7-path, we will get $4n$ counts, for each vertex and each of the four orbits.

Our main result is a dichotomy theorem that precisely characterizes patterns H for which $\mathrm{OrbitHom}_H(G)$ can be computed in near-linear time. We introduce a key definition.

▶ **Definition 3.** *For a pattern H, the* Longest Induced Path Connecting Orbits *of H, denoted $LIPCO(H)$ is defined as follows. It is the length of the longest induced simple path, measured in edges, between any two vertices h, h' in H (where h may be equal to h', forming a cycle) in the same orbit.*

Again refer to Fig. 1. The 7-path has a LIPCO of six, since the ends are in the same orbit. On the other hand, the second pattern (7-path with a triangle) has a LIPCO of 3 due to the triangle.

The Triangle Detection Conjecture was introduced by Abboud and Williams on the complexity of determining whether a graph has a triangle [1]. It is believed that this problem cannot be solved in near-linear time, and indeed, may even require $\Omega(m^{4/3})$ time. We use this conjecture for the lower bound of our main theorem.

▶ **Conjecture 4** (Triangle Detection Conjecture [1])**.** *There exists a constant $\gamma > 0$ such that in the word RAM model of $O(\log n)$ bits, any algorithm to detect whether an input graph on m edges has a triangle requires $\Omega(m^{1+\gamma})$ time in expectation.*

Our main theorem proves that the LIPCO determines the dichotomy. Note that because G is a bounded degeneracy graph we have $m = O(n)$, we will be expressing the bounds in terms of m.

▶ **Theorem 5** (Main Theorem)**.** *Let G be a graph with n vertices, m edges, and bounded degeneracy. Let $\gamma > 0$ denote the constant from the Triangle Detection Conjecture (Conjecture 4).*

- *If $LIPCO(H) \leq 5$: there exists a deterministic algorithm that computes $\mathrm{OrbitHom}_H(G)$ in time $O(m \log n)$.*[1]
- *If $LIPCO(H) > 5$: assume the Triangle Detection Conjecture. There is no algorithm with (expected) running time $O(m^{1+\gamma})$ that computes $\mathrm{OrbitHom}_H(G)$.*

Orbit Counting vs Total Homomorphism Counting

In the following discussion, we use "linear" to actually mean near-linear, we assume that the Triangle Detection Conjecture is true, and we assume that G has bounded degeneracy.

One of the most intriguing aspects of the dichotomy of Theorem 5 is that it differs from the condition for getting the total homomorphism count. As mentioned earlier, the inspiration for Theorem 5 is the analogous result for determining $\mathrm{Hom}_H(G)$. There is a near-linear time algorithm iff the length of the longest induced cycle (LICL) of H is at most five. Since the definition of LIPCO considers induced cycles (induced path between a vertex to itself), if $LIPCO(H) \leq 5$, then $LICL(H) \leq 5$. This implies, not surprisingly, that the total homomorphism counting problem is easier than the orbit counting problem.

But there exist patterns H for which the orbit counting problem is provably harder than total homomorphism counting, a simple example is the 7-path (path with 7 vertices). There is a simple linear time dynamic program for counting the homomorphism of paths. But the endpoints are in same orbit, so the LIPCO is six, and Theorem 5 proves the non-existence of linear time algorithms for orbit counting. On the other hand, the LIPCO of the 6-path is five, so orbit counting can be done in linear time.

Consider the pattern at the bottom of Fig. 1. The LICL is three, so the total homomorphism count can be determined in linear time. Because the ends of the underlying 7-path lie in different orbits, the LIPCO is also three (by the triangle). Theorem 5 provides a linear time algorithm for orbit counting.

[1] The exact dependency on the degeneracy κ of the input graph G is $O\left(\kappa^{|H|-1}\right)$.

1.2 Main Ideas

The starting point for homomorphism counting on bounded degeneracy graphs is the seminal work of Chiba-Nishizeki on using acyclic graph orientations [20]. It is known that, in linear time, the edges of a bounded degeneracy graph can be acyclically oriented while keeping the outdegree bounded [41]. For clique counting, we can now use a brute force algorithm in all out neighborhoods, and get a linear time algorithm. Over the past decade, various researchers observed that this technique can generalize to certain other pattern graphs [21, 45, 43, 44]. Given a pattern H, one can add the homomorphism counts of all acyclic orientations of H for an acyclic orientation of G. In certain circumstances, each acyclic orientation can be efficiently counted by a carefully tailored dynamic program that breaks the oriented H into subgraphs spanned by rooted, directed trees.

Bressan gave a unified treatment of this approach through the notion of *DAG-tree decompositions*. [14] These decompositions give a systematic way of breaking up an oriented pattern into smaller pieces, such that homomorphism counts can be computed by a dynamic program. Bera et al. showed that if the LICL of H is at most 5, then the DAG-treewidth of H is at most one [8, 6]. This immediately implies Bressan's algorithm runs in linear time.

Our result on orbit counting digs deeper into the mechanics of Bressan's algorithm. To run in linear time, Bressan's algorithm requires "compressed" data structures that store information about homomorphism counts. For example, the DAG-tree decomposition based algorithm can count 4-cycles in linear time for bounded degeneracy graphs (this was known from Chiba-Nishizeki as well [20]). But there could exist quadratically many 4-cycles in such a graph. Consider two vertices connected by $\Theta(n)$ disjoint paths of length 2; each pair of paths yields a distinct 4-cycle. Any linear time algorithm for 4-cycle counting has to carefully index directed paths and combine these counts, without actually touching every 4-cycle.

By carefully looking at Bressan's algorithm, we discover that "local" per-vertex information about H-homomorphisms can be computed. Using the DAG-tree decomposition, one can combine these counts into a quantity that looks like orbit counts. Unfortunately, we cannot get exact orbit counts, but rather a weighted sum of homomorphisms.

To extract exact orbit counts, we dig deeper into the relationship between orbit counts and per-vertex homomorphism counts. This requires looking into the behavior of independent sets in the orbits of H. We then design an inclusion-exclusion formula that "inverts" the per-vertex homomorpishm counts into orbit counts. The formula requires orbit counts for other patterns H' that are constructed by merging independent sets in the same orbit of H.

Based on previous results, we can prove that if the LICL of all these H' patterns is at most 5, then $\text{OrbitHom}_H(G)$ can be computed in (near)linear time. This LICL condition over all H' is equivalent to the LIPCO of H being at most 5. Achieving the upper bound of Theorem 5.

The above seemingly ad hoc algorithm optimally characterizes when orbit counting is linear time computable. To prove the matching lower bound, we use tools from the breakthrough work of Curticapean-Dell-Marx [23]. They prove that the complexity of counting linear combinations of homomorphism counts is determined by the hardest individual count (up to polynomial factors). Gishboliner-Levanzov-Shapira give a version of this tool for proving linear time hardness [31]. Consider a pattern H with LIPCO at least six. We can construct a pattern H' with LICL at least six by merging vertices of an orbit in H. We use the tools above to construct a constant number of linear sized graphs $G_1, G_2, \ldots, G_k$ such that a linear combination of H-orbit counts on these graphs yields the total H'-homomorphism count on G. The latter problem is hard by existing bounds, and hence the hardness bounds translate to H-orbit homomorphism counting.

2 Related Work

Homomorphism and subgraph counting on graphs is an immense topic with an extensive literature in theory and practice. For a detailed discussion of practical applications, we refer the reader to a tutorial [53].

Homomorphism counting is intimately connected with the treewidth of the pattern H. The notion of tree decomposition and treewidth were introduced in a seminal work by Robertson and Seymour [47, 48, 49]; although it has been discovered before under different names [11, 33]. A classic result of Dalmau and Jonsson [24] proved that $\mathrm{Hom}_H(G)$ is polynomial time solvable if and only if H has bounded treewidth, otherwise it is $\#W[1]$-complete. Díaz et al [26] gave an algorithm for homomorphism counting with runtime $O(2^k n^{t(H)+1})$ where $t(H)$ is the treewidth of the pattern graph H and k the number of vertices of H.

To improve on these bounds, recent work has focused on restrictions on the input G [51]. A natural restriction is bounded degeneracy, which is a nuanced measure of sparsity introduced by early work of Szekeres-Wilf [56]. Many algorithmic results exploit low degeneracy for faster subgraph counting problems [20, 28, 2, 38, 45, 43, 37, 44].

Pioneering work of Bressan introduced the concept of DAG-treewidth for faster algorithms for homomorphism counting in bounded degeneracy graphs [14]. Bressan gave an algorithm for counting $\mathrm{Hom}_H(G)$ running in time essentially $m^{\tau(H)}$, where τ denotes the DAG-treewidth. The result also proves that (assuming ETH) there is no algorithm running in time $m^{o(\tau(H)/\log \tau(H))}$.

Bera-Pashanasangi-Seshadhri build on Bressan's methods to discover a dichotomy theorem for linear time homomorphism counting in bounded degeneracy graphs [7, 8]. Gishboliner, Levanzov, and Shapira independently proved the same characterization using slightly different methods [31, 6].

We give a short discussion of the Triangle Detection Conjecture. Itai and Rodeh [35] gave the first non-trivial algorithm for the triangle detection and finding problem with $O(m^{3/2})$ runtime. The current best known algorithm runs in time $O(\min\{n^\omega, m^{2\omega/(\omega+1)}\})$ [3], where ω is the matrix multiplication exponent. Even for $\omega = 2$, the bound is $m^{4/3}$ and widely believed to be a lower bound. Many classic graph problems have fine-grained complexity hardness based on Triangle Detection Conjecture [1].

Homomorphism or subgraph orbit counts have found significant use in network analysis and machine learning. Przulj introduced the use of graphlet (or orbit count) degree distributions in bioinformatics [46]. The graphlet kernel of Shervashidze-Vishwanathan-Petri-Mehlhorn-Borgwardt uses vertex orbits counts to get embeddings of vertices in a network [54]. Four vertex subgraph and large cycle and clique orbit counts have been used for discovering special kinds of vertices and edges [59, 50, 60, 61]. Orbits counts have been used to design faster algorithms for finding dense subgraphs in practice [10, 52, 57, 4, 58].

3 Preliminaries

We use G to denote the input graph and H to denote the pattern graph, both $G = (V(G), E(G))$ and $H = (V(H), E(H))$ are simple, undirected and connected graphs. We denote $|V(G)|$ and $|E(G)|$ by n and m respectively and $|V(H)|$ by k.

A pattern graph H is divided into orbits, we use the definition from Bondy and Murty (Chapter 1, Section 2 [12]):

▶ **Definition 6.** *Fix a graph $H = (V(H), E(H))$. An automorphism is a bijection $\sigma : V(H) \to V(H)$ such that $(u, v) \in E(H)$ iff $(\sigma(u), \sigma(v)) \in E(H)$. The group of automorphisms of H is denoted* $\mathrm{Aut}(H)$.

Define an equivalence relation on $V(H)$ as follows. We say that $u \sim v$ $(u, v \in V(H))$ iff there exists an automorphism that maps u to v. The equivalence classes of the relation are called orbits.

We refer to the set of orbits in H as $\Psi(H)$ and to individual orbits in $\Psi(H)$ as ψ. Note that every vertex $h \in V(H)$ belongs to exactly one orbit. We can represent an orbit by a canonical (say lexicographically least) vertex in the orbit. Somewhat abusing notation, we can think of the set of orbits as a subset of vertices of H, where each vertex plays a "distinct role" in H. Fig. 1 has examples of different graphs with their separate orbits.

We now define homomorphisms.

▶ **Definition 7.** *An* H-homomorphism *from H to G is a mapping $\phi : V(H) \to V(G)$ such that for all $(u, v) \in E(H)$, $(\phi(u), \phi(v)) \in E(G)$. We refer to the set of homomorphisms from H to G as $\Phi(H, G)$.*

We now define a series of counts.

- $\mathrm{Hom}_H(G)$: This is the count of H-homomorphisms in G. So $\mathrm{Hom}_H(G) = |\Phi(H, G)|$.
- $\mathrm{OrbitHom}_{H,\psi}(v)$: For a vertex $v \in V(G)$, $\mathrm{OrbitHom}_{H,\psi}(v)$ is the number of H-homomorphisms that map any vertex in the orbit ψ to v. Formally, $\mathrm{OrbitHom}_{H,\psi}(v) = |\{\phi \in \Phi(H, G) \colon \exists u \in \psi,\, \phi(u) = v\}|$.
- $\mathrm{OrbitHom}_{H,\psi}(G), \mathrm{OrbitHom}_H(G)$: We use $\mathrm{OrbitHom}_{H,\psi}(G)$ to denote the list/vector of counts $\{\mathrm{OrbitHom}_{H,\psi}(v)\}$ over all $v \in V(G)$. Similarly, $\mathrm{OrbitHom}_H(G)$ denotes the sequence of lists of counts $\mathrm{OrbitHom}_{H,\psi}(G)$ over all orbits ψ.

Our aim is to compute $\mathrm{OrbitHom}_H(G)$, which are a set of homomorphism counts. We use existing algorithmic machinery to compute homomorphism counts per vertex of H, so part of our analysis will consist of figuring out how to go between these counts. As we will see, this is where the LIPCO parameter makes an appearance.

Acyclic orientations. These are a key algorithmic tool in efficient algorithms for bounded degeneracy graphs. An acyclic orientation of an undirected graph G is a digraph obtained by directing the edges of G such that the digraph is a DAG. We will encapsulate the application of the degeneracy in the following lemma, which holds from a classic result of Matula and Beck [41].

▶ **Lemma 8.** *Suppose G has degeneracy κ. Then, in $O(m + n)$ time, one can compute an acyclic orientation $G^{\rightarrow}$ of G with the following property. The maximum outdegree of $G^{\rightarrow}$ is precisely κ. ($G^{\rightarrow}$ is also called a* degeneracy orientation*.)*

The set of all acyclic orientations of H is denoted $\Sigma(H)$. Our algorithm will enumerate over all such orientations.

Note that all definitions of homomorphisms carry over to DAGs.

3.1 DAG-tree decompositions

A central part of our result is applying intermediate lemmas from an important algorithm of Bressan for homomorphism counting [14]. This subsection gives a technical overview of Bressan's techique of DAG-tree decompositions and related lemmas. Our aim is to state the key lemmas from previous work that can be used as a blackbox.

The setting is as follows. We have an acyclic orientation $G^{\rightarrow}$ and a DAG pattern P (think of P as a member of $\Sigma(H)$; P is an acyclic orientation of H). Bressan's algorithm gives a dynamic programming approach to counting $\Phi(P, G^{\rightarrow})$.

We introduce some notation. We use the standard notion of reachability in digraphs: vertex v is reachable from u if there is a directed path from u to v.

- S: The set of sources in the DAG P.
- $Reach_P(s)$: For source $s \in S$, $Reach_P(s)$ is the set of vertices in P reachable from s.
- $Reach_P(B)$: Let $B \subseteq S$. $Reach_P(B) = \bigcup_{s \in B} Reach_P(s)$.
- $P[B]$: This is the subgraph of P induced by $Reach_P(B)$.

▶ **Definition 9** (DAG-tree decomposition [14]). *Let P be a DAG with source vertices S. A DAG-tree decomposition of P is a tree $T = (\mathcal{B}, \mathcal{E})$ with the following three properties:*

1. *Each node $B \in \mathcal{B}$ (referred to as a "bag" of sources) is a subset of the source vertices S: $B \subseteq S$.*
2. *The union of the nodes in T is the entire set S: $\bigcup_{B \in \mathcal{B}} B = S$.*
3. *For all $B, B_1, B_2 \in \mathcal{B}$, if B lies on the unique path between the nodes B_1 and B_2 in T, then $Reach(B_1) \cap Reach(B_2) \subseteq Reach(B)$.*

▶ **Definition 10.** *Let P be a DAG. For any DAG-tree decomposition T to P, the DAG-treewidth $\tau(T)$ is defined as $\max_{B \in \mathcal{B}} |B|$. The DAG-treewidth of P, denoted $\tau(P)$, is the minimum value of $\tau(T)$ over all DAG-tree decompositions T of P.*

Two important lemmas. We state two critical results from previous work. Both of these are highly non-trivial and technical to prove. We will use them in a black-box manner. The first lemma, by Bera-Pashanasangi-Seshadhri, connects the Largest Induced Cycle Length (LICL) to DAG-treewidth [8].

▶ **Lemma 11** (Theorem 4.1 in [8]). *For a simple graph H: $LICL(H) \leq 5$ iff $\forall P \in \Sigma(H), \tau(P) = 1$.*

The second lemma is an intermediate property of Bressan's subgraph counting algorithm [15]. We begin by defining homomorphism extensions. Think of some directed pattern P that we are trying to count. Fix a (rooted) DAG-tree decomposition T. Let P' be a subgraph of P, P'' be a subgraph of P'. A P'-homomorphism ϕ' *extends* a P''-homomorphism ϕ'' if $\forall v \in V(P'')$, $\phi'(v) = \phi''(v)$. Basically, ϕ' agrees with ϕ'' wherever the latter is defined.

- $\text{ext}(P', G; \phi)$: Let ϕ be a homomorphism from a subgraph of P' to G. Then $\text{ext}(P', G; \phi)$ is the number of P'-homomorphisms extending ϕ.
- $P[\text{down}(B)]$: Let B be a node in the DAG-tree decomposition T of P. The set $\text{down}(B)$ is the union of bags that are descendants of B in T. Furthermore $P[\text{down}(B)]$ is the pattern induced by $Reach(\text{down}(B))$.

A technical lemma in Bressan's result shows that extension counts can be obtained efficiently. We will refer to the procedure in this lemma as "Bressan's algorithm".

▶ **Lemma 12** (Lemma 5 in [15]). *Let $G^{\rightarrow}$ be a digraph with outdegree at most d and P be a DAG with k vertices. Let $T = (\mathcal{B}, \mathcal{E})$ be a DAG-tree decomposition for P, and B any element of $\mathcal{B}$. There is a procedure, that in time $O(|\mathcal{B}| poly(k) d^{k-\tau(T)} n^{\tau(T)} \log n)$, returns a dictionary storing the following values: for every $\phi : P[B] \to G^{\rightarrow}$, it has $\text{ext}(P[\text{down}(B)], G; \phi)$.*

Let us explain this lemma in words. For any bag B, which is a set of sources in P, consider $P[B]$, which is the subgraph induced by $Reach_P(B)$. For every $P[B]$-homomorphism ϕ, we wish to count the number of extensions to $P[\text{down}(B)]$ (the subgraph induced by vertices of P reachable by any source in any descendant bag of B).

4 Obtaining Vertex-Centric Counts

We define vertex-centric homomorphism counts, which allows us to ignore orbits and symmetries in H. Quite simply, for vertices $h \in V(H)$ and $v \in V(G)$, we count the number of homomorphisms from H to G that map h to v.

▶ **Definition 13.** *Vertex-centric Counts: For each vertex $h \in V(H)$ and vertex $v \in V(G)$, let* $\text{VertexHom}_{H,h}(v)$ *be the number of H-homomorphisms that map h to v.*

Let $\text{VertexHom}_H(G)$ *denote the list of* $\text{VertexHom}_{H,h}(v)$ *over all $h \in V(H)$ and $v \in V(G)$.*

We can show that the vertex-centric counts can be obtained in near-linear time when $LICL(H) \leq 5$:

▶ **Theorem 14.** *There is an algorithm that takes as input a bounded degeneracy graph G and a pattern H with $LICL(H) \leq 5$, and has the following properties. It outputs* $\text{VertexHom}_H(G)$ *and runs in $O(n \log n)$ time.*

Before proving this theorem we need to introduce two more lemmas. First, we invoke the following lemma from [15]:

▶ **Lemma 15** (Lemma 4 in [15])**.** *Given any $B \subseteq S$, the set of homomorphisms $\Phi(P[b], G^{\rightarrow})$ has size $O(d^{k-|B|}n^{|B|})$ and can be enumerated in time $O(k^2 d^{k-|B|}n^{|B|})$.*

Second, we show how to use the output of Bressan's algorithm to obtain the Vertex-centric counts:

▶ **Lemma 16.** *Let P be a directed pattern on k vertices, $T = (\mathcal{B}, \mathcal{E})$ be a DAG-tree decomposition of P with $\tau(P) = 1$ (All nodes/bags in T are singletons), and $G^{\rightarrow}$ be a directed graph with n vertices and max degree d. Let b be the root of T and h be any vertex in $P[b]$. We can compute* $\text{VertexHom}_{P,h}(v)$ *in time $O(poly(k)d^{k-1}n \log n)$.*

Proof. The algorithm of Lemma 12 will return a data structure/dictionary that gives the following values. For each $\phi : P[b] \to G^{\rightarrow}$, it provides $\text{ext}(P[\text{down}(b)], G; \phi)$. Note that b is the root of T. By the properties of a DAG-tree decomposition, $\text{down}(b)$ contains all vertices of P and $P[\text{down}(b)] = P$. Hence, the dictionary gives the values $\text{ext}(P, G^{\rightarrow}; \phi)$, that is, the number of homomorphisms $\phi' : P \to G^{\rightarrow}$ that extend ϕ.

Let h be a vertex in $P[b]$. We can partition the set of homomorphisms from $P[b]$ to $G^{\rightarrow}$, $\Phi(P[b], G^{\rightarrow})$, into sets $\Phi_{b,h,v}$ defined as follows. For each $v \in V(G)$, $\Phi_{b,h,v} := \{\phi \in \Phi(P[b], G^{\rightarrow}) : \phi(h) = v\}$.

By Lemma 15 we can list all the homomorphisms $\Phi(P[b], G^{\rightarrow})$ in $O(k^2 d^{k-1}n)$ time, by the same lemma we know that $\Phi(P[b], G^{\rightarrow})$ will have size at most $O(d^{k-1}n)$, hence we can iterate over the list of homomorphisms and check the value of $\phi(h)$. We can then express $\text{VertexHom}_P(G^{\rightarrow})$ as follows:

$$\begin{aligned}\text{VertexHom}_{P,h}(v) &= |\{\phi' \in \Phi(P, G^{\rightarrow}) : \phi'(h) = v\}| \\ &= \sum_{\phi \in \Phi(P[b], G^{\rightarrow}) : \phi(h) = v} ext(P, G^{\rightarrow}; \phi) \\ &= \sum_{\phi \in \Phi_{b,h,v} : \phi(h) = v} ext(P, G^{\rightarrow}; \phi)\end{aligned}$$

We can compute all of these values by enumerating all the elements in $\phi \in \Phi_{b,h,v}$ (over all v), and making a dictionary access to get $ext(P, G^{\rightarrow}; \phi)$. The total running time is $O(k^2 d^{k-1} n \log n)$, where $\log n$ is extra overhead of accessing the dictionary.

By Lemma 12, the dictionary construction takes $O(|\mathcal{B}| poly(k)\ d^{k-\tau(T)} n^{\tau(T)} \log n)$ time. Since $\tau(T) = 1$ and $|\mathcal{B}| = O(k)$, we can express the total complexity as $O(poly(k) d^{k-1} n \log n)$. ◀

We can now complete the proof of Theorem 14:

Proof of Theorem 14. The first step of our algorithm is to construct the degeneracy orientation $G^{\rightarrow}$ of G. By Lemma 8, it can be computed in $O(m+n)$ time. Since G has bounded degeneracy, $G^{\rightarrow}$ has bounded outdegree. When orienting G as $G^{\rightarrow}$, each homomorphism from H to G becomes a homomorphism of exactly one of the directed patterns $P \in \Sigma(H)$ to $G^{\rightarrow}$. We can hence compute $\text{VertexHom}_H(G)$ as the sum of $\text{VertexHom}_P(G^{\rightarrow})$ for every acyclic orientation of H. This is given by the following equation:

$$\text{VertexHom}_H(G) = \sum_{P \in \Sigma(H)} \text{VertexHom}_P(G^{\rightarrow}) \tag{1}$$

Because $LICL(H) \leq 5$, Lemma 11 implies that for all $P \in \Sigma(H)$, $\tau(H) = 1$. There exists a DAG-tree decomposition $T = (\mathcal{B}, \mathcal{E})$ of P with $\tau(T) = 1$. We use the output of Bressan's algorithm to obtain the Vertex-centric counts.

The DAG-tree decomposition T can be arbitrarily rooted at any node b. Moreover, for each $h \in V(P)$, there must exist some source b such that $h \in P[b]$ (meaning, h is reachable from b). So, by rooting T at all possible nodes (singleton bags), we can ensure that h is in $P[b]$. We can apply Lemma 16 to get all counts $\text{VertexHom}_{P,h}(v)$.

We complete the proof by bounding the running time and asserting correctness.

From Lemma 8, we can compute $G^{\rightarrow}$ in $O(m+n)$. Since G has bounded degeneracy, $m = O(n)$ and the outdegree d is bounded. The number of acyclic orientations of H, $|\Sigma(H)|$ is bounded by $O(k!)$. In each iteration, by Lemma 16, we will take $O(poly(k) d^{k-1} n \log n)$. For constant k and constant d, the running time is $O(n \log n)$.

Now we prove the correctness of the algorithm. Consider each $P \in \Sigma(H)$. Let $T = (\mathcal{B}, \mathcal{E})$ be the DAG-tree decomposition of P. For each $b \in \mathcal{B}$, we compute $\text{VertexHom}_{P,h}(G^{\rightarrow})$ for all the vertices in $h \in P[b]$. By looping over each singleton bag b, we update counts for all vertices in P. Hence, we are computing $\text{VertexHom}_P(G^{\rightarrow})$. Finally, we sum over all $P \in \Sigma(H)$, which by Equation 1, gives us $\text{VertexHom}_H(G)$. ◀

5 From Vertex-Centric to Orbit Counts

We now show how to go from vertex-centric to orbit counts, using inclusion-exclusion. Much of our insights are given by the following definitions.

▶ **Definition 17.** *$\mathcal{IS}(\psi)$: Given a pattern graph H, for every orbit $\psi \in \Psi(H)$ we define $\mathcal{IS}(\psi)$ as the collection of all non empty subsets $S \subseteq \psi$, such that S forms an independent set (i.e. there is no edge in $E(H)$ connecting any two vertices in S).*

Formally, $\mathcal{IS}(\psi) = \{S \subseteq \psi, S \neq \emptyset : \forall\ h, h' \in S, (h, h') \notin E(H)\}$.

▶ **Definition 18.** *H_S: For each set $S \in \mathcal{IS}(\psi)$ we define H_S as the graph resulting from merging all the vertices in S into a single new vertex h_S, removing any duplicate edge.*

We state two more tools in our analysis. The first lemma relates the counts obtained in the previous section ($\text{VertexHom}_{H_S}(G)$) to the desired output ($\text{OrbitHom}_H(G)$).

▶ **Lemma 19** (Inclusion-exclusion formula)**.**

$$\text{OrbitHom}_{H,\psi}(v) = \sum_{S \in \mathcal{IS}(\psi)} (-1)^{|S|+1} \text{VertexHom}_{H_S,h_S}(v)$$

In order to prove this lemma, we need to define the Signature of a homomorphisms. Let ϕ be a homomorphism from H to G, we define $\text{Sig}(\phi, \psi, v)$ to be the subset of vertices from the orbit ψ that are mapped to v in ϕ. Formally $\text{Sig}(\phi, \psi, h) = \{h \in \psi : \phi(h) = v\}$.

We prove a series of claims regarding the signature.

▷ Claim 20. The Signature of ϕ from ψ to v, $\text{Sig}(\phi, \psi, v)$, must form an independent set of vertices in $V(H)$, that is, there are no edges in $E(H)$ connecting two vertices in $\text{Sig}(\phi, \psi, v)$.

Proof. We prove by contradiction. Assume that $S = \text{Sig}(\phi, \psi, v)$ is not an Independent Set of vertices of $V(H)$, that means that we have a pair of vertices $h, h' \in S$ such that there is an edge connecting them. But from the definition of signature we have that $\phi(h) = \phi(h') = v$, however this is not a valid homomorphism from H to G as it is not preserving the (h, h') edge. ◁

The next claim allows us to relate the Signature with the Homomorphism Orbit Counts.

▷ Claim 21.

$$\text{OrbitHom}_{H,\psi}(v) = \sum_{S \in \mathcal{IS}(\psi)} |\{\phi \in \Phi(H, G) : S = \text{Sig}(\phi, \psi, v)\}|$$

Proof. From the definition of Homomorphism Orbit Counts we have that $\text{OrbitHom}_{H,\psi}(v) = |\{\phi \in \Phi(H, G) \ : \exists h \in \psi, \ \ \phi(h) = v\}|$. Hence, suffices to show that $|\{\phi \in \Phi(H, G) \ : \exists h \in \psi, \ \ \phi(h) = v\}| = \sum_{S \in \mathcal{IS}(\psi)} |\{\phi \in \Phi(H, G) : S = \text{Sig}(\phi, \psi, v)\}|$.

Let $\phi \in \Phi(H, G)$ be a homomorphism from H to G such that $\exists h \in \psi, \phi(h) = v$. Let $S = \text{Sig}(\phi, \psi, v)$, we know that $S \neq \emptyset$ as h is mapped to v and from Claim 20 we know that it forms an independent set on the vertices of H. Hence $S \in \mathcal{IS}(\psi)$.

To prove the other direction of the equality, suffices to note that if a homomorphism ϕ contributes to the right side of the equation, then its signature S belongs to $\mathcal{IS}(\psi)$, hence there is at least one vertex $h \in V(H)$ that is mapped to v, and thus ϕ contributes to the left side of the equation. ◁

Now, we will relate the Signature with the Vertex-centric Counts:

▷ Claim 22. For each orbit ψ in H and each vertex v in $V(G)$ we have that $\forall\ S \in \mathcal{IS}(\psi)$:

$$|\phi \in \Phi(H, G) : \forall h \in S, \phi(h) = v| = \sum_{\substack{S' \supseteq S \\ S' \in \mathcal{IS}(\psi)}} |\phi : \text{Sig}(\phi, \psi, v) = S'|$$

Proof. If ϕ is mapping all the vertices in S to v, then the Signature of ϕ from ψ to v must be a superset of S, $\text{Sig}(\phi, \psi, v) \supseteq S$. Hence summing over such sets will reach the equality. Note that we can add the restriction of S' belonging to $\mathcal{IS}(\psi)$ as it is implied from Claim 20. ◁

Let $\Phi' = \Phi(H_S, G)$ be the set of homomorphism from H_S to G. When S forms an independent set there is an equivalence between the homomorphisms in Φ' that map h_S to v and the set of homomorphisms in $\Phi(H, G)$ that map all the vertices of S to v. In fact we can prove the following claim:

▷ Claim 23. If S is not empty and form an independent set:

$$|\phi \in \Phi(H,G) :\ \forall\ h \in S\ \phi(h) = v| = \text{VertexHom}_{H_S,h_S}(v)$$

Proof. From the definition of Vertex-centric Counts we have that $\text{VertexHom}_{H_S,h_S}(v) = |\phi' \in \Phi(H_S,G) : \phi(h_S) = v|$. Hence it suffices to show that:

$$|\phi \in \Phi(H,G) : \forall h \in S\ \phi(h) = v| = |\phi' \in \Phi(H_S,G) : \phi(h_S) = v|$$

We do so by proving that there is a bijection between both sets, that is, a one to one correspondence between them. Let $\Phi_S = \{\phi' \in \Phi(H_S,G) : \phi(h_S) = v\}$ and $\Phi'_S = \{\phi \in \Phi(H,G) : \forall h \in S, \phi(h) = v\}$. We show an invertible function $f : \Phi_S \to \Phi'_S$:

- Given a homomorphism $\phi \in \Phi_S$ we obtain $\phi' = f(\phi) \in \Phi'_S$ by setting $\phi'(h) = \phi(h)\ \forall\ h \in H \setminus S$ and $\phi'(h_S) = v$. This is a valid homomorphism as we are mapping all the vertices in H_S to G and we are preserving the edges.
- Given a homomorphism $\phi' \in \Phi'_S$ we obtain $\phi = f'(\phi') \in \Phi_S$ by setting $\phi(h) = \phi'(h)\ \forall\ h \in H \setminus S$ and $\phi(h) = v\ \forall\ h \in S$. Again this is a valid homomorphism as we are mapping all the vertices in H to G and we are still preserving the edges.

Additionally, we have that for all $\phi \in \Phi_S$, $\phi = f'(f(\phi))$, which completes the proof. ◁

We will show one last claim that will be important when deriving the inclusion-exclusion formula:

▷ Claim 24. Given a graph H, for every orbit $\psi \in \Psi(H)$, any subset $S' \in \mathcal{IS}(\psi)$ satisfies:

$$\sum_{\substack{S \subseteq S' \\ S \neq \emptyset}} (-1)^{|S|+1} = 1$$

Proof.

$$\sum_{\substack{S \subseteq S' \\ S \neq \emptyset}} (-1)^{|S|+1} = \sum_{i=1}^{|S'|} \binom{|S'|}{i} (-1)^{i+1}$$

$$= \sum_{i=1}^{|S'|} \left(\binom{|S'|-1}{i-1} + \binom{|S'|-1}{i} \right) (-1)^{i+1} = \binom{|S'-1|}{0} (-1)^2 = 1$$ ◁

We now have all the tools required to prove Lemma 19:

Proof of Lemma 19.

$$\sum_{S \in \mathcal{IS}(\psi)} (-1)^{|S|+1} \text{VertexHom}_{H_S,h_S}(v)$$

$$= \sum_{S \in \mathcal{IS}(\psi)} (-1)^{|S|+1} |\phi \in \Phi(H,G) :\ \forall\ h \in S\ \phi(u) = v| \qquad \text{(Claim 23)}$$

$$= \sum_{S \in \mathcal{IS}(\psi)} (-1)^{|S|+1} \sum_{\substack{S' \supseteq S \\ S' \in \mathcal{IS}(\psi)}} |\phi : \text{Sig}(\phi,\psi,v) = S'| \qquad \text{(Claim 22)}$$

$$= \sum_{S \in \mathcal{IS}(\psi)} \sum_{\substack{S' \supseteq S \\ S' \in \mathcal{IS}(\psi)}} (-1)^{|S|+1} |\phi : \text{Sig}(\phi,\psi,v) = S'| \qquad \text{(Factor in)}$$

$$= \sum_{S' \in \mathcal{IS}(\psi)} \sum_{\substack{S \subseteq S' \\ S \neq \emptyset}} (-1)^{|S|+1} |\phi : \mathrm{Sig}(\phi, \psi, v) = S'| \qquad \text{(Reorder)}$$

$$= \sum_{S' \in \mathcal{IS}(\psi)} |\phi : \mathrm{Sig}(\phi, \psi, v) = S'| \sum_{\substack{S \subseteq S' \\ S \neq \emptyset}} (-1)^{|S|+1} \qquad \text{(Factor out)}$$

$$= \sum_{S' \in \mathcal{IS}(\psi)} |\phi : \mathrm{Sig}(\phi, \psi, v) = S'| \qquad \text{(Claim 24)}$$

$$= \mathrm{OrbitHom}_{H,\psi}(v) \qquad \text{(Claim 21)}$$ ◀

The next lemma relates the Longest Induced Path Connecting Orbits (LIPCO) defined in Definition 3 with the $LICL$ of all the graphs H_S, for all $S \in \mathcal{IS}(\psi)$ and all orbits ψ of H.

▶ **Lemma 25.** *For every graph H, $LIPCO(H) \leq 5$ iff $\forall \psi \in \Psi(H), \forall S \in \mathcal{IS}(\psi)$, $LICL(H_S) \leq 5$.*

Proof. First, we show that if $LIPCO(H) > 5$ then $\exists \psi \in \Psi(H), \exists S \in \mathcal{IS}(\psi), LICL(H_S) > 5$. Consider the longest induced path in H with endpoints in the same orbit $\psi \in \Psi(H)$, let h, h' be the two endpoints of the path. We have two cases:

- $h = h'$: In this case the induced path is actually just an induced cycle of length 6 or more in H including the vertex h. For any ψ and for any $S \subseteq \psi$ with $|S| = 1$ we have that $H_S = H$, and hence $LICL(H_S) > 5$.
- $h \neq h'$: In the other case we have that h, h' are distinct vertices. Consider the set $S = \{h, h'\}$, we have that $S \in \mathcal{IS}(\psi)$ as both $h, h' \in \psi$ and there is no edge connecting them (otherwise we would have a longer induced cycle). We form H_S by combining h and h' into a single vertex, the induced path that we had in H becomes then an induced cycle of length at least 6, which implies $LICL(H_S) > 5$.

Now, we prove that if $\exists \psi \in \Psi(H), \exists S \in \mathcal{IS}(\psi), LICL(H_S) > 5$ then $LIPCO(H) > 5$. Let S be the set such that $LICL(H_S) > 5$. Again, we have two cases:

- $|S| = 1$: We have that $H_S = H$ and hence $LICL(H) > 5$, any vertex in that induced cycle induces a path of the same length with such vertex in both ends, which implies $LIPCO(H) > 5$.
- $|S| > 1$: Let h_S be the vertex in H_S obtained by merging the vertices of S in H. Consider the longest induced cycle in H_S, if that cycle does not contain h_S then that same cycle exists in H and $LICL(H) > 5$, which implies $LIPCO(H) > 5$. Otherwise, we can obtain H by splitting h_S back into separate vertices, there will be two distinct vertices $h, h' \in S$ that are in the two ends of an induced path of the same length in H, thus $LIPCO(H) > 5$. ◀

6 Wrapping it up

In this section we complete the proof of the main theorem for the upper bound. We also give Algorithm 1, which summarizes the entire process.

▶ **Theorem 26.** *There is an algorithm that, given a bounded degeneracy graph G and pattern H with $LIPCO(H) \leq 5$, computes* $\mathrm{OrbitHom}_H(G)$ *in time $O(n \log n)$.*

Proof. Because we have that $LIPCO(H) \leq 5$, using Lemma 25 we get that $\forall \psi \in \Psi(H), \forall S \in \mathcal{IS}(\psi), LICL(H_S) \leq 5$. This means, using Theorem 14, that $\forall \psi \in \Psi(H), \forall S \in \mathcal{IS}(\psi)$ we can compute $\mathrm{VertexHom}_{H_S}(G)$ in time $f(k)O(n \log n)$.

Using Lemma 19 we can compute $\text{OrbitHom}_H(G)$ from the individual counts of $\text{VertexHom}_{H_S}(G)$ (as shown in Algorithm 1), we have at most 2^k sets S, hence the total time complexity necessary to compute $\text{OrbitHom}_H(G)$ is $O(n \log n)$. ◀

Algorithm 1 Homomorphism Orbit Counts $\text{OrbitHom}_H(G)$.

```
for each ψ ∈ Ψ(H) do
    for S ∈ IS(ψ) do
        Compute VertexHom_{H_S,h_S}(G)
    end for
    OrbitHom_{H,ψ}(G) = Σ_{S∈IS(ψ)} (−1)^{|S|+1} VertexHom_{H_S,h_S}(v)
end for
Return OrbitHom_H(G)
```

7 Lower Bound for computing Homomorphism Orbit Counts

In this section we prove the lower bound of Theorem 5. It will be given by the following theorem:

▶ **Theorem 27.** *Let H be a pattern graph on k vertices with $LIPCO(H) > 5$. Assuming the Triangle Detection Conjecture, there exists an absolute constant $\gamma > 0$ such that for any function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, there is no (expected) $f(\kappa, k)O(m^{1+\gamma})$ algorithm for the* OrbitHom *problem, where m and κ are the number of edges and the degeneracy of the input graph, respectively.*

To prove this Theorem we will show how to express the Homomorphism Orbit Counts for some orbit ψ as a linear combination of Homomorphism counts of non-isomorphic graphs H_S for all S in $\mathcal{IS}(\psi)$. Because $LIPCO(H) > 5$ we will have that the $LICL$ of at least one of these graphs is also greater than 5. We will then show that the hardness of computing Orbit counts in the original graph is the same than the hardness of computing the Homomorphisms counts. Finally we use a previous hardness result from [8] to complete the proof.

First, we introduce the following definition:

▶ **Definition 28.** *Given a pattern graph H and an input graph G, for the orbit ψ of H, we define $Agg(H, G, \psi)$ as the sum over every vertex $v \in V(G)$ of homomorphisms that are mapping some vertex in ψ to v, that is:*

$$Agg(H, G, \psi) = \sum_{v \in V(G)} \text{OrbitHom}_{H,\psi}(v)$$

Note that if we can compute $\text{OrbitHom}_{H,\psi}(v)$ for every vertex v in G then we can also compute $Agg(H, G, \psi)$ in additional linear time. Now, we state the following lemma:

▶ **Lemma 29.** *For every pattern graph H and every orbit $\psi \in \Psi(H)$, there is some number $l = l(H)$ such that the following holds. For every graph G there are some graphs $G_1, ..., G_l$, computable in time $O(|V(G)| + |E(G)|)$, such that $|V(G_i)| = O(|V|)$ and $|E(G_i)| = O(|E|)$ for all $i = 1, ..., l$, and such that knowing $Agg(H, G_1, \psi), ..., Agg(H, G_l, \psi)$ allows one to compute $\text{Hom}_{H_S}(G)$ for all $S \in \mathcal{IS}(\psi)$, in time $O(1)$. Furthermore, if G is $O(1)$-degenerate, then so are $G_1, ..., G_l$.*

First, we can relate the Homomorphism Vertex Counts of a vertex $h \in V(H)$ to Homomorphism Counts from H to G, as given in the following claim:

▷ Claim 30. For all $h \in V(H)$:

$$\sum_{v \in V(G)} \text{VertexHom}_{H,h}(v) = \text{Hom}_H(G)$$

Proof.

$$\begin{aligned}
&\sum_{v \in V(G)} \text{VertexHom}_{H,h}(v) \\
&= \sum_{v \in V(G)} |\{\phi \in \Phi(H,G) : \phi(h) = v\}| && \text{(Def. of VertexHom)} \\
&= |\{\phi \in \Phi(H,G) : \phi(h) \in V(G)\}| && \text{(Sum over whole set)} \\
&= |\Phi(H,G)| && (\forall \phi : \phi(u) \in V(G)) \\
&= \text{Hom}_H(G) && \text{(Def. of Hom)}
\end{aligned}$$ ◁

We now state the following Lemma from [6]:

▶ **Lemma 31** (Lemma 4.2 from [6]). *Let $H_1, ..., H_l$ be pairwise non-isomorphic graphs and let $c_1, ..., c_l$ be non-zero constants. For every graph G there are graphs $G_1, ..., G_l$, computable in time $O(|V(G)| + |E(G)|)$, such that $|V(G_i)| = O(|V(G)|)$ and $|E(G_i)| = O(|E(G)|)$ for every $i = 1, ..., l$, and such that knowing $b_j := c_1 \cdot \text{Hom}_{H_1}(G_j) + ... + c_l \cdot \text{Hom}_{H_l}(G_j)$ for every $j = 1, ..., l$ allows one to compute $\text{Hom}_{H_1}(G), ..., \text{Hom}_{H_l}(G)$ in time $O(1)$. Furthermore, if G is $O(1)$-degenerate, then so are $G_1, ..., G_l$.*

We will apply the previous lemma in a similar way as it is used the proof of Lemma 4.1 in [6].

Proof of Lemma 29. Let $H_1, ..., H_l$ be an enumeration of all the graphs H_S for all $S \in \mathcal{IS}(\psi)$, up to isomorphism. This means that $H_1, ..., H_l$ are pairwise non-isomorphic and $\{H_1, ..., H_l\} = \{H_S : S \in \mathcal{IS}(\psi)\}$.

Let $f(i) = (-1)^{|S|+1}|\{S \in \mathcal{IS}(\psi) : H_S \text{ is isomorphic to } H_i\}|$ be the number of sets $S \in \mathcal{IS}(\psi)$ such that H_S is isomorphic to H_i, with the sign being $(-1)^{|S|+1}$. Note that all such sets have equal $|S|$ and that the value of $f(i)$ is always non-zero. We will use h_i to denote the vertex of H_i that correspond to the vertices h_S of the graphs H_S that are isomorphic to H_i. We can express $Agg(H, G, \psi)$ as follows:.

$$\begin{aligned}
Agg(H,G,\psi) &= \sum_{v \in V(G)} \text{OrbitHom}_{H,\psi}(v) && \text{(Def. 28)} \\
&= \sum_{v \in V(G)} \sum_{S \in \mathcal{IS}(\psi)} (-1)^{|S|+1} \text{VertexHom}_{H_S,h_S}(v) && \text{(Lemma 19)} \\
&= \sum_{v \in V(G)} \sum_{i=1}^{l} f(i) \text{VertexHom}_{H_i,h_i}(v) && \text{(Def. of } f(i)\text{)} \\
&= \sum_{i=1}^{l} f(i) \sum_{v \in V(G)} \text{VertexHom}_{H_i,h_i}(v) && \text{(Reorder)} \\
&= \sum_{i=1}^{l} f(i) \text{Hom}_{H_i}(G) && \text{(Claim 30)}
\end{aligned}$$

Hence, we have that $Agg(H, G, \psi)$ is a linear combination of homomorphism counts of $H_1, ..., H_l$. We can then use Lemma 31 to complete the proof. ◀

Before we prove Theorem 27, we need to state the following theorem from [8], which gives a hardness result on Homomorphism Counting:

▶ **Theorem 32** (Theorem 5.1 from [8]). *Let H be a pattern graph on k vertices with $LICL \geq 6$. Assuming the Triangle Detection Conjecture, there exists an absolute constant γ such that for any function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, there is no (expected) $f(\kappa, k)O(m^{1+\gamma})$ algorithm for the* Hom_H *problem, where m and κ are the number of edges and the degeneracy of the input graph, respectively.*

We now have all the tools required to proof Theorem 27:

Proof of Theorem 27. We prove by contradiction. Given a graph G and a pattern H with $LIPCO(H) > 5$, suppose there exists an algorithm that allows us to compute $\mathrm{OrbitHom}_H(G)$ in time $f(\kappa, k)O(m)$, by Lemma 29 we have the existence of some graphs $G_1, ..., G_l$. We can compute $\mathrm{OrbitHom}_H(G_i)$ for all of these graphs in time $f(\kappa, k)O(m)$ and then aggregate the results into $Agg(H, G_i, \psi)$ for all G_i and all $\psi \in \Psi(H)$. Using Lemma 29, that implies that we can compute $\mathrm{Hom}_{H_S}(G)$ for all $S \in \mathcal{IS}(\psi)$ for all $\psi \in \Psi(H)$ in time $f(\kappa, k)O(m)$.

However, if $LIPCO(H) > 5$ then, by Lemma 25, we have that there exists a $S \subseteq \psi$ for some $\psi \in \Psi(H)$ such that $LICL(H_S) > 5$. From Theorem 32 we know that in that case there is no algorithm that computes $\mathrm{Hom}_{H_S}(G)$ in time $f(\kappa, k)O(m^{1+\gamma})$ for some constant $\gamma > 0$. This is a contradiction, and hence no algorithm can compute $\mathrm{OrbitHom}_H(G)$ in $f(\kappa, k)O(m)$ time. ◀

References

1. Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science*, 2014. doi:10.1109/FOCS.2014.53.
2. Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick Duffield. Efficient graphlet counting for large networks. In *Proceedings, SIAM International Conference on Data Mining (ICDM)*, 2015. doi:10.1109/ICDM.2015.141.
3. Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.
4. A. Benson, D. F. Gleich, and J. Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016. doi:10.1126/science.aad9029.
5. Suman K Bera, Amit Chakrabarti, and Prantar Ghosh. Graph coloring via degeneracy in streaming and other space-conscious models. In *International Colloquium on Automata, Languages and Programming*, 2020. doi:10.4230/LIPIcs.ICALP.2020.11.
6. Suman K. Bera, Lior Gishboliner, Yevgeny Levanzov, C. Seshadhri, and Asaf Shapira. Counting subgraphs in degenerate graphs. *Journal of the ACM (JACM)*, 69(3), 2022. doi:10.1145/3520240.
7. Suman K Bera, Noujan Pashanasangi, and C Seshadhri. Linear time subgraph counting, graph degeneracy, and the chasm at size six. In *Proc. 11th Conference on Innovations in Theoretical Computer Science*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.38.
8. Suman K. Bera, Noujan Pashanasangi, and C. Seshadhri. Near-linear time homomorphism counting in bounded degeneracy graphs: The barrier of long induced cycles. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2315–2332, 2021. doi:10.1137/1.9781611976465.138.
9. Suman K Bera and C Seshadhri. How the degeneracy helps for triangle counting in graph streams. In *Principles of Database Systems*, pages 457–467, 2020. doi:10.1145/3375395.3387665.

10 Jonathan W. Berry, Bruce Hendrickson, Randall A. LaViolette, and Cynthia A. Phillips. Tolerating the community detection resolution limit with edge weighting. *Phys. Rev. E*, 83:056119, 2011. doi:10.1103/PhysRevE.83.056119.

11 Umberto Bertele and Francesco Brioschi. On non-serial dynamic programming. *J. Comb. Theory, Ser. A*, 14(2):137–148, 1973. doi:10.1016/0097-3165(73)90016-2.

12 J.A. Bondy and U.S.R Murty. *Graph Theory*, volume 244. Springer, 2008. doi:10.1007/978-1-84628-970-5.

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

14 Marco Bressan. Faster subgraph counting in sparse graphs. In *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.IPEC.2019.6.

15 Marco Bressan. Faster algorithms for counting subgraphs in sparse graphs. *Algorithmica*, 83:2578–2605, 2021. doi:10.1007/s00453-021-00811-0.

16 Marco Bressan, Leslie Ann Goldberg, Kitty Meeks, and Marc Roth. Counting subgraphs in somewhere dense graphs. In *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, pages 27:1–27:14, 2023. doi:10.4230/LIPIcs.ITCS.2023.27.

17 Marco Bressan and Marc Roth. Exact and approximate pattern counting in degenerate graphs: New algorithms, hardness results, and complexity dichotomies. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 276–285, 2022. doi:10.1109/FOCS52979.2021.00036.

18 Graham R Brightwell and Peter Winkler. Graph homomorphisms and phase transitions. *Journal of combinatorial theory, series B*, 77(2):221–262, 1999. doi:10.1006/jctb.1999.1899.

19 Ashok K Chandra and Philip M Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. 9th Annual ACM Symposium on the Theory of Computing*, pages 77–90, 1977. doi:10.1145/800105.803397.

20 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing (SICOMP)*, 14(1):210–223, 1985. doi:10.1137/0214017.

21 Jonathan Cohen. Graph twiddling in a mapreduce world. *Computing in Science & Engineering*, 11(4):29, 2009. doi:10.1109/MCSE.2009.120.

22 J. Coleman. Social capital in the creation of human capital. *American Journal of Sociology*, 94:S95–S120, 1988. doi:10.1086/228943.

23 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 210–223, 2017. doi:10.1145/3055399.3055502.

24 Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theoretical Computer Science*, 329(1-3):315–323, 2004. doi:10.1016/j.tcs.2004.08.008.

25 Holger Dell, Marc Roth, and Philip Wellnitz. Counting answers to existential questions. In *Proc. 46th International Colloquium on Automata, Languages and Programming*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.113.

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

27 Martin Dyer and Catherine Greenhill. The complexity of counting graph homomorphisms. *Random Structures & Algorithms*, 17(3-4):260–289, 2000. doi:10.1002/1098-2418(200010/12)17:3/4\%3C260::AID-RSA5\%3E3.0.CO;2-W.

28 David Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information processing letters*, 51(4):207–211, 1994. doi:10.1016/0020-0190(94)90121-X.

29 G. Fagiolo. Clustering in complex directed networks. *Phys. Rev. E*, 2007. doi:10.1103/PhysRevE.76.026107.

30 Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM Journal on Computing (SICOMP)*, 33(4):892–922, 2004. doi:10.1137/S0097539703427203.

31 Lior Gishboliner, Yevgeny Levanzov, and Asaf Shapira. Counting subgraphs in degenerate graphs, 2020. arXiv:2010.05998, doi:10.48550/arXiv.2010.05998.

32 Gaurav Goel and Jens Gustedt. Bounded arboricity to determine the local structure of sparse graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 159–167. Springer, 2006. doi:10.1007/11917496_15.

33 Rudolf Halin. S-functions for graphs. *Journal of geometry*, 8(1-2):171–186, 1976. doi:10.1007/BF01917434.

34 P. Holland and S. Leinhardt. A method for detecting structure in sociometric data. *American Journal of Sociology*, 76:492–513, 1970. doi:10.1016/B978-0-12-442450-0.50028-6.

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

36 Shalev Itzkovitz, Reuven Levitt, Nadav Kashtan, Ron Milo, Michael Itzkovitz, and Uri Alon. Coarse-graining and self-dissimilarity of complex networks. *Phys. Rev. E*, 71(016127), January 2005. doi:10.1103/PhysRevE.71.016127.

37 Shweta Jain and C Seshadhri. A fast and provable method for estimating clique counts using Turán's theorem. In *Proceedings, International World Wide Web Conference (WWW)*, pages 441–449, 2017. doi:10.1145/3038912.3052636.

38 Madhav Jha, C Seshadhri, and Ali Pinar. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proc. 24th Proceedings, International World Wide Web Conference (WWW)*, pages 495–505. International World Wide Web Conferences Steering Committee, 2015. doi:10.1145/2736277.2741101.

39 László Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3-4):321–328, 1967. doi:10.1007/BF02280291.

40 László Lovász. *Large networks and graph limits*, volume 60. American Mathematical Soc., 2012. URL: http://www.ams.org/bookstore-getitem/item=COLL-60.

41 David W Matula and Leland L Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)*, 30(3):417–427, 1983. doi:10.1145/2402.322385.

42 Derek O'Callaghan, Martin Harrigan, Joe Carthy, and Pádraig Cunningham. Identifying discriminating network motifs in youtube spam, 2012. arXiv:1202.5216, doi:10.48550/arXiv.1202.5216.

43 Mark Ortmann and Ulrik Brandes. Efficient orbit-aware triad and quad census in directed and undirected graphs. *Applied network science*, 2(1), 2017. doi:10.1007/s41109-017-0027-2.

44 Noujan Pashanasangi and C Seshadhri. Efficiently counting vertex orbits of all 5-vertex subgraphs, by evoke. In *Proc. 13th International Conference on Web Search and Data Mining (WSDM)*, pages 447–455, 2020. doi:10.1145/3336191.3371773.

45 Ali Pinar, C Seshadhri, and Vaidyanathan Vishal. Escape: Efficiently counting all 5-vertex subgraphs. In *Proceedings, International World Wide Web Conference (WWW)*, pages 1431–1440, 2017. doi:10.1145/3038912.3052597.

46 Natasa Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):177–183, 2007. doi:10.1093/bioinformatics/btl301.

47 Neil Robertson and Paul D. Seymour. Graph minors. i. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983. doi:10.1016/0095-8956(83)90079-5.

48 Neil Robertson and Paul D. Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984. doi:10.1016/0095-8956(84)90013-3.

49 Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

50 Rahmtin Rotabi, Krishna Kamath, Jon M. Kleinberg, and Aneesh Sharma. Detecting strong ties using network motifs. In *Proceedings, International World Wide Web Conference (WWW)*, 2017. doi:10.1145/3041021.3055139.

51 Marc Roth and Philip Wellnitz. Counting and finding homomorphisms is universal for parameterized complexity theory. In *Proc. 31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2161–2180, 2020. doi:10.1137/1.9781611975994.133.

52 Ahmet Erdem Sariyuce, C. Seshadhri, Ali Pinar, and Umit V. Catalyurek. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proceedings, International World Wide Web Conference (WWW)*, pages 927–937, 2015. doi:10.1145/2736277.2741640.

53 C. Seshadhri and Srikanta Tirthapura. Scalable subgraph counting: The methods behind the madness: WWW 2019 tutorial. In *Proceedings, International World Wide Web Conference (WWW)*, 2019. doi:10.1145/3308560.3320092.

54 Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten M. Borgwardt. Efficient graphlet kernels for large graph comparison. In *AISTATS*, pages 488–495, 2009. URL: http://proceedings.mlr.press/v5/shervashidze09a.html.

55 K. Shin, T. Eliassi-Rad, and C. Faloutsos. Patterns and anomalies in k-cores of real-world graphs with applications. *Knowledge and Information Systems*, 54(3):677–710, 2018. doi:10.1007/s10115-017-1077-6.

56 George Szekeres and Herbert S Wilf. An inequality for the chromatic number of a graph. *Journal of Combinatorial Theory*, 4(1):1–3, 1968. doi:10.1016/S0021-9800(68)80081-X.

57 Charalampos E. Tsourakakis. The k-clique densest subgraph problem. In *Proceedings, International World Wide Web Conference (WWW)*, pages 1122–1132, 2015. doi:10.1145/2736277.2741098.

58 Charalampos E. Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. Scalable motif-aware graph clustering. In *Proceedings, International World Wide Web Conference (WWW)*, pages 1451–1460, 2017. doi:10.1145/3038912.3052653.

59 Johan Ugander, Lars Backstrom, and Jon M. Kleinberg. Subgraph frequencies: mapping the empirical and extremal geography of large graph collections. In *Proceedings, International World Wide Web Conference (WWW)*, pages 1307–1318, 2013. doi:10.1145/2488388.2488502.

60 Hao Yin, Austin R. Benson, and Jure Leskovec. Higher-order clustering in networks. *Phys. Rev. E*, 97:052306, 2018. doi:10.1103/PhysRevE.97.052306.

61 Hao Yin, Austin R. Benson, and Jure Leskovec. The local closure coefficient: A new perspective on network clustering. In *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 303–311, 2019. doi:10.1145/3289600.3290991.