Counting Subgraphs in Degenerate Graphs

SUMAN K. BERA, University of California
LIOR GISHBOLINER and YEVGENY LEVANZOV, School of Mathematics, Tel Aviv University
C. SESHADHRI, University of California
ASAF SHAPIRA, School of Mathematics, Tel Aviv University

We consider the problem of counting the number of copies of a fixed graph H within an input graph G. This is one of the most well-studied algorithmic graph problems, with many theoretical and practical applications. We focus on solving this problem when the input G has *bounded degeneracy.* This is a rich family of graphs, containing all graphs without a fixed minor (e.g., planar graphs), as well as graphs generated by various random processes (e.g., preferential attachment graphs). We say that H is *easy* if there is a linear-time algorithm for counting the number of copies of H in an input G of bounded degeneracy. A seminal result of Chiba and Nishizeki from '85 states that every H on at most 4 vertices is easy. Bera, Pashanasangi, and Seshadhri recently extended this to all H on 5 vertices and further proved that for every $k > 5$ there is a k-vertex H which is not easy. They left open the natural problem of characterizing all easy graphs H.

Bressan has recently introduced a framework for counting subgraphs in degenerate graphs, from which one can extract a sufficient condition for a graph H to be easy. Here, we show that this sufficient condition is also necessary, thus fully answering the Bera–Pashanasangi–Seshadhri problem. We further resolve two closely related problems; namely characterizing the graphs that are easy with respect to counting induced copies, and with respect to counting homomorphisms.

CCS Concepts: • **Theory of computation → Graph algorithms analysis**; **Design and analysis of algorithms**; **Streaming, sublinear and near linear time algorithms;**

Additional Key Words and Phrases: Graph algorithms, subgraph counting, linear time algorithms, degeneracy

ACM Reference format:
Suman K. Bera, Lior Gishboliner, Yevgeny Levanzov, C. Seshadhri, and Asaf Shapira. 2022. Counting Subgraphs in Degenerate Graphs. *J. ACM* 69, 3, Article 23 (June 2022), 21 pages.
https://doi.org/10.1145/3520240

1 INTRODUCTION

Subgraph counting refers to the algorithmic task of computing the number of copies (i.e., occurences) of a given graph H in an input graph G. Due to its fundamental nature, this problem

Suman K. Bera and C. Seshadhri are supported by NSF TRIPODS grant CCF-1740850, NSF CCF-1813165, CCF-1909790, and ARO Award W911NF1910294. Lior Gishboliner and Yevgeny Levanzov are supported by the ERC Starting Grant 633509. Asaf Shapira is supported in part by ISF Grant 1028/16 and ERC Starting Grant 633509.
Authors' addresses: S. K. Bera and C. Seshadhri, University of California, Santa Cruz, CA 95060; emails: sumankalyanbera@gmail.com, sesh@ucsc.edu; L. Gishboliner, Y. Levanzov, and A. Shapira, School of Mathematics, Tel Aviv University, Tel Aviv 69978, Israel; emails: {liorgis1, yevgenyl}@mail.tau.ac.il, asafico@mail.tau.ac.il.

0004-5411/2022/06-ART23 $15.00
https://doi.org/10.1145/3520240

has been studied extensively, both from a theoretical perspective and for practical applications. In practice, subgraph counts are widely used to analyze real-world graphs, such as graphs representing telecommunication networks, biological structures, and social interactions. Consequently, subgraph counts feature prominently in studies of biological [28, 41, 42] and sociological [15, 27] networks, as well as in the network science literature in general [7, 30, 38, 44–46]. For example, [38] observed that networks coming from different areas of science (such as biochemistry, neurobiology, ecology, and engineering) have significantly different counts of small subgraphs. Such frequently occurring subgraphs are called *motifs*, and, quoting [38], "may uncover the basic building blocks of most networks". Needless to say, some real-world graphs can be very large—having billions of vertices—thus making it all the more desirable to have fast subgraph counting algorithms.

In theoretical computer science, subgraph counting and detection[1] are fundamental and widely-studied problems. Much of the research focused on counting special kinds of graphs, such as cliques [23, 29, 40, 48]; cycles [3, 21, 29]; paths and matchings (and other graphs with bounded pathwidth) [8, 9]; graphs with a bounded vertex-cover number [19, 20, 32, 50]. Many of the algorithms use fast matrix multiplication [3, 23, 29, 32, 40]. For example, the best-known algorithm for counting k-cliques [40] runs in time $n^{\omega k/3+O(1)}$, where $\omega < 2.373$ is the matrix multiplication constant [49]. On the negative side, k-clique counting is the canonical #W[1]-hard problem, and it is thus unlikely that there exists an algorithm that solves this problem in time $f(k) \cdot n^{o(k)}$ (for any function f). We refer the reader to [10] for further references on both theoretical and practical aspects of subgraph counting.

Subgraph counts also play a fundamental role in extremal graph theory, where subgraph densities are the basic notion used for studying sequences of dense graphs [33]. In particular, knowing approximate subgraph counts of small graphs inside a given graph G allows one to decide whether G is quasirandom [18] or, more generally, whether G consists of a bounded number of quasirandom pieces with prescribed edge densities [34].

Given the importance of the subgraph counting problem on the one hand, and its hardness in general graphs on the other, it is natural to consider special classes of graphs which admit faster counting algorithms, while also being rich enough to include many of the real-world graphs mentioned above. One prime example of such a family of graph classes is classes having bounded degeneracy. Recall that a graph G is *κ-degenerate*[2] if there is an ordering $v_1, \dots, v_n$ of the vertices of G such that v_i has at most κ neighbors in $\{v_{i+1}, \dots, v_n\}$ (for each $1 \leq i \leq n$). We say that a class of graphs has *bounded degeneracy* if there is an integer κ such that all graphs in the class are κ-degenerate. With a slight abuse of terminology, we will refer to graphs belonging to such classes as having bounded-degeneracy or being $O(1)$-degenerate.

There are many examples of well-studied graph classes having bounded degeneracy. These include all minor-closed classes (including planar graphs and, more generally, graphs embeddable into a given surface), preferential attachment graphs [4], and bounded expansion graphs [39].

The first result on subgraph counting in bounded-degeneracy graphs is probably the classical result of Chiba and Nishizeki [17], who showed that in κ-degenerate graphs, one can count r-cliques in time $O(n\kappa^{r-2})$ (for each $r \geq 3$), and 4-cycles in time $O(n\kappa)$. Bera, Pashanasangi, and Seshadhri [10] recently extended this result, by showing that the H-counting problem in bounded-degeneracy graphs can be solved in time $O(n)$ for every graph H on at most 5 vertices (here, the implicit constant in the big-O notation depends on the degeneracy κ). They further

[1]The H-detection problem is the problem of deciding whether an input graph contains a copy of H.

[2]We note that degeneracy is closely related to another well-studied graph parameter, namely arboricity, which is the minimum number of forests into which the edge-set of a graph can be partitioned. It is well-known that the arboricity of a κ-degenerate graph is between $(\kappa + 1)/2$ and κ.

showed that under a certain widely-believed hardness assumption in fine-grained complexity, the problem of counting 6-cycles cannot be solved in linear time in bounded-degeneracy graphs. (They also proved a similar result for all longer cycles, with the exception of the cycle of length 8.) The hardness assumption used asserts that detecting a triangle in a (general) graph with m edges requires significantly more than $O(m)$ time. This assumption will also serve as the foundation for our complexity-theoretic hardness results, and we state it here as follows. We refer the reader to [1], where Conjecture 1.1 was first formulated, for a detailed overview of the conjecture and its relations to many other computational problems.

Conjecture 1.1 (Triangle Detection Conjecture [1]). *There exists $\gamma > 0$ such that in the word RAM model of $O(\log n)$ bits, any algorithm to decide whether an input graph with n vertices and m edges is triangle-free requires $\Omega(m^{1+\gamma})$ time in expectation.*

It is believed that the constant γ in Conjecture 1.1 could be as large as 1/3. The reason for this is that the best-known algorithm for triangle detection [3] runs in time $O(\min(n^{\omega}, m^{2\omega/(\omega+1)}))$, where ω is the matrix multiplication constant. If $\omega = 2$ (which would be optimal), then the running time of this algorithm is $O(\min(n^2, m^{4/3}))$.

Having established both positive and negative results, Bera, Pashanasangi, and Seshadhri [10] asked whether one can characterize all graphs H for which the H-counting problem can be solved in linear time in bounded-degeneracy graphs. Before proceeding to our resolution of this question, let us recall some definitions and introduce some notation.

Definition 1.2. Let G, H be graphs. A map $\varphi : V(H) \to V(G)$ is

(1) a *homomorphism* if $\{\varphi(u), \varphi(v)\} \in E(G)$ for every $\{u, v\} \in E(H)$;
(2) an *injective homomorphism* if it is a homomorphism and a one-to-one function;
(3) an *isomorphism* if it is an injective homomorphism and $\{\varphi(u), \varphi(v)\} \notin E(G)$ if $\{u, v\} \notin E(H)$.

Let G, H be graphs. Following [33], we denote by $\hom(H, G)$ the number of homomorphisms from H to G, by $\mathrm{inj}(H, G)$ the number of injective homomorphisms (i.e., embeddings) from H to G, and by $\mathrm{ind}(H, G)$ the number of isomorphisms from H to (induced subgraphs of) G. For a graph H, we denote by $\textsc{hom-cnt}_H$ the problem of computing $\hom(H, G)$ for a given input graph G. The problems $\textsc{inj-cnt}_H$ and $\textsc{ind-cnt}_H$ are defined analogously. In what follows, we say that $\textsc{hom-cnt}_H/\textsc{inj-cnt}_H/\textsc{ind-cnt}_H$ is *easy* if it can be solved in time $f(\kappa, H) \cdot \tilde{O}(n)$ in n-vertex graphs of degeneracy κ (for some function f); and otherwise, we say that it is *hard*. We will usually avoid mentioning the function f, and just speak of running time $\tilde{O}(n)$, namely (nearly) linear time, with the implicit constant in the big-$\tilde{O}$ notation allowed to depend on κ and H. The following is the main open problem raised in [10].

Problem 1.3 ([10]). *Characterize the graphs H for which $\textsc{inj-cnt}_H$ is easy.*

In this article, we completely resolve Problem 1.3 by giving a very clean characterization of the graphs H for which $\textsc{inj-cnt}_H$ is easy. We will also solve the related problems of characterizing the graphs H for which $\textsc{hom-cnt}_H$ is easy and the graphs H for which $\textsc{ind-cnt}_H$ is easy. It will turn out to be more convenient to first deal with the problem of obtaining a characterization for $\textsc{hom-cnt}_H$. This characterization, discussed in Section 1.2, constitutes the main result of this article. In Section 1.3, we describe how the solution of Problem 1.3 regarding $\textsc{inj-cnt}_H$ can be derived from our result regarding $\textsc{hom-cnt}_H$. Then, in Section 1.4, we describe how a characterization for $\textsc{ind-cnt}_H$ can be derived from the one for $\textsc{inj-cnt}_H$. This approach—of relating homomorphisms to copies and copies to induced copies—was pioneered in [19] and [16], and is based on the framework developed in [33]. Finally, in Section 1.5, we briefly consider subgraph-counting in *general* (i.e., not necessarily degenerate) graphs. By applying the methods used for proving Theorem 1, we show that for a graph H, $\hom(H,G)$ can be computed in time $\tilde{O}(|V(G)|+|E(G)|)$ in general graphs if and only if H is a forest (again, the "only-if" direction assumes Conjecture 1.1).

1.1 α-acyclic Hypergraphs and Bressan's Algorithm

Bressan [13] provided a dynamic programming algorithm for computing $\hom(H,G)$ in $O(1)$-degenerate graphs G. His main result is that computing $\hom(H,G)$ in n-vertex $O(1)$-degenerate graphs can be done in time[3] $\tilde{O}(n^{\tau_1(H)})$, where $\tau_1(H)$ is a certain "width parameter" called the *DAG treewidth* of H (we use the notation τ_1 to be consistent with [13]). In our language, the special case $\tau_1(H) = 1$ of Bressan's result states that HOM-CNT$_H$ is easy if $\tau_1(H) = 1$. As we will mainly focus on the case $\tau_1(H) = 1$, we shall not define $\tau_1(H)$ here, but shall only define what it means for a graph to satisfy $\tau_1(H) = 1$. To do so, we first need to recall the well-known notion of hypergraph α-acyclicity, which is closely related to the parameter τ_1.

Definition 1.4 (α-acyclic Hypergraph). A hypergraph F is called α-acyclic if there exists a tree T whose vertices are the hyperedges of F, such that the following condition is satisfied: for all $e_1, e_2, e \in E(F) = V(T)$, if e is on the unique path in T between e_1 and e_2, then $e_1 \cap e_2 \subseteq e$.

Hypergraph α-acyclicity was introduced by Beeri, Fagin, Maier, Mendelzon, Ullman, and Yannakakis [5] in the early 1980s in connection with relational database schemes, and has subsequently been widely studied. For further information, we refer the reader to [6, 12]. Note that in the special case that F is a graph, α-acyclicity is equivalent to being a forest.

Next, one associates to each directed acyclic graph[4] (or DAG, for short) $\vec{H}$ a hypergraph that captures the *reachability structure* of $\vec{H}$. For a vertex u in a directed graph $\vec{H}$, we denote by $R(u)$ the set of vertices that are *reachable* from u (namely, the set of all $v \in V(\vec{H})$ such that there is a directed path from u to v). Note that $u \in R(u)$. Recall that any DAG has at least one source (i.e., vertex of in-degree 0). Given a DAG $\vec{H}$ with source vertices $u_1, \dots, u_r$, define an associated hypergraph $F_{\vec{H}}$, as follows: the vertices of $F_{\vec{H}}$ are the vertices of $\vec{H}$, and the hyperedges of $F_{\vec{H}}$ are the sets $R(u_i)$ for $1 \le i \le r$. Using this definition (and with a slight abuse of terminology), we can now define what it means for a DAG or an undirected graph to be α-acyclic.

Definition 1.5 (α-acyclic Graphs and Digraphs). We say that a DAG $\vec{H}$ is α-acyclic if the hypergraph $F_{\vec{H}}$ is α-acyclic. We say that an undirected graph H is α-acyclic if every acyclic orientation $\vec{H}$ of H is α-acyclic.

For an undirected graph H, having $\tau_1(H) = 1$ is precisely equivalent to being α-acyclic (this is merely a restatement of the definition of τ_1 for the case $\tau_1(H) = 1$ in a different language). As mentioned above, a special case of the main result of [13] implies that $\hom(H,G)$ can be computed in time $\tilde{O}(n)$ whenever H is α-acyclic. Our main result, described in the following section, states that this sufficient condition is also necessary, thus giving a complete characterization of the graphs H for which HOM-CNT$_H$ is easy.

[3]If one forgoes the requirement that the algorithm be deterministic, instead allowing Las Vegas randomized algorithms (as is done for example in [10]), then one can avoid the implicit polylogarithmic factor, obtaining an algorithm which runs in expected time $O(n^{\tau_1(H)})$. Indeed, the polylogarithmic factor arises from the need to search and update a dictionary with $O(n^{\tau_1(H)})$ entries. This can be done in time $O(1)$ by using perfect hashing [26], at the cost of needing expected time $O(n^{\tau_1(H)})$ to generate the hash table. Once generated, the algorithm proceeds deterministically.

[4]We note that directed acyclic graphs arise naturally in the context of counting subgraphs in degenerate graphs. Indeed, many of the known counting algorithms [10, 13, 17] begin by orienting the edges of the given κ-degenerate input graph G according to some *degeneracy ordering*, thus obtaining an acyclic directed graph $\vec{G}$ in which all out-degrees are at most κ. Then the task becomes to compute $\hom(\vec{H}, \vec{G})$ for every acyclic orientation $\vec{H}$ of H. Summing these directed homomorphism counts then gives $\hom(H, G)$.

1.2 Main Result: Counting Homomorphisms in Linear Time

Our main result in this article is as follows.

THEOREM 1 (MAIN RESULT). *Assuming Conjecture 1.1, HOM-CNT$_H$ is hard whenever H is not α-acyclic.*

As mentioned above, Theorem 1 shows that the sufficient condition (for HOM-CNT$_H$ being easy) supplied by Bressan's algorithm [13] is in fact necessary. Thus we obtain the following characterization:

COROLLARY 2. *HOM-CNT$_H$ is easy if and only if H is α-acyclic. The "only if" part is under Conjecture 1.1.*

Given Theorem 1, it is natural to ask if there is a cleaner description of the α-acyclic graphs. Actually, another reason for seeking such a clean description is that in order to prove Theorem 1, it would be desirable to know that graphs that are not α-acyclic have certain easy-to-describe obstructions. Luckily (and somewhat surprisingly), we have the following concise equivalent description of the α-acyclic graphs. Throughout the article, C_k denotes the cycle of length k. For graphs H, H_0, we say that H is *induced H_0-free* if H contains no induced copy of H_0.

THEOREM 3. *An undirected graph H is α-acyclic if and only if H is induced C_k-free for every $k \geq 6$.*

By combining Corollary 2 and Theorem 3, we immediately obtain the following very clean characterization of the graphs H for which HOM-CNT$_H$ is easy.

COROLLARY 4. *HOM-CNT$_H$ is easy if and only if H is induced C_k-free for all $k \geq 6$. The "only if" part is under Conjecture 1.1.*

Let us now discuss the ideas that go into the proofs of Theorems 1 and 3. The proof of Theorem 3 relies on a useful characterization of α-acyclic hypergraphs given in [6] (and stated here as Theorem 8). This characterization asserts that a hypergraph is α-acyclic if and only if it does not contain two certain types of *obstructions*. These two types of obstructions generalize in different ways the notion of an induced cycle from graphs to hypergraphs. The main difficulty in the proof of Theorem 3 lies in translating these "hypergraph obstructions" into "digraph obstructions", i.e., recognizing the digraph structures whose reachability hypergraphs correspond to these obstructions.

We now move on to discuss the proof of Theorem 1. With Theorem 3 at hand, it is natural to first try and show that HOM-CNT$_{C_k}$ is hard for all $k \geq 6$. This is indeed accomplished in the following lemma.

LEMMA 1.6. *Assuming Conjecture 1.1, HOM-CNT$_{C_k}$ is hard for every $k \geq 6$.*

The proof of Lemma 1.6 works by reducing triangle detection (in general graphs) to the problem of counting colorful C_k-homomorphisms (in bounded-degeneracy graphs), which is in turn reduced to HOM-CNT$_{C_k}$.

The remaining ingredient in the proof of Theorem 1 is the following lemma, which states that if HOM-CNT$_H$ is easy then so is HOM-CNT$_{H'}$ for every induced subgraph H' of H. It is easy to see that the combination of Theorem 3 and Lemmas 1.6 and 1.7 implies Theorem 1.

LEMMA 1.7. *If HOM-CNT$_H$ is easy, then HOM-CNT$_{H'}$ is easy for every induced subgraph H' of H.*

The proof of Lemma 1.7 uses an innovative application of a powerful technique developed recently in several works on homomorphism-counting, see [16, 19]. At the heart of this technique is the observation that for (non-isomorphic) graphs $H_1, \ldots, H_k$ and non-zero constants $c_1, \ldots, c_k$, the problem of computing the linear combination $c_1 \hom(H_1, \cdot) + \cdots + c_k \hom(H_k, \cdot)$ is as hard as computing $\hom(H_i, \cdot)$ for every $1 \leq i \leq k$. The proof of this fact (appearing in [19]) uses tensor products of graphs and a result of Erdős, Lovász, and Spencer [24] (stated here as Lemma A.2) regarding linear independence of homomorphism counts. For completeness, we present this proof (adapted to the setting of input graphs of bounded degeneracy and stated here as Lemma 4.2) in the appendix. The use of linear combinations of homomorphism-counts plays a crucial role in many of the reductions presented in this article.

Our proof of Lemma 1.7 proceeds by showing that for every graph G, one can (efficiently) construct a graph G' such that $\hom(H, G')$ equals to a linear combination of the homomorphism counts of all induced subgraphs of H in G. Unlike Lemmas 1.8 and 1.9, which are similar (both in their statements and their proofs) to results obtained in [19], Lemma 1.7 is (to the best of our knowledge) a new application of the homomorphism-linear-combination technique.

In the next two sections, we take advantage of known relations between the subgraph counts hom, inj, ind in order to derive analogues of Corollary 4 for the problems INJ-CNT and IND-CNT. A similar approach was taken in [19].

1.3 From Homomorphisms to Copies

In this section, we obtain a characterization of graphs H for which INJ-CNT_H is easy, thus resolving Problem 1.3. To state this characterization, we first need to introduce the notion of a *quotient graph*. For a graph H and a partition $P = \{U_1, \ldots, U_k\}$ of $V(H)$, the *quotient graph* H/P is the graph on P in which, for every $1 \leq i, j \leq k$, there is an edge[5] between U_i and U_j if and only if there are $u_i \in U_i, u_j \in U_j$ such that $\{u_i, u_j\} \in E(H)$. The following is our main result for INJ-CNT_H.

Theorem 5. *For a graph H, INJ-CNT_H is easy if and only if every quotient graph of H is induced C_k-free for all $k \geq 6$. The "only if" part is under Conjecture 1.1.*

One can use Theorem 5 to easily obtain the (finite) list of minimal obstructions (forbidden induced subgraphs) for INJ-CNT_H being easy. However, since this is somewhat lengthy, we shall not give the details.

The reason for the appearance of quotient graphs in Theorem 5 is that they can be used to relate homomorphism counts to injective homomorphism counts. Indeed, it is well-known and easy to show (see [33, Section 5.2.3]) that

$$\hom(H, G) = \sum_P \text{inj}(H/P, G), \tag{1}$$

where P runs over all partitions of $V(H)$. Equation (1) can be thought of as a relation over the poset of all partitions of $V(H)$. As is well-known [33, Section 5.2.3], one can invert this relation using Möbius inversion, thus obtaining the following:

$$\text{inj}(H, G) = \sum_P \mu_{\text{part}}(P) \cdot \hom(H/P, G). \tag{2}$$

Here, μ_{part} is the Möbius function of the partition poset.

Equation (2) expresses $\text{inj}(H, G)$ as a linear combination of $\hom(H/P, G)$ (where P runs over all partitions of $V(H)$). As mentioned above, this means that computing $\text{inj}(H, G)$ is exactly as hard as computing $\hom(H/P, G)$ for all P. Thus, we have the following:

Lemma 1.8. *Let H be a graph. Then INJ-CNT_H is easy if and only if $\text{HOM-CNT}_{H/P}$ is easy for every partition P of $V(H)$.*

[5]Note that if some U_i is not an independent set in H, then the vertex U_i has a loop in H/P. Such partitions P can be safely ignored in all of our arguments, since our input graphs G are always assumed to be simple, and hence $\hom(H/P, G) = 0$ if H/P has a loop.

A similar result has appeared in [19]. It is easy to see that Lemma 1.8 and Corollary 4 together imply Theorem 5. A corollary of Lemma 1.8 is that INJ-CNT$_H$ is at least as hard as HOM-CNT$_H$.

1.4 From Copies to Induced Copies

In this subsection, we apply the results of the previous two subsections in order to obtain a characterization of the graphs H for which IND-CNT$_H$ is easy. Here, we shall also take the further (simple) step of identifying the minimal obstructions for IND-CNT$_H$ being easy. This is done in Theorem 6. Here and throughout the article, a *supergraph* of H is any graph on $V(H)$ of which H is a subgraph.

THEOREM 6. *For a graph H, IND-CNT$_H$ is easy if and only if H does not contain as an induced subgraph any (not necessarily induced) spanning subgraph of C_6. The "only if" part is under Conjecture 1.1.*

Theorem 6 implies, for example, that IND-CNT$_H$ is hard if H contains an independent set of size 6, an induced path of length 5, or an induced matching of size 3. In the positive direction, we can conclude, for example, that IND-CNT$_H$ is easy whenever H has no independent set of size 3 (i.e., is the complement of a triangle-free graph).

Theorem 6 is derived from a result analogous to Lemma 1.8, this time relating the problems IND-CNT and INJ-CNT. Again, this relation has been previously used in [19]. It is well-known and easy to see that the following holds for every pair of graphs G, H.

$$\mathrm{inj}(H,G) = \sum_{E \subseteq \binom{V(H)}{2} \setminus E(H)} \mathrm{ind}(H \cup E, G). \tag{3}$$

Here, $H \cup E$ is the graph obtained from H by adding to it all edges in E. Hence, $H \cup E$ runs over all supergraphs of H. The Equation (3) can be thought of as a relation over the boolean poset of all subsets of $\binom{V(H)}{2} \setminus E(H)$. Just like Equation (1), it is well-known that this relation can be inverted using Möbius inversion (which in this case boils down to the inclusion-exclusion principle, see [33, Section 5.2.3]). The resulting inverted relation is

$$\mathrm{ind}(H,G) = \sum_{E \subseteq \binom{V(H)}{2} \setminus E(H)} (-1)^{|E|} \cdot \mathrm{inj}(H \cup E, G). \tag{4}$$

Equation (4) shows that computing $\mathrm{inj}(H', G)$ for every supergraph H' of H is sufficient for computing $\mathrm{ind}(H, G)$. The following lemma states that it is also necessary. The lemma also gives a (seemingly weaker but in fact equivalent) necessary and sufficient condition in terms of computing $\mathrm{hom}(H', G)$ for every supergraph H' of H.

LEMMA 1.9. *For every graph H, the following are equivalent.*

(1) *IND-CNT$_H$ is easy.*
(2) *INJ-CNT$_{H'}$ is easy for every supergraph H' of H.*
(3) *HOM-CNT$_{H'}$ is easy for every supergraph H' of H.*

Given the analogy between Equations (2) and (4), one could hope for a reduction between IND-CNT and INJ-CNT that does not go through HOM-CNT. We are not aware of such a reduction, however. Instead, we again exploit the homomorphism-linear-combination framework in order to reduce HOM-CNT$_{H'}$ (for all supergraphs H' of H) to IND-CNT$_H$. We then complete the picture by proving that solving HOM-CNT$_{H'}$ for all supergraphs H' of H allows one to solve INJ-CNT$_{H'}$ for all such H', which in turn allows one to solve IND-CNT$_H$ using Equation (4). This step (i.e., proving the implication $3 \Rightarrow 2$) requires Lemma 1.7.

A corollary of Lemma 1.9 is that IND-CNT$_H$ is at least as hard as INJ-CNT$_H$ (which itself is at least as hard as HOM-CNT$_H$). At the end of Section 6, we give the simple derivation of Theorem 6 from Lemma 1.9 and Corollary 4.

We conclude by noting that the reductions used to prove Lemmas 1.7, 1.8, and 1.9 are more robust than is stated in those lemmas. Namely, these reductions pertain not only to algorithms running in time $\tilde{O}(n)$, but to larger running times as well. See Lemmas 4.1, 5.1, and 6.1 for the general statements.

1.5 Counting Homomorphisms in Linear Time in General Graphs

In this section, we obtain a characterization of the graphs H such that $\hom(H, \cdot)$ can be computed in (near-)linear time in *general* (i.e., not necessarily degenerate) input graphs. Dalmau and Jonsson [22] have shown that the complexity of counting H-homomorphisms (in general graphs) is essentially controled by the tree-width $\mathrm{tw}(H)$ of H: it had been previously shown (see [25, Proposition 7]) that $\hom(H, \cdot)$ can be computed in time $O(n^{\mathrm{tw}(H)+1})$ in n-vertex graphs; and conversely, Dalmau and Jonsson [22] have shown that there is a function $f : \mathbb{N} \to \mathbb{N}$ with $f(t) \to \infty$ (as $t \to \infty$), such that $\hom(H, \cdot)$ cannot be computed in time $O(n^{f(\mathrm{tw}(H))})$ in n-vertex graphs (under the assumption that FPT does not equal #W[1]). Here, we obtain a sharper[6] result for the special case of linear runtime; we show that $\hom(H, \cdot)$ can be computed in linear time *if and only if* $\mathrm{tw}(H) = 1$, namely H is a forest.

THEOREM 7. *For a graph H, $\hom(H, G)$ can be computed in time $\tilde{O}(|V(G)| + |E(G)|)$ if and only if H is a forest. The "only if" part is under Conjecture 1.1.*

Some Open Questions. Through a series of works [10, 11], including the present article, we have gained a deep understanding of what kind of patterns can be counted in near-linear time in degenerate graphs. However, from a theoretical standpoint, it would be interesting to explore beyond linear-time algorithms. Specifically, we pose the following question: *Can we characterize patterns that are countable in quadratic time?*

A more general question is to determine to which extent the DAG treewidth $\tau_1(H)$ controls the runtime of counting H-homomorphisms in bounded-degeneracy graphs. There exist small graphs H for which HOM-CNT$_H$ can be solved in time n^k for $k < \tau_1(H)$, meaning that the exponent does not always equal $\tau_1(H)$. Still, it could be the case that counting H-homomorphisms (in bounded-degeneracy graphs) requires time $n^{f(\tau_1(H))}$, for a function $f(t) : \mathbb{N} \to \mathbb{N}$ which tends to infinity with t; namely, the runtime exponent grows with $\tau_1(H)$. It would be very interesting to determine whether this is indeed the case.[7] This question was also considered in the very recent article of Bressan and Roth [14]. While the question for HOM-CNT$_H$ remains open, [14] obtained nearly tight bounds for the problems INJ-CNT$_H$ and IND-CNT$_H$; for these problems, the parameters governing the runtime exponent are respectively the induced matching number of H (for INJ-CNT$_H$) and the independence number of H (for IND-CNT$_H$). These results are incomparable to ours, since we obtain a precise characterization for the case of linear runtime (see Theorems 5 and 6), while the results of [14] are about the asymptotics of the runtime exponent.

Note. The main results presented in this article have been obtained independently by two groups: one consisting of the first and fourth authors and N. Pashanasangi, and the other consisting of the

[6]Comparing Theorem 7 to the result of [22], we note that a special case of the latter also uses a reduction from the triangle-counting problem; it shows that, assuming Conjecture 1.1, $\hom(H, \cdot)$ cannot be computed in time $O(|V(G)|)$ whenever H contains the 3×3 grid as a minor. Evidently, this condition does not capture all non-forest graphs H (cf. Theorem 7).
[7]As mentioned above, a result of this type for counting homomorphisms in general graphs is known [22]; in this setting, the appropriate parameter is treewidth (and not DAG treewidth).

second, third, and fifth authors. In particular, in a preliminary version of the work, Bera et al. [11] effectively proved the results stated in Theorems 1 and 3. In their presentation, they use the notion of DAG treewidth instead of the language of α-acyclicity. Since the results obtained by the two groups were very similar, we decided to prepare a unified article.

Article organization. The rest of the article is organized as follows. Section 2 contains the proof of Theorem 3. Section 3 is devoted to proving Lemma 1.6. The proofs of Lemmas 1.7, 1.8 and 1.9 appear in Sections 4, 5, and 6, respectively, with Section 6 also containing the proof of Theorem 6. Finally, in Section 7, we prove Theorem 7.

2 A CHARACTERIZATION OF α-ACYCLIC GRAPHS: PROOF OF THEOREM 3

The key step in the proof of Theorem 3 is the following lemma. Recall that for a vertex u in a digraph, $R(u)$ denotes the set of all vertices reachable from u.

Lemma 2.1. *Let $\vec{H}$ be an acyclic orientation of an undirected graph H. Let $k \geq 3$, and assume that there exist distinct vertices $u_0, \dots, u_{k-1}, x_0, \dots, x_{k-1} \in V(\vec{H})$ such that for all $0 \leq i \leq k-1$, we have that $x_i \in R(u_{i-1}) \cap R(u_i)$ and $x_i \notin R(u_j)$ for all $j \neq i, i-1$ (with indices taken modulo k). Then, H contains an induced copy of a cycle C_ℓ for some $\ell \geq 6$.*

Proof. We say that a $2k$-tuple $(v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1})$ of distinct vertices of $\vec{H}$ is *good* if for every $0 \leq i \leq k-1$, $y_i \in R(v_i) \cap R(v_{i-1})$ and $y_i \notin R(v_j)$ for all $j \neq i, i-1$ (with indices taken modulo k). In other words, $(v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1})$ is good if and only if for every $0 \leq i \leq k-1$, there are directed paths from v_i to y_i and y_{i+1}, and there is no directed path from v_i to y_j for any $j \neq i, i+1$. By assumption, the tuple $(u_0, \dots, u_{k-1}, x_0, \dots, x_{k-1})$ is good, implying that the set of good $2k$-tuples is non-empty. Observe that for a good $2k$-tuple $(v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1})$ and $0 \leq i \leq k-1$, it holds that $R(v_i) \cap R(v_{i-1}) \cap \{y_0, \dots, y_{k-1}\} = \{y_i\}$ (here we use the assumption that $k \geq 3$). This implies that $R(v_i) \cap R(v_{i-1}) \cap \{v_0, \dots, v_{k-1}\} = \emptyset$, because if $v_j \in R(v_i) \cap R(v_{i-1})$ (for some $0 \leq j \leq k-1$), then $y_j, y_{j+1} \in R(v_j) \subseteq R(v_i) \cap R(v_{i-1})$, which we just ruled out. Similarly, the definition of a good $2k$-tuple implies that if $y_i, y_{i+1} \in R(v_j)$ (for some $0 \leq i, j \leq k-1$), then $j = i$ (again, we are using here the assumption that $k \geq 3$). This implies that there are no $0 \leq i, j \leq k-1$ such that $y_i, y_{i-1} \in R(y_j)$, since otherwise we would have $y_i, y_{i-1} \in R(y_j) \subseteq R(v_j) \cap R(v_{j-1})$, which is impossible since we cannot have both $j = i$ and $j - 1 = i$. We have thus established the following fact, which will be used several times.

Fact 2.2. *Let $(v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1})$ be a good tuple, let $z \in \{v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1}\}$ and let $0 \leq i \leq k-1$. If $z \in R(v_i) \cap R(v_{i-1})$ then $z = y_i$, and if $y_i, y_{i+1} \in R(z)$ then $z = v_i$.*

For vertices $a, b \in V(H) = V(\vec{H})$, denoted by $\overrightarrow{\text{dist}}(a, b)$ the length of a shortest directed path from a to b (in $\vec{H}$). Now, fix a good $2k$-tuple $M = (v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1})$ which minimizes the sum

$$\sum_{i=0}^{k-1} \left(\overrightarrow{\text{dist}}(v_i, y_i) + \overrightarrow{\text{dist}}(v_{i-1}, y_i) \right). \tag{5}$$

Let us now fix specific shortest (directed) paths $P(v_i, y_j)$ from v_i to y_j for $0 \leq i \leq k-1$ and $j = i, i+1$ (as always, indices are taken modulo k). We will denote by $P\{v_i, y_j\}$ the underlying undirected path of $P(v_i, y_j)$. Let C be the (undirected) closed walk obtained by concatenating the paths

$$P\{y_0, v_0\}, P\{v_0, y_1\}, P\{y_1, v_1\}, \dots, P\{v_{k-2}, y_{k-1}\}, P\{y_{k-1}, v_{k-1}\}, P\{v_{k-1}, y_0\}. \tag{6}$$

We now show that C is a simple cycle. We will then show that C is induced. While the proof idea is rather simple, the details are somewhat lengthy.

Two paths appearing consecutively (in a cyclic manner) in Equation (6) will be called consecutive. In other words, the pairs of consecutive paths are $(P\{y_i, v_i\}, P\{v_i, y_{i+1}\})$ and $(P\{v_i, y_{i+1}\}, P\{y_{i+1}, v_{i+1}\})$ (for $0 \le i \le k-1$). If, by contradiction, C is not a simple cycle, then either there is a pair of non-consecutive paths which intersect or a pair of consecutive paths which intersect outside of the endpoint they share. We now rule out each of these possibilities.

Case 1: We consider the intersection of $P(v_i, y_i)$ and $P(v_i, y_{i+1})$ for some $0 \le i \le k-1$. Assume that there exists a vertex $z \ne v_i$ such that $z \in P(v_i, y_i) \cap P(v_i, y_{i+1})$. Then $y_i, y_{i+1} \in R(z)$. By Fact 2.2, we have $z \notin \{v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1}\}$. Now, replacing v_i with z in the tuple M, we get a new $2k$-tuple of distinct vertices which is also good. Indeed, there are paths from z to y_i, y_{i+1}, and there is no path from z to y_j for all $j \ne i, i+1$, as otherwise we would have a path from v_i to y_j (via z), contradicting the assumption. Since $z \ne v_i$, the two new paths we get by replacing v_i with z are strictly shorter than the two original ones, which contradicts the minimality of Equation (5).

Case 2: We consider the intersection of $P(v_{i-1}, y_i)$ and $P(v_i, y_i)$ for some $0 \le i \le k-1$. Assume that there exists a vertex $z \ne y_i$ such that $z \in P(v_{i-1}, y_i) \cap P(v_i, y_i)$. Then $z \in R(v_i) \cap R(v_{i-1})$. By Fact 2.2, $z \notin \{v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1}\}$. Now, replacing y_i with z in the tuple M, we get a new $2k$-tuple of distinct vertices which is also good. Indeed, there are paths from v_{i-1}, v_i to z, and there is no path from v_j to z for any $j \ne i, i-1$, as otherwise, we would have a path from v_j to y_i (via z), which contradicts the assumption. Since $z \ne y_i$, the two new paths we get by replacing y_i with z are strictly shorter than the two original ones, which contradicts the minimality of Equation (5).

Case 3: We consider the intersection of $P(v_i, y_s)$ and $P(v_j, y_t)$ for some $i \ne j$ and $s \ne t$ (where $s \in \{i, i+1\}$ and $t \in \{j, j+1\}$). Assume that there exists a vertex z such that $z \in P(v_i, y_s) \cap P(v_j, y_t)$. In this case, as there are paths from v_i and v_j to y_s, y_t (via z), we must have $\{s, t\} = \{i, i+1\} = \{j, j+1\}$, which implies that $i = j$ (as $k \ge 3$), in contradiction to the assumption that $i \ne j$.

From Cases 1–3 it follows that the closed walk C is indeed a simple cycle in H. We now prove that C is an induced cycle. We first observe that for a path $P(v_i, y_j)$ with $j \in i, i+1$ and two vertices $z_1, z_2 \in P(v_i, y_j)$ such that z_2 comes after z_1 along the path, we cannot have the edge (z_2, z_1) as $\vec{H}$ is acyclic. In addition, we cannot have the edge (z_1, z_2), unless it is an edge of the path, as this would contradict the fact that $P(v_i, y_j)$ is a shortest path from v_i to y_j. Thus, the (undirected) path $P\{v_i, y_j\}$ is induced. We conclude that if C has a chord, then it must connect two distinct paths among the paths in Equation (6). We now rule out the existence of such a chord by analyzing several cases.

Case 1: Consider an edge between the two paths $P(v_i, y_i)$ and $P(v_i, y_{i+1})$ for some $0 \le i \le k-1$. We assume, without loss of generality, that there is an edge $(z_1, z_2) \in E(\vec{H})$ such that $z_1 \in P(v_i, y_i)$, $z_2 \in P(v_i, y_{i+1})$, and $z_1, z_2 \ne v_i$. Then $y_i \in R(z_1)$ and $y_{i+1} \in R(z_2) \subseteq R(z_1)$. By Fact 2.2, $z_1 \notin \{v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1}\}$. Now, replacing v_i with z_1 in the tuple M, we get a new $2k$-tuple of distinct vertices which is also good. Indeed, there are paths from z_1 to y_i, y_{i+1}, and there is no path from z_1 to y_j for all $j \ne i, i+1$, as otherwise we would have a path from v_i to y_j (via z_1), which is impossible. Since $z_1, z_2 \ne v_i$, we have $\overrightarrow{\text{dist}}(z_1, y_i) + \overrightarrow{\text{dist}}(z_1, y_{i+1}) < \overrightarrow{\text{dist}}(v_i, y_i) + \overrightarrow{\text{dist}}(v_i, y_{i+1})$, which contradicts the minimality of Equation (5).

Case 2: Consider an edge between the two paths $P(v_{i-1}, y_i)$ and $P(v_i, y_i)$ for some $0 \le i \le k-1$. We assume, without loss of generality, that there is an edge $(z_1, z_2) \in E(\vec{H})$ such that $z_1 \in P(v_{i-1}, y_i)$, $z_2 \in P(v_i, y_i)$, and $z_1, z_2 \ne y_i$. Then $z_2 \in R(v_i) \cap R(z_1) \subseteq R(v_i) \cap R(v_{i-1})$. By Fact 2.2, $z_2 \notin \{v_0, \dots, v_{k-1}, y_0, \dots, y_{k-1}\}$. Now, replacing y_i with z_2 in the tuple M, we get a new $2k$-tuple of distinct vertices which is also good. Indeed, there are paths from v_{i-1}, v_i to z_2, and there is no path from v_j to z_2 for all $j \ne i, i-1$, as otherwise we would have a path from v_j to y_i (via z_2), which is impossible. Since $z_1, z_2 \ne y_i$, we have $\overrightarrow{\text{dist}}(v_i, z_2) + \overrightarrow{\text{dist}}(v_{i-1}, z_2) < \overrightarrow{\text{dist}}(v_i, y_i) + \overrightarrow{\text{dist}}(v_{i-1}, y_i)$, which contradicts the minimality of (5).

Case 3: Consider an edge between the two paths $P(v_i, y_s)$ and $P(v_j, y_t)$ for some $0 \leq i \neq j \leq k-1$ and $s \neq t$ (where $s \in \{i, i+1\}$ and $t \in \{j, j+1\}$). We assume, without loss of generality, that there is an edge $(z_1, z_2) \in E(\vec{H})$ such that $z_1 \in P(v_i, y_s)$ and $z_2 \in P(v_j, y_t)$. In this case, as there is a path from v_i to y_t (via z_1, z_2), we must have $\{s, t\} = \{i, i+1\}$. In addition, as $t \in \{j, j+1\}$, we either have that $s = i, t = i+1$, which implies $t = j$ (as $i \neq j$); or that $s = i+1, t = i$, which implies $t = j+1$ (as $i \neq j$).

We first consider the case when $s = i, t = i+1$, and $j = i+1$. In other words, we are considering the situation where there is an edge $(z_1, z_2) \in E(\vec{H})$ from $z_1 \in P(v_i, y_i)$ to $z_2 \in P(v_{i+1}, y_{i+1})$. Note that $y_i \in R(z_1)$ and $y_{i+1} \in R(z_2) \subseteq R(z_1)$. By Fact 2.2, either $z_1 = v_i$ or $z_1 \notin \{v_0, \ldots, v_{k-1}, y_0, \ldots, y_{k-1}\}$. Similarly, note that $z_2 \in R(z_1) \cap R(v_{i+1}) \subseteq R(v_i) \cap R(v_{i+1})$, so by Fact 2.2 either $z_2 = y_{i+1}$ or $z_2 \notin \{v_0, \ldots, v_{k-1}, y_0, \ldots, y_{k-1}\}$. Furthermore, we cannot have both $z_1 = v_i$ and $z_2 = y_{i+1}$, because then z_1, z_2 are both contained in the path $P(v_i, y_{i+1})$, and we have already ruled out the possibility of such a chord.

Now, replacing v_i with z_1 and y_{i+1} with z_2 in the tuple M, we get a new $2k$-tuple of distinct vertices which is also good. Indeed, there are paths from z_1 to y_i, z_2, and there is no path from z_1 to y_j for all $j \neq i, i+1$, as otherwise we would have a path from v_i to y_j (via z_1), which contradicts the assumption. Similarly, there are paths from z_1, v_{i+1} to z_2, and there is no path from v_j to z_2 for all $j \neq i, i+1$, as otherwise, we would have a path from v_j to y_{i+1} (via z_2), which contradicts the assumption. Since either $z_1 \neq v_i$ or $z_2 \neq y_{i+1}$, we have $\overrightarrow{\text{dist}}(z_1, y_i) + \overrightarrow{\text{dist}}(z_1, z_2) + \overrightarrow{\text{dist}}(v_{i+1}, z_2) < \overrightarrow{\text{dist}}(v_i, y_i) + \overrightarrow{\text{dist}}(v_i, y_{i+1}) + \overrightarrow{\text{dist}}(v_{i+1}, y_{i+1})$, which contradicts the minimality of (5).

We now consider the (symmetrical) case when $s = i+1, t = i$, and $j = i-1$. In other words, we are considering the situation where there is an edge $(z_1, z_2) \in E(\vec{H})$ from $z_1 \in P(v_i, y_{i+1})$ to $z_2 \in P(v_{i-1}, y_i)$. Note that $y_{i+1} \in R(z_1)$ and $y_i \in R(z_2) \subseteq R(z_1)$. By Fact 2.2, either $z_1 = v_i$ or $z_1 \notin \{v_0, \ldots, v_{k-1}, y_0, \ldots, y_{k-1}\}$. Similarly, note that $z_2 \in R(z_1) \cap R(v_{i-1}) \subseteq R(v_i) \cap R(v_{i-1})$, so by Fact 2.2 either $z_2 = y_i$ or $z_2 \notin \{v_0, \ldots, v_{k-1}, y_0, \ldots, y_{k-1}\}$. Furthermore, we cannot have both $z_1 = v_i$ and $z_2 = y_i$, because then z_1, z_2 are both contained in the path $P(v_i, y_i)$, and we have already ruled out the possibility of such a chord.

Now, replacing v_i with z_1 and y_i with z_2 in the tuple M, we get a new $2k$-tuple of distinct vertices which is also good. Indeed, there are paths from z_1 to z_2, y_{i+1}, and there is no path from z_1 to y_j for all $j \neq i, i+1$, as otherwise we would have a path from v_i to y_j (via z_1), which contradicts the assumption. Similarly, there are paths from z_1, v_{i-1} to z_2, and there is no path from v_j to z_2 for all $j \neq i, i-1$, as otherwise, we would have a path from v_j to y_i (via z_2), which contradicts the assumption. Since either $z_1 \neq v_i$ or $z_2 \neq y_i$, we have $\overrightarrow{\text{dist}}(z_1, y_{i+1}) + \overrightarrow{\text{dist}}(z_1, z_2) + \overrightarrow{\text{dist}}(v_{i-1}, z_2) < \overrightarrow{\text{dist}}(v_i, y_{i+1}) + \overrightarrow{\text{dist}}(v_i, y_i) + \overrightarrow{\text{dist}}(v_{i-1}, y_i)$, which contradicts the minimality of (5).

Cases 1–3 imply that C is an induced cycle. The length of C is evidently at least $2k \geq 6$. This completes the proof of the lemma. □

To prove Theorem 3, we combine Lemma 2.1 with the following theorem, which gives a structural characterization of α-acyclic hypergraphs. The proof of this theorem can be found in [5, 6].

Theorem 8 ([5, 6]). *A hypergraph F is α-acyclic if and only if for every $k \geq 3$, there is* no *$S = \{x_0, x_1, \ldots, x_{k-1}\} \subseteq V(F)$ such that one of the following conditions holds:*

(1) *For every $0 \leq i \leq k-1$ there exists $e \in E(F)$ such that $e \cap S = \{x_i, x_{i+1}\}$, and there is no $e \in E(F)$ with $|e \cap S| \geq 2$ such that $e \cap S \neq \{x_i, x_{i+1}\}$ for all $0 \leq i \leq k-1$. (All indices are taken modulo k.)*

(2) *For every $0 \leq i \leq k-1$ there exists $e \in E(F)$ such that $e \cap S = S \backslash \{x_i\}$, and there is no $e \in E(F)$ such that $S \subseteq e$.*

We are now ready to prove Theorem 3.

Proof of Theorem 3. We first prove the "only if" part of the theorem, or, more precisely, the contrapositive of this statement. Suppose that H contains an induced copy of C_ℓ for some $\ell \geq 6$, and let $C = (c_0, c_1, \ldots, c_{\ell-1})$ be such a cycle. We now orient the edges of H as follows: the edges of C are oriented alternatingly along the cycle. More precisely, for an even $0 \leq i \leq \ell - 1$, we orient the edge $\{c_i, c_{i+1}\} \in E(H)$ from c_i to c_{i+1}, and for an odd $0 \leq i \leq \ell - 1$, we orient the edge $\{c_i, c_{i+1}\} \in E(H)$ from c_{i+1} to c_i (with indices taken modulo ℓ). Now, orient all edges between C and $V(H)\backslash C$ from C to $V(H)\backslash C$, and orient all edges in $V(H)\backslash C$ arbitrarily while avoiding the creation of a directed cycle. We denote the resulting orientation of H by $\vec{H}$. One can easily verify that $\vec{H}$ is acyclic. Indeed, by our construction, the induced subgraphs $\vec{H}[C]$ and $\vec{H}[V(H)\backslash C]$ do not contain directed cycles, and all the edges between C and $V(H)\backslash C$ go in the same direction. Let $S = \{c_j \mid j \text{ is odd}\}$, noting that $|S| \geq \lfloor \ell/2 \rfloor \geq 3$. Observe that if ℓ is even, then the vertices c_i for even $0 \leq i \leq \ell - 1$ are sources, and $R(c_i) \cap S = \{c_{i-1}, c_{i+1}\}$ for each such i. If ℓ is odd, then the sources are the vertices c_i with even $2 \leq i \leq \ell - 1$; also, $R(c_i) \cap S = \{c_{i-1}, c_{i+1}\}$ for each even $2 \leq i \leq \ell - 3$, and $R(c_{\ell-1}) \cap S = \{c_{\ell-2}, c_1\}$, as $R(c_{\ell-1}) \cap V(C) = \{c_{\ell-2}, c_0, c_1\}$. Observe also that for every source vertex u of $\vec{H}$ which is not in C, we have that $R(u) \cap S = \emptyset$. So we see that in both cases (ℓ even or odd), the situation in Item 1 of Theorem 8 holds with respect to the above-chosen S. Hence, by Theorem 8, $\vec{H}$ is not α-acyclic. Thus, H has an acyclic orientation which is not α-acyclic, as required.

We now establish the "if" part of the theorem. Again, we will prove the contrapositive. Suppose that there exists an acyclic orientation $\vec{H}$ of H which is not α-acyclic. Let $F_{\vec{H}}$ be the hypergraph as in Definition 1.5. Then $F_{\vec{H}}$ is not α-acyclic. Hence, by Theorem 8, there exists $S = \{x_0, x_1, \ldots, x_{k-1}\} \subseteq V(F_{\vec{H}})$ with $k \geq 3$ such that (at least) one of the conditions 1–2 in that theorem holds with respect to S. Our goal is to show that this implies the existence of an induced ℓ-cycle in H for some $\ell \geq 6$.

Assume first that S satisfies Condition 1 of Theorem 8. For each $0 \leq i \leq k - 1$, let $e_i \in E(F_{\vec{H}})$ be such that $e_i \cap S = \{x_i, x_{i+1}\}$. By the definition of $F_{\vec{H}}$, for each $0 \leq i \leq k-1$ there is a source u_i of $\vec{H}$ such that $e_i = R(u_i)$. So for every $0 \leq i \leq k - 1$, we have that $x_i \in R(u_{i-1}) \cap R(u_i)$ and $x_i \notin R(u_j)$ for all $j \neq i, i - 1$. Moreover, $u_0, \ldots, u_{k-1}, x_0, \ldots, x_{k-1}$ are pairwise-distinct because $x_0, \ldots, x_{k-1}$ are pairwise-distinct, $u_0, \ldots, u_{k-1}$ are pairwise-distinct, and $u_0, \ldots, u_{k-1}$ are sources of $\vec{H}$ while $x_0, \ldots, x_{k-1}$ are not. Therefore, we can apply Lemma 2.1 and get that H contains an induced copy of C_ℓ for some $\ell \geq 6$, as required.

Now assume that S satisfies Condition 2 of Theorem 8. For each $0 \leq i \leq k - 1$, let $e_i \in E(F_{\vec{H}})$ be such that $e_i \cap S = S\backslash\{x_i\}$. By the definition of $F_{\vec{H}}$, for each $0 \leq i \leq k-1$ there is a source u_i of $\vec{H}$ such that $e_i = R(u_i)$. Considering only $R(u_0), R(u_1), R(u_2)$, and setting $y_0 := x_1, y_1 := x_2, y_2 := x_0$, we have that $y_0 \in R(u_2) \cap R(u_0)$ but $y_0 \notin R(u_1)$, $y_1 \in R(u_0) \cap R(u_1)$ but $y_1 \notin R(u_2)$, and $y_2 \in R(u_1) \cap R(u_2)$ but $y_2 \notin R(u_0)$. Furthermore, just like in the previous case, $u_0, u_1, u_2, y_0, y_1, y_2$ are pairwise-distinct. Therefore, we can again apply Lemma 2.1 and get that H contains an induced copy of C_ℓ for some $\ell \geq 6$, as required. □

3 HARDNESS RESULTS FOR HOM-CNT$_{C_k}$: PROOF OF LEMMA 1.6

The proof of Lemma 1.6 uses so-called colorful subgraph counts. Let H be a graph on h vertices. For a graph G and a coloring $c : V(G) \rightarrow [h]$ of its vertices, a homomorphism $\varphi : H \rightarrow G$ is called *colorful* if every color appears exactly once in the image of φ, namely, if $\{c(\varphi(v)) : v \in V(H)\} = [h]$. (Note that the number of colors is always the number of vertices in H.) Evidently, a colorful homomorphism must be injective. We denote by col-inj(H, G, c) the number of colorful homomorphisms from H to G, and by COL-INJ-CNT$_H$ the problem of computing the number of colorful homomorphisms of H in an h-vertex-colored input graph. We use the known fact that COL-INJ-CNT$_H$ reduces to HOM-CNT$_H$ (see e.g., [37]). For completeness, we include a proof.

LEMMA 3.1. *If one can solve* HOM-CNT$_H$ *in time $f(n)$ in n-vertex graphs then one can also solve* COL-INJ-CNT$_H$ *in time $O(f(n))$ in n-vertex graphs. Moreover, the reduction preserves degeneracy.*

PROOF. Suppose we are given a graph G and a coloring $c : V(G) \to [h]$, and wish to compute the number of colorful H-homomorphisms in G. For each set $I \subseteq [h]$, let V_I be the set of vertices of G whose color belongs to I. By definition, a homomorphism $\varphi : H \to G$ is colorful if and only if $c(\text{Image}(\varphi)) = [h]$. For every $I \subseteq [h]$, the number of homomorphisms $\varphi : V(H) \to V(G)$ such that $c(\text{Image}(\varphi)) \subseteq I$ is exactly $\text{hom}(H, G[V_I])$. So by inclusion-exclusion, we have

$$\text{col-inj}(H, G, c) = \sum_{I \subseteq [h]} (-1)^{|I|} \cdot \text{hom}(H, G[V_I]).$$

So by computing $\text{hom}(H, G[V_I])$ for each $I \subseteq [h]$, we can recover $\text{col-inj}(H, G, c)$. Evidently, each of the graphs $G[V_I]$ has at most n vertices and its degeneracy is at most that of G. This completes the proof. □

We will also need the fact that detecting a colorful triangle in a 3-vertex-colored graph is at least as hard as uncolored triangle detection.

LEMMA 3.2. *If one can detect a colorful triangle in a* 3*-vertex-colored graph with m edges in time $f(m)$, then one can also detect a triangle in an uncolored graph with m edges in time $\tilde{O}(f(m))$.*

PROOF. This follows from the color-coding method, see [2]. Suppose we are given an n-vertex graph G and need to decide if G is triangle-free. Color the vertices of G randomly with three colors. If G contains a triangle, then the resulting 3-vertex-colored graph will contain a colorful triangle with constant positive probability. Hence, if there is a $f(n)$-time algorithm for colorful triangle detection, then there is also a $O(f(n))$-time randomized algorithm for (uncolored) triangle detection which has a constant success probability. Using color-coding [2], this can be derandomized at the cost of a logarithmic factor, thus giving the desired $\tilde{O}(f(n))$-time triangle detection algorithm. □

The following lemma constitutes the main step in the proof of Lemma 1.6.

LEMMA 3.3. *Let $k \geq 4$. For every graph G and coloring $c : V(G) \to [3]$, one can construct in time $O(|V(G)| + |E(G)|)$ a graph G' with $|V(G')|, |E(G')| = O(|V(G)| + |E(G)|)$ and a coloring $c' : V(G') \to [k]$, such that* $\text{col-inj}(C_k, G', c') = \frac{k}{3} \cdot \text{col-inj}(C_3, G, c)$*. Moreover, if $k \geq 6$ then G' is* 2*-degenerate.*

PROOF. Fix numbers $\ell_{1,2}, \ell_{1,3}, \ell_{2,3} \in \{\lfloor k/3 \rfloor - 1, \lceil k/3 \rceil - 1\}$ such that $\ell_{1,2} + \ell_{1,3} + \ell_{2,3} = k - 3$. Fix a partition $\{4, \ldots, k\} = I_{1,2} \cup I_{1,3} \cup I_{2,3}$ where $|I_{i,j}| = \ell_{i,j}$ for each pair $1 \leq i < j \leq 3$. We define G' and c' as follows. For each edge $\{x, y\} \in E(G)$, if $c(x) \neq c(y)$, say $i = c(x)$ and $j = c(y)$, $1 \leq i \neq j \leq 3$, then replace the edge $\{x, y\}$ with a path of length $\ell_{i,j} + 1$ (between x and y), and color the $\ell_{i,j}$ internal vertices of this path with the $\ell_{i,j}$ colors in $I_{i,j}$. If $c(x) = c(y)$ then simply remove the edge $\{x, y\}$. To complete the definition of c', we set $c'(v) = c(v) \in [3]$ for every $v \in V(G) \subseteq V(G')$. The resulting graph is G' and the resulting coloring is c'. Note that the vertices in $V(G)$ are colored with $1, 2, 3$, while the vertices in $V(G') \backslash V(G)$ are colored with $4, \ldots, k$. We have $|V(G')| \leq |V(G)| + (\lceil k/3 \rceil - 1) \cdot |E(G)| = O(|V(G)| + |E(G)|)$ and $|E(G')| \leq \lceil k/3 \rceil \cdot |E(G)| = O(|V(G)| + |E(G)|)$. Moreover, if $k \geq 6$ then G' is 2-degenerate. Indeed, observe that the original vertices of G form an independent set in G', because each edge of G is either removed or replaced by a path of length at least $\lfloor k/3 \rfloor \geq 2$. Also, all vertices in $V(G') \backslash V(G)$ have degree 2 in G'. Upon removing these vertices, we are left with an empty graph; hence G' is 2-degenerate, as claimed.

Let us now consider the colorful k-cycles in G'. It is easy to see that every cycle C in G' corresponds to a cycle in G, in the sense that there is a cycle $x_1, x_2, \dots, x_r, x_1$ in G such that C consists of $x_1, \dots, x_r$ and of the paths which replaced the edges $\{x_1, x_2\}, \dots, \{x_{r-1}, x_r\}, \{x_r, x_1\}$. Now, if C is a colorful k-cycle, then it must be the case that $r = 3$ and that the triangle x_1, x_2, x_3 is colorful. Indeed, if C is colorful then $x_1, \dots, x_r$ must have distinct colors, which is impossible if $r \geq 4$ (as the vertices of G are colored with $1, 2, 3$). In the other direction, the definition of G' and c' implies that every colorful triangle in G gives rise to a colorful k-cycle in G'. It is now easy to see that $\text{col-inj}(C_k, G', c') = \frac{k}{3} \cdot \text{col-inj}(C_3, G, c)$, as required □

Lemma 1.6 now follows easily from the above lemmas:

Proof of Lemma 1.6. Let $k \geq 6$, and suppose that HOM-CNT_{C_k} can be solved in time $\tilde{O}(n)$ in 2-degenerate n-vertex graphs. By Lemma 3.1, one can also solve COL-INJ-CNT_{C_k} in time $\tilde{O}(n)$ in 2-degenerate k-vertex-colored n-vertex graphs. One can then use the construction given by Lemma 3.3 to obtain an algorithm for colorful triangle detection in (3-vertex-colored) m-edge graphs, running in time $\tilde{O}(m)$. Using Lemma 3.2, one obtains an algorithm for (uncolored) triangle detection, which runs in time $\tilde{O}(m)$ on m-edge graphs. The existence of such an algorithm is ruled out by Conjecture 1.1. □

Using the same argument, one obtains the following statement regarding the hardness of counting C_k-homomorphisms in general (i.e., not necessarily degenerate) graphs.

Lemma 3.4. *Let $k \geq 4$. Assuming Conjecture 1.1, HOM-CNT_{C_k} cannot be solved in time $\tilde{O}(|V(G)| + |E(G)|)$.*

4 SOLVABILITY OF HOM-CNT$_H$ IS HEREDITARY: PROOF OF LEMMA 1.7

In this section, we prove Lemma 1.7. We will actually prove the following more general statement.

Lemma 4.1. *For every graph H there is $k = k(H)$ such that the following holds. For every graph G there are graphs $G_1, \dots, G_k$, 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 $i = 1, \dots, k$, and such that knowing $\hom(H, G_1), \dots, \hom(H, G_k)$ allows one to compute $\hom(H', G)$ for all induced subgraphs H' of H in time $O(1)$. Furthermore, if G is $O(1)$-degenerate, then so are $G_1, \dots, G_k$.*

It is easy to see that Lemma 4.1 implies Lemma 1.7.

We now state an important lemma concerned with computing *linear combinations of homomorphism counts*. This is implicit in [19]. As mentioned in the introduction, several of the reductions presented in this article—including the proof of Lemma 4.1—rely on it.

Lemma 4.2. *Let $H_1, \dots, H_k$ be pairwise non-isomorphic graphs and let $c_1, \dots, c_k$ be non-zero constants. For every graph G there are graphs $G_1, \dots, G_k$, 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 $i = 1, \dots, k$, and such that knowing $b_j := c_1 \cdot \hom(H_1, G_j) + \cdots + c_k \cdot \hom(H_k, G_j)$ for every $j = 1, \dots, k$ allows one to compute $\hom(H_1, G), \dots, \hom(H_k, G)$ in time $O(1)$. Furthermore, if G is $O(1)$-degenerate, then so are $G_1, \dots, G_k$.*

For completeness, we give a proof of Lemma 4.2 in the appendix.

In the proof of Lemma 4.1, it will be convenient to consider the empty graph K_0, and to define $\hom(K_0, G) = 1$ for every graph G.

Proof of Lemma 4.1. Let H be a graph on h vertices. For a graph F, we denote by $F + K_h$ the graph obtained from F by adding a clique of size h and connecting it to $V(F)$ with a complete bipartite graph. We start by showing that for every graph F, it holds that

$$\hom(H, F + K_h) = \sum_{U \subseteq V(H)} \hom(H[U], F) \cdot \hom(H[V(H)\backslash U], K_h). \tag{7}$$

To see that Equation (7) holds, let us assign to each function $\varphi : V(H) \to V(F + K_h)$ the set $U = U(\varphi) := \varphi^{-1}(V(F)) \subseteq V(H)$. Our definition of the graph $F + K_h$ guarantees that a function $\varphi : V(H) \to V(F + K_h)$ is a homomorphism if and only if $\varphi|_{U(\varphi)}$ is a homomorphism from $H[U]$ to F and $\varphi|_{V(H)\backslash U(\varphi)}$ is a homomorphism from $H[V(H)\backslash U(\varphi)]$ to K_h. By summing over all possible values of U, we get Equation (7). Note that $U(\varphi)$ may be empty (in case $\mathrm{Im}(\varphi) \subseteq K_h$); in this case $\hom(H[U], F) = 1$ (as $H[U]$ is the empty graph).

Let $H_1, H_2, \ldots, H_k$ be an enumeration of all induced subgraphs of H (including the empty one), up to isomorphism (that is, $H_1, \ldots, H_k$ are pairwise non-isomorphic). For each $1 \le i \le k$, set

$$c_i := \sum_{\substack{U \subseteq V(H): \\ H[U] \cong H_i}} \hom(H[V(H)\backslash U], K_h).$$

Note that $c_1, \ldots, c_k$ depend only on H. With this notation, we can rewrite Equation (7) as follows:

$$\hom(H, F + K_h) = \sum_{i=1}^{k} c_i \cdot \hom(H_i, F). \tag{8}$$

Note that for each $1 \le i \le k$ we have $c_i > 0$, since there is some $U \subseteq V(H)$ for which $H[U] \cong H_i$ (by our choice of $H_1, \ldots, H_k$), and for this U it clearly holds that $\hom(H[V(H)\backslash U], K_h) > 0$.

Now let G be a graph. Apply Lemma 4.2 (to the graph G), and let $G'_1, \ldots, G'_k$ be the graphs given by that lemma. For each $1 \le i \le k$, set $G_i := G'_i + K_h$, noting that $|V(G_i)| = |V(G'_i)| + h = O(|V(G)| + |E(G)|)$ and $|E(G_i)| = |E(G'_i)| + |V(G'_i)| \cdot h + \binom{h}{2} = O(|V(G)| + |E(G)|)$, where in both cases the last equality is guaranteed by Lemma 4.2. Furthermore, if G is $O(1)$-degenerate then so is G'_i for every $1 \le i \le k$ (by Lemma 4.2), and hence so is G_i for every $1 \le i \le k$ (indeed, if a graph F is κ-degenerate then $F + K_h$ is $(\kappa + h)$-degenerate). Crucially, observe that knowing $\hom(H, G_1), \ldots, \hom(H, G_k)$ allows one to compute $c_1 \cdot \hom(H_1, G'_j) + \cdots + c_k \cdot \hom(H_k, G'_j)$ for every $1 \le j \le k$ (by Equation (8)), which in turn allows one to compute $\hom(H_1, G), \ldots, \hom(H_k, G)$ in time $O(1)$ (by our choice of $G'_1, \ldots, G'_k$ via Lemma 4.2). This completes the proof, as every induced subgraph of H is isomorphic to one of the graphs $H_1, \ldots, H_k$. $\square$

5 COUNTING COPIES: PROOF OF LEMMA 1.8

We prove the following more general lemma, from which Lemma 1.8 will easily follow.

Lemma 5.1. *For every graph H there is $k = k(H)$ such that the following holds. For every graph G there are graphs $G_1, \ldots, G_k$, 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 $i = 1, \ldots, k$, and such that knowing $\mathrm{inj}(H, G_1), \ldots, \mathrm{inj}(H, G_k)$ allows one to compute $\hom(H/P, G)$ for all partitions P of $V(H)$ in time $O(1)$. Furthermore, if G is $O(1)$-degenerate, then so are $G_1, \ldots, G_k$.*

In the proof, we will need the value (or rather, the sign) of the Möbius function of the partition poset (which plays a role in Equation (2)). This Möbius function is given by the Frucht–Rota–Schützenberger formula (see, e.g., [47, Chapter 25]): for a partition P of the vertex-set of H, we have

$$\mu_{\text{part}}(P) = (-1)^{|V(H)|-|P|} \cdot \prod_{U \in P} (|U| - 1)!. \tag{9}$$

Proof of Lemma 5.1. The lemma follows easily by combining Equation (2) and Lemma 4.2. Let $H_1, \dots, H_k$ be an enumeration of all quotient graphs of H, up to isomorphism. That is, $H_1, \dots, H_k$ are pairwise non-isomorphic and $\{H_1, \dots, H_k\} = \{H/P : P \in \mathcal{P}(H)\}$, where, as before, $\mathcal{P}(H)$ denotes the set of all partitions of $V(H)$. Let G be a graph. By using (2) and "combining like terms", we get that

$$\mathrm{inj}(H,G) = \sum_{P \in \mathcal{P}(H)} \mu_{\mathrm{part}}(P) \cdot \hom(H/P, G) = \sum_{i=1}^{k} \left(\sum_{\substack{P \in \mathcal{P}(H): \\ H/P \cong H_i}} \mu_{\mathrm{part}}(P) \right) \cdot \hom(H_i, G). \tag{10}$$

From Equation (9) we know that $\mu_{\mathrm{part}}(P) \neq 0$ for all $P \in \mathcal{P}(H)$, and that the sign of $\mu_{\mathrm{part}}(P)$ depends only on the number of parts in P. In particular, if $P, Q \in \mathcal{P}(H)$ are such that $H/P \cong H/Q$, then $\mu_{\mathrm{part}}(P)$ and $\mu_{\mathrm{part}}(Q)$ have the same sign. Now, setting

$$c_i := \sum_{\substack{P \in \mathcal{P}(H): \\ H/P \cong H_i}} \mu_{\mathrm{part}}(P),$$

we see that c_i is non-zero since the summands in the above sum cannot cancel each other. With this notation, Equation (10) becomes

$$\mathrm{inj}(H,G) = \sum_{i=1}^{k} c_i \cdot \hom(H_i, G). \tag{11}$$

Now the lemma immediately follows from Equation (11) and Lemma 4.2, as every quotient graph of H is isomorphic to one of the graphs $H_1, \dots, H_k$. □

Proof of Lemma 1.8. The "if" part of the lemma follows from Equation (2), and the "only-if" part of the lemma follows from Lemma 5.1. □

6 COUNTING INDUCED COPIES: PROOF OF THEOREM 6 AND LEMMA 1.9

Before establishing Lemma 1.9, we again prove a more general statement.

Lemma 6.1. *For every graph H there is $k = k(H)$ such that the following holds. For every graph G there are graphs $G_1, \dots, G_k$, 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 $i = 1, \dots, k$, and such that the following holds:*

(1) *Knowing $\mathrm{ind}(H, G_i)$ for every $1 \le i \le k$ allows one to compute $\hom(H', G)$ for all supergraphs H' of H in time $O(1)$.*
(2) *Knowing $\hom(H', G_i)$ for every supergraph H' of H and every $1 \le i \le k$, allows one to compute $\mathrm{inj}(H', G)$ for all supergraphs H' of H in time $O(1)$.*

Furthermore, if G is $O(1)$-degenerate, then so are $G_1, \dots, G_k$.

Proof. We will show that there are $G_1, \dots, G_k$ which satisfy the assertion of each of the items 1–2 separately. One can then take the union of these two families of graphs to obtain the desired graphs $G_1, \dots, G_k$ for which both items hold.

Starting with Item 2, we begin by proving the following preliminary claim: for every supergraph H' of H and every partition P of $V(H)$, there is a supergraph H'' of H such that H'/P is an induced subgraph of H''. Indeed, fixing H' and P as above, we define H'' as follows: for each $\{U, V\} \in E(H'/P)$ (so $U, V \in P$), add to H all edges between U and V; the resulting graph is H''. It is easy to see that H'/P is indeed an induced subgraph of H'', as required.

Now let G be a graph. By combining the above claim with Lemma 4.1 (which we apply to all supergraphs H' of H at once), we see that there are graphs $G_1, \ldots, G_k$ with $|V(G_i)| = O(|V(G)|)$ and $|E(G_i)| = O(|E(G)|)$ for every $i = 1, \ldots, k$, such that knowing $\hom(H', G_i)$ for every supergraph H' of H and every $1 \leq i \leq k$, allows one to compute $\hom(H'/P, G)$ for all supergraphs H' of H and all partitions P of $V(H)$ in time $O(1)$. With this information at hand, one can use Equation (1) to compute $\mathrm{inj}(H', G)$ for all supergraphs H' of H in time $O(1)$, as required.

We now move on to establish Item 1. For convenience, we denote by $\mathcal{E}$ the set of all subsets of $\binom{V(H)}{2} \backslash E(H)$, and by $\mathcal{P}$ the set of all partitions of $V(H)$. We start by observing that for every graph F,

$$\begin{aligned} \mathrm{ind}(H, F) &= \sum_{E \in \mathcal{E}} (-1)^{|E|} \cdot \mathrm{inj}(H \cup E, F) \\ &= \sum_{E \in \mathcal{E}} \sum_{P \in \mathcal{P}} (-1)^{|E|} \cdot \mu_{\mathrm{part}}(P) \cdot \hom((H \cup E)/P, F), \end{aligned} \tag{12}$$

where the first equality uses Equation (4) and the second equality uses Equation (2). For an (unlabeled) graph H', put

$$c_{H'} := \sum_{\substack{E \in \mathcal{E},\ P \in \mathcal{P}: \\ (H \cup E)/P \cong H'}} (-1)^{|E|} \cdot \mu_{\mathrm{part}}(P). \tag{13}$$

With this notation, Equation (12) becomes

$$\mathrm{ind}(H, F) = \sum_{H' :\, c_{H'} \neq 0} c_{H'} \cdot \hom(H', F).$$

Let $H_1, \ldots, H_k$ be an enumeration of all graphs H' such that $c_{H'} \neq 0$, up to isomorphism. In other words, $H_1, \ldots, H_k$ are pairwise non-isomorphic and $\{H_1, \ldots, H_k\} = \{H' : c_{H'} \neq 0\}$. For each $1 \leq i \leq k$, we put $c_i := c_{H_i}$. Finally, we can write

$$\mathrm{ind}(H, F) = \sum_{i=1}^{k} c_i \cdot \hom(H_i, F). \tag{14}$$

Crucially, observe that every supergraph of H is isomorphic to one of the graphs $H_1, \ldots, H_k$. To see this, let H' be a (representative of the isomorphism class of a) supergraph of H. Clearly, if $E \in \mathcal{E}$ and $P \in \mathcal{P}$ are such that $(H \cup E)/P \cong H'$, then P is the partition of $V(H)$ into singletons, and $|E| = |E(H')| - |E(H)|$. Hence, all summands on the right-hand side of Equation (13) have the same sign, implying that $c_{H'} \neq 0$. This in turn implies that H' is isomorphic to one of the graphs $H_1, \ldots, H_k$, as required. Now Item 1 of the lemma immediately follows from Equation (14) and Lemma 4.2. □

Proof of Lemma 1.9. The implication $2 \Rightarrow 1$ follows from Equation (4), the implication $3 \Rightarrow 2$ follows from Item 2 of Lemma 6.1, and the implication $1 \Rightarrow 3$ follows from Item 1 of Lemma 6.1. □

Proof of Theorem 6. By putting together Lemma 1.9 and Corollary 4, we see that IND-CNT$_H$ is easy if and only if every supergraph of H is induced C_k-free for every $k \geq 6$ (with the "only if" part requiring Conjecture 1.1). Thus, to prove the theorem, it remains to observe that the following are equivalent:

(1) Every supergraph of H is induced C_k-free for every $k \geq 6$.
(2) H does not contain as an induced subgraph any (not necessarily induced) spanning subgraph of C_6.

The implication 1 $\Rightarrow$ 2 is immediate. For the reverse implication, observe that if some supergraph H' of H contains an induced copy of C_k (for some $k \geq 6$) then by adding an appropriate chord to this C_k, we obtain a supergraph H'' of H which contains an induced copy of C_6. But then H contains an induced copy of some spanning subgraph of C_6. This shows that 2 $\Rightarrow$ 1. □

7 PROOF OF THEOREM 7

The "if" part of Theorem 7 follows from the following proposition.

Proposition 7.1. *Let H be a forest. Then for every graph G,* $\hom(H, G)$ *can be computed in time* $\tilde{O}(|V(G)| + |E(G)|)$.

Proof. First, observe that if H is a disconnected graph with connected components $H_1, \ldots, H_k$, then $\hom(H, G) = \prod_{i=1}^{k} \hom(H_i, G)$ for every graph G. Thus, to prove the proposition, it suffices to consider the case that H is a tree.

Suppose then that H is a tree. To compute $\hom(H, G)$, we use dynamic programming to compute, for each subtree H' of H, $v \in V(H')$ and $x \in V(G)$, the number $N(H', v, x)$ of homomorphisms φ from H' to G such that $\varphi(v) = x$. To compute $N(H', v, x)$ for given H' and v (simultaneously for all $x \in V(G)$), we consider two cases according to whether or not v is a leaf of H'. Suppose first that v is a leaf of H', let u be the only neighbor of v in H', and set $H'' := H' - v$. Now, for each $x \in V(G)$, compute $N(H', v, x) = \sum_{y:\{x,y\} \in E(G)} N(H'', u, y)$ (where the sum is over all neighbours y of x). Suppose now that v is not a leaf, let $C_1, \ldots, C_k$ be the connected components of the forest $H' - v$, and set $H'_i := H'[C_i \cup \{v\}]$. Now, for each $x \in V(G)$, compute $N(H', v, x) = \prod_{i=1}^{k} N(H'_i, v, x)$. Thus, in both cases, one can compute $N(H', v, x)$ for all $x \in V(G)$ by relying on counts for smaller trees that have already been computed and stored. Retrieving or updating stored values can be done in time $\tilde{O}(1)$. □

Proof of Theorem 7. The "if" part of the theorem follows from Proposition 7.1. The "only if" part follows from the combination of Lemmas 3.4 and 4.1 (as any non-forest evidently contains an induced cycle). □

APPENDIX

A PROOF OF LEMMA 4.2

We start by recalling the definition of the tensor product of graphs. The tensor product of graphs G_1 and G_2, denoted $G_1 \times G_2$ has vertex set $V(G_1 \times G_2) = V(G_1) \times V(G_2)$ and edge-set

$$E(G_1 \times G_2) = \{\{(x_1, x_2), (y_1, y_2)\} : \{x_1, y_1\} \in E(G_1) \text{ and } \{x_2, y_2\} \in E(G_2)\}.$$

A key property of the tensor product is that the parameter $\hom(H, \cdot)$ is multiplicative with respect to it (for any graph H). That is, for every pair of graphs G_1, G_2, it holds that

$$\hom(H, G_1 \times G_2) = \hom(H, G_1) \cdot \hom(H, G_2). \tag{15}$$

To see that Equation (15) holds, simply observe that for functions $\varphi_i : V(H) \to V(G_i)$ (where $1 \leq i \leq 2$), the function $v \mapsto (\varphi_1(v), \varphi_2(v))$ is a homomorphism from H to $G_1 \times G_2$ if and only if φ_i is a homomorphism from H to G_i for each $1 \leq i \leq 2$. In what follows, we will use the following (trivial) observation regarding tensor products and degeneracy.

Observation A.1. *Let F, G be graphs. If G is κ-degenerate, then $F \times G$ is $(v(F) \cdot \kappa)$-degenerate.*

Proof. It is easy to see that for each $x \in V(F)$ and $y \in V(G)$, the degree of (x, y) in $F \times G$ is $d_{F \times G}((x, y)) = d_F(x) \cdot d_G(y) < v(F) \cdot d_G(y)$. It follows that every subgraph of $F \times G$ contains a vertex of degree at most $v(F) \cdot \kappa$ (since G is κ-degenerate). □

We now state a lemma of Erdős, Lovász and Spencer [24] (see also Proposition 5.44(b) in [33]), which will play a crucial role in the proof of Lemma 4.2.

Lemma A.2 ([33]). *Let $H_1, \ldots, H_k$ be pairwise non-isomorphic graphs, and let $c_1, \ldots, c_k \neq 0$ be non-zero constants. Then there exist graphs $F_1, \ldots, F_k$ such that the $k \times k$ matrix $M_{i,j} = c_j \cdot \hom(H_j, F_i)$, $1 \leq i, j \leq k$, is invertible.*

Finally, we are ready to prove Lemma 4.2.

Proof of Lemma 4.2. By Lemma A.2, there are graphs $F_1, \ldots, F_k$ such that the $k \times k$ matrix $M_{i,j} := c_j \cdot \hom(H_j, F_i)$ $(1 \leq i, j \leq k)$ is invertible. Given an input graph G, we set $G_i := F_i \times G$ and $b_i := c_1 \cdot \hom(H_1, G_i) + \cdots + c_k \cdot \hom(H_k, G_i)$ for each $1 \leq i \leq k$. Observe that

$$b_i = \sum_{j=1}^{k} c_j \cdot \hom(H_j, F_i \times G) = \sum_{j=1}^{k} c_j \cdot \hom(H_j, F_i) \cdot \hom(H_j, G) = \sum_{j=1}^{k} M_{i,j} \cdot \hom(H_j, G). \quad (16)$$

In the third equality above, we used Equation (15). We will treat Equation (16) $(1 \leq i \leq k)$ as a system of linear equations, where $\hom(H_1, G), \ldots, \hom(H_k, G)$ are the variables, M is the matrix of the system, and $b_1, \ldots, b_k$ are the constant terms. Since M is invertible (as guaranteed by our choice of $F_1, \ldots, F_k$), knowing $b_1, \ldots, b_k$ indeed enables us to compute $\hom(H_1, G), \ldots, \hom(H_k, G)$ in time $O(1)$, as required. To complete the proof, we note that $|V(G_i)| = |V(G)| \cdot |V(F_i)| = O(|V(G)|)$ and $|E(G_i)| \leq |E(G)| \cdot |E(F_i)|^2 = O(|E(G)|)$ for every $1 \leq i \leq k$, and that if G is $O(1)$-degenerate then so are $G_1, \ldots, G_k$ by Observation A.1. $\square$

ACKNOWLEDGMENTS

We would like to thank an anonymous referee for kindly pointing out that our methods can be used to study homomorphism-counting in general graphs, thus leading to Theorem 7. We are also grateful to Marc Roth and Raphael Yuster for informing us of the reduction currently used to prove Lemma 1.6 (a previous version of the paper used a different, more complicated reduction).

REFERENCES

[1] A. Abboud and V. Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. 434–443.
[2] N. Alon, R. Yuster, and U. Zwick. 1995. Color-coding. *Journal of the ACM* 42, 4 (1995), 844–856.
[3] N. Alon, R. Yuster, and U. Zwick. 1997. Finding and counting given length cycles. *Algorithmica* 17, 3 (1997), 209–223.
[4] A. L. Barabási and R. Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.
[5] C. Beeri, R. Fagin, D. Maier, A. Mendelzon, J. Ullman, and M. Yannakakis. 1981. Properties of acyclic database schemes. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*. 355–362.
[6] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. 1983. On the desirability of acyclic database schemes. *Journal of the ACM* 30, 3 (1983), 479–513.
[7] A. Benson, D. F. Gleich, and J. Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.
[8] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. 2009. Counting paths and packings in halves. In *Proceedings of the 17th Annual European Symposium on Algorithms*. 578–586.
[9] A. Björklund, P. Kaski, and Ł. Kowalik. 2017. Counting thin subgraphs via packings faster than meet-in-the-middle time. *ACM Transactions on Algorithms* 13, 4 (2017), 1–26.
[10] S. K. Bera, N. Pashanasangi, and C. Seshadhri. Linear time subgraph counting, graph degeneracy, and the chasm at size six. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference*. 38:1–38:20.
[11] S. K. Bera, N. Pashanasangi, and C. Seshadhri. Near-linear time homomorphism counting in bounded degeneracy graphs: The barrier of long induced cycles. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*. 2315–2332.
[12] J. Brault-Baron. 2016. Hypergraph acyclicity revisited. *ACM Computing Surveys* 49, 3 (2016), 1–26.
[13] M. Bressan. 2021. Faster algorithms for counting subgraphs in sparse graphs. *Algorithmica*, 83 (2021), 1–28.

[14] M. Bressan and M. Roth. 2021. Exact and approximate pattern counting in degenerate graphs: New algorithms, hardness results, and complexity dichotomies. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS'21)*. 276–285.

[15] R. Burt. 2004. Structural holes and good ideas. *American Journal of Sociology* 110, 2 (2004), 349–399.

[16] H. Chen and S. Mengel. 2016. Counting answers to existential positive queries: A complexity classification. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 315–326.

[17] N. Chiba and T. Nishizeki. 1985. Arboricity and subgraph listing algorithms. *SIAM Journal on computing* 14, 1 (1985), 210–223.

[18] F. R. K. Chung, R. L. Graham, and R. M. Wilson. 1989. Quasi-random graphs. *Combinatorica* 9, 4 (1989), 345–362.

[19] R. Curticapean, H. Dell, and D. Marx. 2017. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 210–223.

[20] R. Curticapean and D. Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. 130–139.

[21] M. Dalirrooyfard, T. D. Vuong, and V. Vassilevska Williams. 2019. Graph pattern detection: Hardness for all induced patterns and faster non-induced cycles. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 1167–1178.

[22] V. Dalmau and P. Jonsson. 2004. The complexity of counting homomorphisms seen from the other side. *Theoretical Computer Science* 329, 1–3 (2004), 315–323.

[23] F. Eisenbrand and F. Grandoni. 2004. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science* 326, 1–3 (2004), 57–67.

[24] P. Erdős, L. Lovász, and J. Spencer. 1979. Strong independence of graphcopy functions. *Graph Theory and Related Topics*. 165–172.

[25] J. Flum and M. Grohe. 2004. The parameterized complexity of counting problems. *SIAM Journal on Computing* 33, 4 (2004), 892–922.

[26] M. L. Fredman, J. Komlós, and E. Szemerédi. 1984. Storing a sparse table with $O(1)$ worst-case access time. *Journal of the ACM* 31, 3 (1984), 538–544.

[27] P. Holland and S. Leinhardt. 1970. A method for detecting structure in sociometric data. *American Journal of Sociology*, 76, 492–513.

[28] F. Hormozdiari, P. Berenbrink, N. Prulj, and S. Cenk Sahinalp. 2007. Not all scale-free networks are born equal: The role of the seed graph in PPI network evolution. *PLoS Computational Biology*, 3, 7 (2007), 118.

[29] A. Itai and M. Rodeh. 1978. Finding a minimum circuit in a graph. *SIAM Journal on Computing* 7, 4 (1978), 413–423.

[30] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao. 2011. Neighborhood based fast graph search in large networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*.

[31] T. Kloks, D. Kratsch, and H. Müller. 2000. Finding and counting small induced subgraphs efficiently. *Information Processing Letters* 74, 3 (2000), 115–121.

[32] M. Kowaluk, A. Lingas, and E. M. Lundell. 2013. Counting and detecting small subgraphs via equations. *SIAM Journal on Discrete Mathematics* 27, 2 (2013), 892–909.

[33] L. Lovász. 2012. *Large Networks and Graph Limits*. Providence: American Mathematical Society.

[34] L. Lovász and V. T. Sós. 2008. Generalized quasirandom graphs. *Journal of Combinatorial Theory, Series B* 98, 1 (2008), 146–163.

[35] D. W. Matula and L. L. Beck. 1983. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM* 30, 3 (1983), 417–427.

[36] A. McGregor, S. Vorotnikova, and H. T. Vu. 2016. Better algorithms for counting triangles in data streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 401–411.

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

[38] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. 2002. Network motifs: Simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

[39] J. Nešetřil and P. O. De Mendez. 2012. *Sparsity: Graphs, Structures, and Algorithms*. Springer Science & Business Media.

[40] J. Nešetřil and S. Poljak. 1985. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae* 26, 2 (1985), 415–419.

[41] N. Przulj. 2007. Biological network comparison using graphlet degree distribution. *Bioinformatics* 23, 2 (2007), 177–183.

[42] N. Przulj, D. G. Corneil, and I. Jurisica. 2004. Modeling interactome: Scale-free or geometric? *Bioinformatics* 20, 18 (2004), 3508–3515.

[43] S. Rudich and A. Wigderson. 2004. *Computational Complexity Theory*. American Mathematical Society.

[44] A. E. Sariyuce, C. Seshadhri, A. Pinar, and U. V. Catalyurek. 2015. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proceedings of the International World Wide Web Conference*. 927–937.

[45] C. E. Tsourakakis. 2015. The k-clique densest subgraph problem. In *Proceedings of the International World Wide Web Conference*. 1122–1132.

[46] J. Ugander, L. Backstrom, and J. M. Kleinberg. 2013. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *Proceedings of the International World Wide Web Conference*. 1307–1318.

[47] J. H. Van Lint and R. M. Wilson. 2001. *A Course in Combinatorics*. Cambridge University Press.

[48] V. Vassilevska Williams. 2009. Efficient algorithms for clique problems. *Information Processing Letters* 109, 4 (2009), 254–257.

[49] V. Vassilevska Williams. 2012. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference*. 887–898.

[50] V. Vassilevska Williams and R. Williams. 2009. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the 41st Annual ACM Symposium on the Theory of Computing*. 455–464.

Received December 2020; revised December 2021; accepted February 2022