The Complexity of Homomorphism Reconstructibility

Jan Böker
RWTH Aachen University, Germany

Louis Härtel
RWTH Aachen University, Germany

Nina Runde
RWTH Aachen University, Germany

Tim Seppelt
RWTH Aachen University, Germany

Christoph Standke
RWTH Aachen University, Germany

Abstract

Representing graphs by their homomorphism counts has led to the beautiful theory of homomorphism indistinguishability in recent years. Moreover, homomorphism counts have promising applications in database theory and machine learning, where one would like to answer queries or classify graphs solely based on the representation of a graph G as a finite vector of homomorphism counts from some fixed finite set of graphs to G. We study the computational complexity of the arguably most fundamental computational problem associated to these representations, the *homomorphism reconstructability problem*: given a finite sequence of graphs and a corresponding vector of natural numbers, decide whether there exists a graph G that realises the given vector as the homomorphism counts from the given graphs.

We show that this problem yields a natural example of an $\mathsf{NP}^{\#\mathsf{P}}$-hard problem, which still can be NP-hard when restricted to a fixed number of input graphs of bounded treewidth and a fixed input vector of natural numbers, or alternatively, when restricted to a finite input set of graphs. We further show that, when restricted to a finite input set of graphs and given an upper bound on the order of the graph G as additional input, the problem cannot be NP-hard unless $\mathsf{P} = \mathsf{NP}$. For this regime, we obtain partial positive results. We also investigate the problem's parameterised complexity and provide fpt-algorithms for the case that a single graph is given and that multiple graphs of the same order with subgraph instead of homomorphism counts are given.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Graph theory

Keywords and phrases graph homomorphism, counting complexity, parameterised complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2024.19

Related Version *Full Version*: `https://arxiv.org/abs/2310.09009` [4]

Funding *Jan Böker*: European Union (ERC, SymSim, 101054974).
Nina Runde: German Research Foundation (DFG) – 453349072.
Tim Seppelt: European Union (ERC, SymSim, 101054974), German Research Foundation (DFG) – GRK 2236/2 (UnRAVeL).
Christoph Standke: German Research Foundation (DFG) – GR 1492/16-1, GRK 2236/2 (UnRAVeL).

Acknowledgements We would like to thank Martin Grohe, Athena Riazsadri, and Jan Tönshoff for fruitful discussions.

1 Introduction

Representing a graph in terms of homomorphism counts has proven to be fruitful in theory and applications. Many graph properties studied in logic [16, 21], algebraic graph theory [15], quantum information theory [40], and convex optimisation [51, 23] can be expressed as homomorphism counts from some family of graphs. Homomorphism counts provide a basis for other counting problems [10] and have been studied extensively using diverse tools ranging from algorithmics [15] to algebra [23, 40], from combinatorics [50, 53] to category theory [14, 1]. In database theory, they correspond to evaluations of Boolean conjunctive queries under bag semantics [7, 34]. In graph learning, representations of graphs as vectors of homomorphism counts yield embeddings into a continuous latent space and underpin theoretically meaningful and successfully field-tested machine learning architectures [45, 6, 58].

In this work, we consider representations of a graph G as a finite vector of homomorphisms counts $\hom(\mathcal{I}, G) \in \mathbb{N}^{\mathcal{I}}$ for some finite set of graphs $\mathcal{I}$, which we call a *homomorphism embedding*. The rich theory of homomorphism counts calls for algorithmic applications of homomorphism embeddings: in database theory, one would ideally like to decide properties of the graph G having access to the vector $\hom(\mathcal{I}, G)$ only [22, 8]. In graph learning, certain entries of the homomorphism embedding might be associated with desirable properties of the graph being embedded, and one would like to be able to synthesise a graph having these desirable properties from a vector in the latent space [5, 24]. Despite its ubiquity in the contexts described above, homomorphism embeddings have not undergone a systematic complexity-theoretic analysis yet. The arguably most fundamental computational problem associated to them is to decide whether a vector $h \in \mathbb{N}^{\mathcal{I}}$ actually represents a graph, i.e. it is of the form $h = \hom(\mathcal{I}, G)$ for some graph G. Therefore, we consider the following *homomorphism reconstructability problem* for graph classes $\mathcal{F}$ and $\mathcal{G}$:

$\textsc{HomRec}(\mathcal{F}, \mathcal{G})$
Input Pairs $(F_1, h_1), \dots, (F_m, h_m) \in \mathcal{F} \times \mathbb{N}$ where $h_1, \dots, h_m$ are given in binary.
Question Is there a graph $G \in \mathcal{G}$ such that $\hom(F_i, G) = h_i$ for every $i \in [m]$?

We simply write $\textsc{HomRec}(\mathcal{F})$ if $\mathcal{G}$ is the class of all graphs. While the problem has a clean-cut motivation in practical applications, one quickly encounters surprising connections to deep theoretical results and long-standing open questions. One does not only have to carefully keep distance to the notorious graph reconstruction conjecture of Ulam [46], but also be aware that various decision problems involving the set $R(\mathcal{I})$ of all vectors $\hom(\mathcal{I}, G) \in \mathbb{N}^{\mathcal{I}}$ where G ranges over all graphs have received much attention recently, e.g. the homomorphism domination problem [32], whose decidability is open, the homomorphism determinacy problem [34], or the undecidable problem of determining whether inequalities of homomorphism counts hold [28, 25]. Beyond computational concerns, an abstract characterisation of the set $R(\mathcal{I})$ akin to [18, 37] is desirable, yet elusive [3].

In this paper, we establish a firm grasp on the computational complexity of the homomorphism reconstructability problem $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ by exploring from which of its aspects computational hardness arises and then finding restrictions for which efficient algorithms can be found. Despite the interest in the problem, surprisingly little progress on this question has been made. The only result related to our work is a theorem from [31], which asserts that a variant of $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ with Boolean subgraph constraints instead of homomorphism counts is $\mathsf{NP}^{\mathsf{NP}}$-complete. In particular, a formal definition of $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ has not been made before, and we would like to remark on a curious peculiarity of the definition we have chosen: a polynomial-time algorithm for $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ may exploit algebraic properties of

homomorphism numbers and deduce via arithmetic operations on the given numbers whether these can be realised by a graph G or not; let us call an algorithm of this type *arithmetic*. However, it is also conceivable that an algorithm for $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ would operate by explicitly constructing the graph G, for example, in a dynamic-programming fashion; let us call such an algorithm *constructive*. A constructive algorithm seems just as reasonable as an arithmetic one but may not be a polynomial-time algorithm for $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ since the order of G does not have to be polynomial in the length of the binary encoding of the given homomorphism numbers, even if one of the constraint graphs is $F_i = \bullet$. Hence, it also seems reasonable to define the *bounded homomorphism reconstructability problem* $\textsc{BHomRec}(\mathcal{F}, \mathcal{G})$ where an additional input $n \in \mathbb{N}$ given in unary imposes a bound on the number of vertices in G that is linear in the input encoding. This bound, however, poses an additional constraint to the graph G, which may make the design of arithmetic algorithms more difficult or impossible. Hence, both $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ and $\textsc{BHomRec}(\mathcal{F}, \mathcal{G})$ are arguably reasonable definitions of the reconstructability problem, each in their own right, and we consider both.

The Ocean of Hardness

Let $\mathsf{C_{=}P}$ denote the class of all decision problems L for which there is a function $f \in \#\mathsf{P}$ and a polynomial-time computable function g such that, for every instance x of L, we have $x \in L \iff f(x) = g(x)$ [54, 57, 26]. We first show that both the unbounded and the bounded reconstructability problem are $\mathsf{NP}^{\mathsf{C_{=}P}}$-hard when not restricted in any way. Note that $\mathsf{NP}^{\mathsf{C_{=}P}} = \mathsf{NP}^{\#\mathsf{P}}$ since, whenever we issue a call to the $\#\mathsf{P}$-oracle, we may guess the output using nondeterminism and verify it using a $\mathsf{C_{=}P}$-oracle instead [56, 9]; we refer to the full version [4] for more details about counting classes.

▶ **Theorem 1.** *Let $\mathcal{F}$ denote the class of all graphs. Then, $\textsc{HomRec}(\mathcal{F})$ is in NEXP and $\textsc{BHomRec}(\mathcal{F})$ is in $\mathsf{NP}^{\mathsf{C_{=}P}}$. Moreover, both problems are $\mathsf{NP}^{\mathsf{C_{=}P}}$-hard. Hence, they are not contained in the polynomial hierarchy unless it collapses.*

This result illustrates two intertwined sources of hardness in the reconstructability problem: first, the *reconstruction hardness* of finding a graph G, which corresponds to the class NP, and secondly, the *counting hardness* of verifying that G actually satisfies the given constraints, which corresponds to the $\mathsf{C_{=}P}$-oracle. The reconstruction hardness is what we are interested in since the counting hardness simply reflects the hardness of counting homomorphisms, which is well understood: the problem $\#\textsc{Hom}$ of counting the number of homomorphisms from a graph F to a graph G is $\#\mathsf{P}$-complete and, under the complexity-theoretic assumption that $\mathsf{FPT} \neq \#\mathsf{W}[1]$, becomes tractable if and only if one restricts the graphs F to come from a (recursively enumerable) class of bounded treewidth [12].

To isolate the reconstruction hardness, we restrict $\mathcal{F}$ to be a class of bounded treewidth, which allows us to verify in polynomial time that a graph G satisfies all given constraints. In particular, the bounded problem is then in NP since one can guess a graph G of the given size and verify that it satisfies all constraints in polynomial time. We show that the intuition conveyed by Theorem 1 is correct: $\textsc{HomRec}(\mathcal{F})$ is still NP-hard if $\mathcal{F}$ has bounded treewidth and even remains NP-hard when only a constant number of constraints is allowed to appear in the input.

▶ **Theorem 2.** *There is a class $\mathcal{F}$ of graphs of bounded treewidth such that $\textsc{HomRec}(\mathcal{F})$ and $\textsc{BHomRec}(\mathcal{F})$ are NP-hard.*

The reduction used to prove Theorem 2 further demonstrates that both problems remain NP-hard when only allowing some fixed number of constraints: it produces a family of instances with graphs from $\mathcal{F}$ where the number of constraints, all homomorphism numbers,

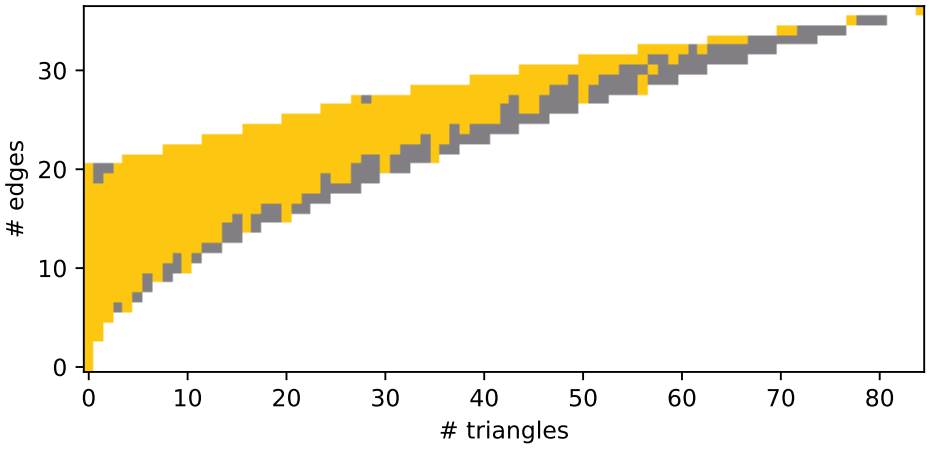


Figure 1 Reconstructable △ and ⊶ subgraph counts (yellow) for graphs on nine vertices. The grey area depicts the values which are not ruled out by the Kruskal–Katona bound [33, 30], cf. [36, 13.31b], or the Razborov bound [49], cf. [38, Theorem 16.14]. The △-counts realisable by graphs on nine vertices correspond to columns with at least one yellow box.

and all but one constraint graph are fixed. This raises the question what happens if we fix all input graphs and allow the homomorphism numbers to vary instead, i.e. does $\textsc{HomRec}(\mathcal{F})$ become tractable if $\mathcal{F}$ is *finite*? Even though the input to $\textsc{HomRec}(\mathcal{F})$ for finite $\mathcal{F}$ essentially consists only of natural numbers encoded in binary, we are still able to show that $\textsc{HomRec}(\mathcal{F})$ is also NP-hard in this case. In contrast, however, $\textsc{BHomRec}(\mathcal{F})$ is *sparse* for finite $\mathcal{F}$, i.e. it only has polynomial number of yes-instances, and hence, unlikely to be NP-hard [39].

▶ **Theorem 3.** *There is a finite set $\mathcal{F}$ of graphs such that $\textsc{HomRec}(\mathcal{F})$ is NP-hard. If $\textsc{BHomRec}(\mathcal{F})$ is NP-complete for a finite set $\mathcal{F}$ of graphs, then $\mathsf{P} = \mathsf{NP}$.*

Hence, the complexity of the reconstructability problem becomes much more nuanced for finite $\mathcal{F}$, and in order to design efficient algorithms for it, we seemingly have to focus on $\textsc{BHomRec}(\mathcal{F})$ for finite $\mathcal{F}$. The fact that there are only polynomially many feasible combinations of homomorphism numbers in this case might be somehow exploitable, e.g. by a dynamic-programming algorithm operating on a table indexed by them.

Islands of Tractability

The first tractable instance of $\textsc{HomRec}(\mathcal{F})$ that comes to mind is given by $F_1 = \bullet$ and $F_2 = \bullet\!\!-\!\!\bullet$ and $h_1, h_2 \in \mathbb{N}$. In this case, we need to decide whether $h_2 \leq h_1(h_1 - 1)$ and h_2 is even. Although this is fairly trivial, we encounter severe combinatorial difficulties when attempting to generalise this even to $F_1 = \bullet$ and $F_2 = \triangle$. Figure 1 shows that the set of reconstructible vectors has a non-trivial shape. In particular, while highly engineered results from extremal combinatorics [38, 49, 27, 20] provide insights in the (asymptotic) behaviour of the upper and lower boundary of that set, we are unable to characterise the seemingly erratic gaps and spikes depicted in Figure 1. On the positive side, we are able to map out large regions of reconstructible vectors using number-theoretic insights:

▶ **Theorem 4.** *There exists a function $\gamma\colon \mathbb{N} \to \mathbb{N}$ such that for every $k \geq 2$, $n \geq 1$, $h \leq \binom{n}{k}$, there exists a graph G on $n + \gamma(k-1) - 1$ vertices such that $\mathrm{sub}(K_k, G) = h$.*

Theorem 4 allows for the construction of graphs with almost all possible numbers of clique subgraphs. Indeed, the proportion of covered values is $\binom{n}{k} \Big/ \binom{n+\gamma(k-1)-1}{k} = 1 - o(1)$ for $n \to \infty$ and fixed k. In other words, all sensible values can be realised by only slightly deviating from the stipulated size constraint.

The problem arising when dealing with the remaining admissible parameters, i.e. $\binom{n}{k} < h \leq \binom{n+\gamma(k-1)-1}{k}$, seems to be that the constraints on the number of vertices and cliques interact in an elusive fashion. Although understanding such interactions better remains a direction for future investigations, we are able to identify certain combinatorial conditions under which the constraints are somewhat independent. This yields fixed-parameter algorithms for variants of $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ contrasting Theorem 3. Here, Proviso 20 stipulates mild constraints on the graph classes $\mathcal{F}$ and $\mathcal{G}$. For example, $\mathcal{G}$ can be taken to be the class of all graphs and $\mathcal{F}$ to be the class of all connected graphs.

▶ **Theorem 5.** *For graph classes $\mathcal{F}$, $\mathcal{G}$ as in Proviso 20, the following problem is in* FPT*:*

p-$\textsc{SingleHomRec}(\mathcal{F}, \mathcal{G})$
Input a graph $F \in \mathcal{F}$, an integer $h \in \mathbb{N}$ given in binary
Parameter $\lvert V(F)\rvert$
Question Does there exist a graph $G \in \mathcal{G}$ such that $\hom(F, G) = h$?

Curiously, any fpt-algorithm for p-$\textsc{SingleHomRec}(\mathcal{F}, \mathcal{G})$ has to be arithmetic: In FPT, one can neither construct the graph G nor count homomorphisms from F to G [12]. Indeed, our algorithm essentially only operates with integers and exploits number-theoretic properties of the set of reconstructible numbers. In Theorem 21, we apply similar ideas to derive an fpt-algorithm for a version of $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ with multiple equi-sized subgraph constraints.

2 Preliminaries and Conventions

Write $\mathbb{N} = \{0, 1, 2, \dots\}$ for the set of natural numbers. $A \leq_p B$ denotes that a decision problem A is polynomial-time many-one reducible to the decision problem B. A *graph* is a pair $G = (V, E)$ of a set of *vertices* V and a set of *edges* $E \subseteq \binom{V}{2}$. We usually write $V(G)$ and $E(G)$ for V and E, respectively, and use n to denote the *order* $n := |V(G)|$ of G. For ease of notation, we denote an edge $\{u, v\}$ by uv or vu. A *homomorphism* from a graph F to a graph G is a mapping $h\colon V(F) \to V(G)$ such that $h(uv) \in E(G)$ for every $uv \in E(F)$. A *($\mathcal{C}$-vertex-)coloured graph* is a triple $G = (V, E, c)$ where (V, E) is a graph, the *underlying graph*, and $c\colon V(G) \to \mathcal{C}$ a function assigning a *colour* from a set $\mathcal{C}$ to every vertex of G. An *($\mathcal{L}$-)labelled* graph is defined analogously with a function $\ell\colon \mathcal{L} \to V(G)$ assigning a vertex of G to every *label* from a set of labels $\mathcal{L}$ instead. Homomorphisms between coloured graphs and between labelled graphs are then defined as homomorphisms of the underlying graphs that respect colours and labels, respectively.

A graph G' is a *subgraph* of a graph G, written $G' \subseteq G$, if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The *subgraph induced by a set* $U \subseteq V(G)$, written $G[U]$, is the subgraph of G with vertices U and edges $E(G) \cap \binom{U}{2}$. We write $\hom(F, G)$ for the number of homomorphisms from F to G, $\mathrm{sub}(F, G)$ for the number of subgraphs $G' \subseteq G$ such that $G' \cong F$, and $\mathrm{indsub}(F, G)$ for the number of subsets $U \subseteq V(G)$ such that $G[U] \cong F$. This notation generalises to coloured and labelled graphs in the straightforward way.

The definition of $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ from the introduction directly generalises to classes $\mathcal{F}$ and $\mathcal{G}$ of relational structures over the same signature, and in particular, classes of labelled and coloured graphs. We call a pair (F, h) of a structure F and a number $h \in \mathbb{N}$ a *constraint*

and also denote the pair by $\hom(F) = h$, or for example in the context of reconstructability of subgraph counts, by $\mathrm{sub}(F) = h$. As stipulated in the introduction, if $\mathcal{F}$ is a class of graphs, we abbreviate $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ for the class of all graphs $\mathcal{G}$ to $\textsc{HomRec}(\mathcal{F})$. We use the abbreviation $\textsc{BHomRec}(\mathcal{F})$ in the same way, and for a class of labelled or coloured graphs $\mathcal{F}$, we analogously abbreviate the problem name if $\mathcal{G}$ is the class of all labelled or all coloured graphs, respectively. We define the problems $\textsc{SubRec}(\mathcal{F}, \mathcal{G})$ and $\textsc{BSubRec}(\mathcal{F}, \mathcal{G})$ analogously to $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ and $\textsc{BHomRec}(\mathcal{F}, \mathcal{G})$, respectively, with subgraph counts instead of homomorphism counts and also follow the conventions agreed upon above. See the full version [4] for details.

3 Decidability

The problem $\textsc{BHomRec}(\mathcal{F}, \mathcal{G})$ is trivially decidable if membership in $\mathcal{G}$ is decidable since it is possible to perform a brute-force search for G by testing all graphs up to order n. For $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$, decidability is implied by the following lemma:

▶ **Lemma 6.** *Let $\mathcal{F}$ and $\mathcal{G}$ be classes of structures over the same signature. Suppose that $\mathcal{G}$ is closed under taking induced substructures. Let $(F_1, h_1), \dots, (F_m, h_m) \in \mathcal{F} \times \mathbb{N}$. Let $G \in \mathcal{G}$ be such that $\hom(F_i, G) = h_i$ for all $i \in [m]$. Then there exists $H \in \mathcal{G}$ such that $|H| \leq \sum_{i=1}^{m} h_i |F_i|$ and $\hom(F_i, H) = h_i$ for all $i \in [m]$.*

Proof. Let U denote the union over all images of homomorphisms $F_i \to G$, $i \in [m]$. Clearly, $|U| \leq \sum_{i=1}^{m} \hom(F_i, G) |F_i|$. Let $H := G[U] \in \mathcal{G}$. For all $i \in [m]$, $\hom(F_i, H) = \hom(F_i, G)$. Indeed, every homomorphism $F_i \to H$ gives rise to a homomorphism $F_i \to G$ by composition with the embedding $H \hookrightarrow G$. Conversely, observe that every homomorphism $F_i \to G$ is in fact a homomorphism $F_i \to H$ since its image is contained in U. It remains to observe that this correspondence establishes a bijection. ◀

Under the assumptions of the previous lemma, $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ can be decided via a brute-force search on all graphs in $\mathcal{G}$ up to order $\sum_{i=1}^{m} h_i |F_i|$. Together with our insights on $\textsc{BHomRec}(\mathcal{F}, \mathcal{G})$, this shows the following theorem.

▶ **Theorem 7.** *Let $\mathcal{F}$ and $\mathcal{G}$ be classes of structures over the same signature. Suppose that membership in $\mathcal{G}$ is decidable. Then $\textsc{BHomRec}(\mathcal{F}, \mathcal{G})$ is decidable. If $\mathcal{G}$ is closed under taking induced substructures, then also $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ is decidable.*

4 Hardness

In this section, we prove the hardness results presented in the introduction. First, we remark that, for the class $\mathcal{F}$ of all graphs, $\textsc{HomRec}(\mathcal{F})$ is in NEXP since we can non-deterministically guess a graph G of exponential size by Lemma 6 and then count homomorphisms to G in exponential time by simply going through all mappings from the given constraint graphs to G. For the bounded problem, we are given a size bound on G as part of the input, which means that $\textsc{BHomRec}(\mathcal{F}, \mathcal{F})$ is in $\mathsf{NP}^{\mathsf{C}_=\mathsf{P}}$ since we can non-deterministically guess a graph G of linear size and then verify that G satisfies all constraints by using the $\mathsf{C}_=\mathsf{P}$-oracle.

▶ **Theorem 8.** *Let $\mathcal{F}$ denote the class of all graphs. Then, $\textsc{HomRec}(\mathcal{F})$ is in NEXP and $\textsc{BHomRec}(\mathcal{F})$ is in $\mathsf{NP}^{\mathsf{C}_=\mathsf{P}}$.*

Let $\mathcal{F}$ denote the class of all graphs. The fact that $\textsc{BHomRec}(\mathcal{F}) \in \mathsf{NP}^{\mathsf{C}_=\mathsf{P}}$ reflects the intuition on the hardness on $\textsc{BHomRec}(\mathcal{F})$ that we gave in the introduction, i.e. that there are two intertwined sources of hardness, the *reconstruction hardness*, manifested as NP,

and the *counting hardness*, manifested as the $\mathsf{C}_{=}\mathsf{P}$-oracle. In Section 4.1, we show that this intuition is in fact correct and that HomRec($\mathcal{F}$) and BHomRec($\mathcal{F}$) are $\mathsf{NP}^{\mathsf{C}_{=}\mathsf{P}}$-hard. In Section 4.2, we further reinforce this intuition by presenting a reduction from the well-known NP-complete problem SetSplitting to HomRec($\mathcal{F}$) and BHomRec($\mathcal{F}$) for a class of bounded treewidth $\mathcal{F}$ that proves that these problems are NP-hard for a family of inputs where the number of constraints, all homomorphism numbers, and all graphs but one are fixed. This isolates the reconstruction hardness and supports our intuition of the reconstruction hardness being NP-hardness.

In our reduction from SetSplitting, the given instance to SetSplitting is encoded as a constraint graph that grows with the size of the given instance while the number of produced constraints and the produced homomorphism numbers remain fixed. This raises the question if we can achieve tractability by restricting the order of the constraint graphs instead, i.e. if HomRec($\mathcal{F}$) and BHomRec($\mathcal{F}$) become tractable if $\mathcal{F}$ is *finite*. In Section 4.3, we show that there is a finite class $\mathcal{F}$ for which HomRec($\mathcal{F}$) is NP-hard by reducing from the NP-complete problem QPoly of solving an equation involving a quadratic polynomial. We further show that this reduction cannot be adapted to work for BHomRec($\mathcal{F}, \mathcal{G}$) and then prove that BHomRec($\mathcal{F}, \mathcal{G}$) is sparse, i.e. it only has polynomial number of yes-instances, which means that it cannot be NP-hard under the assumption that $\mathsf{P} \neq \mathsf{NP}$. In Section 4.4, we briefly discuss which of our hardness results also hold for the subgraph reconstructability problem.

For the sake of presentability, we only provide the reductions to the reconstructability problem for labelled or coloured graphs in the main body of this paper. In the full version [4], we show that all these reductions be adapted to (unlabelled and uncoloured) graphs via gadget constructions that employ *Kneser graphs*, cf. the full version [4].

4.1 $\mathsf{NP}^{\mathsf{C}_{=}\mathsf{P}}$-Hardness

We first show that both problems HomRec($\mathcal{F}$) and BHomRec($\mathcal{F}$) are $\mathsf{NP}^{\mathsf{C}_{=}\mathsf{P}}$-hard. We reduce from the following $\mathsf{NP}^{\mathsf{C}_{=}\mathsf{P}}$-complete variant of 3-Colouring, cf. the full version [4].

EC-3-Colouring
Input A graph G, a subset of vertices $S \subseteq V(G)$, and a $k \in \mathbb{N}$ given in binary.
Question Is there a homomorphism $c\colon G[S] \to$  such that there are exactly k homomorphisms $\hat{c}\colon G \to$ such that $\hat{c}|_S = c$?

The idea is that the number of homomorphisms from a graph F to the complete graph on three vertices  is precisely the number of 3-colourings of F. Then, one can formulate constraints that can only be satisfied by  , and by adding an additional constraint $\hom(F) = h$, one obtains a yes-instance to the reconstructability problem if and only if the number of 3-colourings of F is exactly h. By additionally employing labels on F and  , we obtain a reduction from EC-3-Colouring.

▶ **Theorem 9.** *Let $\mathcal{LF}$ denote the class of all labelled graphs. Then,* EC-3-Colouring $\leq_p$ HomRec($\mathcal{LF}$) *and* EC-3-Colouring $\leq_p$ BHomRec($\mathcal{LF}$).

Proof. Given an instance (F, S, k) of EC-3-Colouring, let $m := |S|$. Fix an arbitrary linear order on S, i.e. $S = \{s_1, \ldots, s_m\}$. We use the labels $\ell_1, \ldots, \ell_m$ to classify vertices of F as members of S: let F' be the labelled graph obtained from F and S by assigning label ℓ_i to the vertex $s_i \in S$ for every $i \in [m]$. The reduction then produces the following constraints, where for BHomRec($\mathcal{LF}$), we set the additional size constraint to three:

(a) $\hom(F') = k$,
(b) $\hom(\bullet) = 3$,
(c) $\hom(\bullet\!\!-\!\!\bullet) = 6$, and
(d) $\hom(\bullet\ \ell_i) = 1$ for every $i \in [m]$.
Note that  is the unique graph that satisfies $\hom(\bullet) = 3$ and $\hom(\bullet\!\!-\!\!\bullet) = 6$. The remaining constraints enforce that each label from $\{\ell_1, \dots, \ell_m\}$ appears exactly once in G. For a partition $\mathcal{L} = \{L_1, L_2, L_3\}$ of these labels into at most three parts, let $G_{\mathcal{L}}$ denote the with its three vertices labelled by the labels in L_1, L_2, and L_3, respectively. Then, (F, S, k) is in EC-3-Colouring if and only if there is a partition $\mathcal{L}$ as above such that $\hom(F', G_{\mathcal{L}}) = k$, which again is the case if and only if there is a labelled graph G that satisfies the constraints produced by the reduction. ◀

By encoding vertex labels by gadgets consisting of Kneser graphs, we can also obtain a reduction for uncoloured graphs. Since the number of labels used in the reduction above depends on the input instance, we can view every label as indexed by a binary number and construct a gadget from distinct Kneser graphs for 0 and 1 to encode its index. This allows us to only use a constant and finite set of Kneser graphs, which means that we do not have to worry about their size, and guarantees that the resulting reduction still runs in polynomial time. The proof can be found in the full version [4].

▶ **Theorem 10.** *Let $\mathcal{F}$ denote the class of all graphs. Then,* EC-3-Colouring $\leq_p$ HomRec($\mathcal{F}$) *and* EC-3-Colouring $\leq_p$ BHomRec($\mathcal{F}$).

Together with the following observation, this then finishes the proof of Theorem 1.

▶ **Corollary 11.** *Let $\mathcal{F}$ denote the class of all graphs. Then,* HomRec($\mathcal{F}$) *and* BHomRec($\mathcal{F}$) *are not in* PH *unless* PH *collapses.*

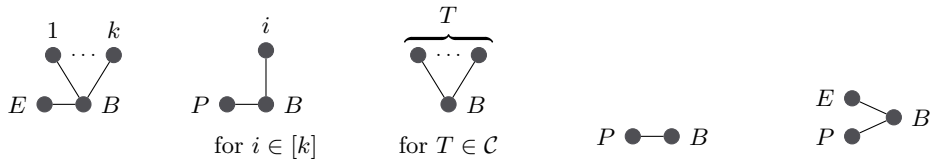
Proof. This follows from Toda's Theorem [55] since every oracle query in the computation of a $\mathsf{P}^{\#\mathsf{P}}$-machine can be simulated by a nondeterministic polynomial-time machine guessing the answer and then verifying it with an oracle query to a problem in $\mathsf{C}_{=}\mathsf{P}$. ◀

4.2 NP-Hardness for Constraints of Bounded Treewidth

The reduction used to prove $\mathsf{NP}^{\mathsf{C}_{=}\mathsf{P}}$-hardness in the previous section uses the graph given as input for EC-3-Colouring as a constraint, which has the side effect that the treewidth of the produced instances is not bounded. Moreover, the same reduction cannot be easily adapted to prove NP-hardness of HomRec($\mathcal{F}$) for a class of graphs $\mathcal{F}$ of bounded treewidth by simply considering input graphs of bounded treewidth: the NP-complete problem 3-Colouring [19] restricted to graphs of bounded treewidth is polynomial-time solvable [2]. Hence, we need a different approach to such a reduction. We reduce from the following problem SetSplitting, which is well-known to be NP-complete [35].

SetSplitting
Input A collection $\mathcal{C}$ of subsets of a finite set S.
Question Is there a partition of S into two subsets S_1 and S_2 such that no subset in $\mathcal{C}$ is entirely contained in either S_1 or S_2?

The idea is that we represent every element i of S by a vertex of colour i. A set $T \in \mathcal{C}$ is encoded by a star that has a leaf for every element of T and a root vertex of some fixed colour that is distinct from the colours used for elements of S. Intuitively, the graph G



(a) $\hom(F_1) = 1$. **(b)** $\hom(F_2) = 2$. **(c)** $\hom(F_3) = 0$.

Figure 2 The three constraints produced by the reduction of Theorem 12.

then consists of two stars that encode the two sets S_1 and S_2. To ensure that no subset in $\mathcal{C}$ is entirely contained in S_1 or S_2, we use the constraints to require that there are no homomorphisms from our constraint graphs to G.

The idea sketched above produces a single constraint for every set $T \in \mathcal{C}$. We can use the following trick to combine these constraints into a single one: a graph F consisting of several connected components has exactly one homomorphism to G if and only if all its connected components have exactly one homomorphism to G. Hence, if we choose G to consist of two stars that encode S_1 and S_2 and also add a star that has all elements of S as its leaves, we can instead require to have exactly one homomorphism from all our constraint graphs to G and, thus, combine all these constraint graphs into a single (disconnected) graph from which we require to have exactly one homomorphism.

▶ **Theorem 12.** *Let $\mathcal{CS}$ denote the class of all disjoint unions of coloured stars. Then,* SetSplitting $\leq_p$ HomRec($\mathcal{CS}$) *and* SetSplitting $\leq_p$ BHomRec($\mathcal{CS}$)*, where the reduction only produces three constraints* $\hom(F_1) = 1$*,* $\hom(F_2) = 2$*, and* $\hom(F_3) = 0$*.*

Proof. Given a collection $\mathcal{C}$ of subsets of a finite set S, we may assume that $S = [k]$ by re-labelling the elements of S. In the construction of the graphs F_1, F_2, F_3, we use the colours $1, \ldots, k$ and also B ("black"), E ("everything"), and P ("partition"). We construct these graphs as shown in Figure 2 and add the constraints $\hom(F_1) = 1$, $\hom(F_2) = 2$, $\hom(F_3) = 0$. Note that the constraint $\hom(F_1) = 1$ is equivalent to $\hom(F) = 1$ for every connected component F of F_1. For BHomRec($\mathcal{CS}$), we set the size bound to $2k + 6$.

Given a partition of S into sets S_1 and S_2 such that no subset in $\mathcal{C}$ is entirely contained in either S_1 or S_2, the graph G_{S_1,S_2} in Figure 3 satisfies all constraints. Conversely, let G be a coloured graph that satisfies all constraints. By the constraint $\hom(F_2) = 2$, the graph F_2 occurs exactly twice as a subgraph in G; call these occurrences G_1 and G_2. Let $S_1 \subseteq [k]$ be the set of all $i \in [k]$ such that a vertex of colour i is connected to the B-vertex of G_1. If the B-vertex in $G_2 \subseteq G$ is distinct from the B-vertex in $G_1 \subseteq G$, then let $S_2 \subseteq [k]$ be the set of all $i \in [k]$ such that a vertex of colour i is connected to the B-vertex of G_2; otherwise, let $S_2 \coloneqq \emptyset$. The first constraint yields that the i-B-P-graph occurs exactly once as a subgraph of G for every $i \in [k]$. By the second constraint, every i-B-P graph has to be a supergraph of one of the two occurrences G_1 and G_2 of F_2. Hence, the sets S_1 and S_2 cover S. Moreover, as every i-B-P graph occurs exactly once as a subgraph of G, the sets S_1 and S_2 have to be a partition of S. Finally, by the first constraint, the 1-k-B-E-graph is a subgraph of G. Observe that, for every $T \in \mathcal{C}$, the set T cannot be a subset of either S_1 or S_2 as the T-B-graph occurs exactly once in G by the first constraint and it already is a subgraph of the 1-k-B-E-graph whose B-vertex has to be distinct from the B-vertices of the occurrences of F_2 by the third constraint. ◀

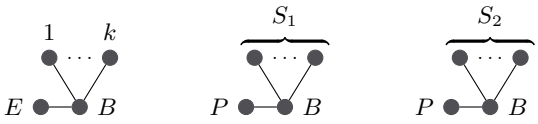


Figure 3 The graph G_{S_1,S_2} constructed from S_1 and S_2 in the proof of Theorem 12.

Note the following curiosity of Theorem 12: the numbers in the constraints produced by the reduction are constant, i.e. they do not depend on the specific problem instance given as an input. Hence, the hardness solely lies in the graphs and not the homomorphism numbers; more specifically, there is only a single constraint graph that depends on the input instance and is the cause of hardness.

We again use Kneser graphs to turn this reduction into one for uncoloured graphs. We view the colours as numbers and use gadgets of Kneser graphs that encode these numbers in binary. This allows us to argue that the reduction is still correct, where in particular, we have to argue that, if there is a graph G satisfying all constraints, then we can extract a solution to the given SetSplitting instance from it; this is not straightforward since such a graph G does not have to adhere to our encoding of coloured graphs. This then yields the following theorem, which implies Theorem 2. The proof can be found in the full version [4].

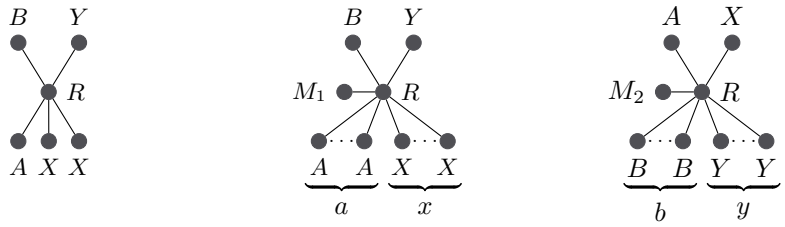
▶ **Theorem 13.** *There is a class of graphs $\mathcal{F}$ of bounded treewidth such that* SetSplitting $\leq_p$ HomRec($\mathcal{F}$) *and* SetSplitting $\leq_p$ BHomRec($\mathcal{F}$)*, where the number of constraints, the homomorphism numbers, and all constraint graphs but one are constant.*

4.3 NP-Hardness for a Finite Set of Graphs

The previous sections shows that restricting the constraint graphs to be from a class $\mathcal{F}$ of bounded treewidth and the number of constraints to a constant is not enough to achieve tractability: both HomRec($\mathcal{F}$) and BHomRec($\mathcal{F}$) remain NP-hard. What happens if we go even further and consider a finite class $\mathcal{F}$? Then, the treewidth of $\mathcal{F}$ is trivially bounded and so is the number of constraints. At first glance, hardness in this case seems unlikely since the reductions presented in the previous sections rely heavily on encoding the input instance as the constraint graphs and only used small constants for the homomorphism numbers. Now, the input essentially consists just of homomorphism numbers encoded in binary and, for BHomRec($\mathcal{F}$), also the size of the desired graph encoded in unary. This makes it all the more surprising that we can actually prove the NP-hardness of HomRec($\mathcal{F}$). We reduce from the following decision problem, which only takes three natural numbers in binary encoding as input and is NP-complete [41, Theorem 1]:

QPoly
Input Natural numbers a, b, and c in binary encoding.
Question Are there natural numbers x and y such that $ax^2 + by = c$?

The idea of the reduction is simple: we encode the polynomial $ax^2 + by$ as a coloured star F from which we require exactly c homomorphisms. This star has a leaf of colour A and two leaves of colour X to encode the monomial ax^2. Furthermore, it has a leaf of colour B and of colour Y to encode the monomial by. Then, the sum $ax^2 + by$ is realised by encoding x and y as two separate components of G – additional constraints are used to enforce that G has precisely two components, that the first component has exactly a leaves of colour A, and that the second component has exactly b leaves of colour B.



(a) The graph F_{poly}. **(b)** The graph $G_{a,b,x,y}$ constructed from $a, b, x, y \in \mathbb{N}$.

Figure 4 The most important graphs used in the reduction of Theorem 14.

▶ **Theorem 14.** *There is a finite set $\mathcal{F}$ of coloured stars such that* QPoly $\leq_p$ HomRec($\mathcal{F}$).

Proof. We use the colours R, A, X, B, Y, M_1 and M_2. The main observation is that, for the graphs F_{poly} and $G_{a,b,x,y}$ from Figure 4, we have $\hom(F_{\mathsf{poly}}, G_{a,b,x,y}) = ax^2 + by$. Note that, however, the size of $G_{a,b,x,y}$ is not polynomial in $\log a + \log b + \log c$, i.e. in the size of an instance (a, b, c) of QPoly. Given an instance (a, b, c) of QPoly, we produce the following constraints and denote the set of all coloured graphs used in these constraints by $\mathcal{F}$; note that $\mathcal{F}$ is independent of the instance (a, b, c):

(a) $\hom(A \bullet\!\!-\!\!\bullet R) = a + 1$,
(b) $\hom(B \bullet\!\!-\!\!\bullet R) = b + 1$,
(c) $\hom(F_{\mathsf{poly}}) = c$,
(d) $\hom(\bullet R) = 2$,
(e) $\hom(M_1 \bullet\!\!-\!\!\bullet R) = 1$,
(f) $\hom(M_2 \bullet\!\!-\!\!\bullet R) = 1$,
(g) $\hom\left(\begin{smallmatrix} M_1 \\ M_2 \end{smallmatrix} \!\!-\!\!\bullet R\right) = 0$,
(h) $\hom\left(\begin{smallmatrix} B & & Y \\ & M_1 \; R & \end{smallmatrix}\right) = 1$,
(i) $\hom\left(\begin{smallmatrix} A & & X \\ & M_2 \; R & \end{smallmatrix}\right) = 1$.

If (a, b, c) is an instance of QPoly with $x, y \in \mathbb{N}$ such that $ax^2 + by = c$, then $G_{a,b,x,y}$ from Figure 4b satisfies all these constraints. Conversely, if there is a coloured graph G that satisfies all constraints, we know by (d)–(g) that there are two R-coloured vertices v_1 and v_2 such that v_1 is connected to an M_1-coloured vertex but not an M_2-coloured vertex and v_2 is connected to an M_2-coloured vertex but not an M_1-coloured vertex. By (h) and (i), v_1 has exactly one B-coloured neighbor and exactly one Y-coloured neighbor and v_2 has exactly one A-coloured neighbor and exactly one X-coloured neighbor. Hence, by (a) and (b), v_1 has exactly a neighbors of colour A and v_2 has exactly b neighbors of colour B. Let x be the number of X-coloured neighbors of v_1 and y be the number of Y-coloured neighbors of v_2. Then, we have $c = \hom(F_{\mathsf{poly}}, G) = ax^2 + by$. ◀

We can again use Kneser graphs to obtain a reduction for uncoloured graphs, which implies the hardness stated in Theorem 3. The proof can be found in the full version [4].

▶ **Theorem 15.** *There is a finite set $\mathcal{F}$ of graphs such that* QPoly $\leq_p$ HomRec($\mathcal{F}$).

Can such a reduction also be used to prove NP-hardness of BHomRec($\mathcal{F}$) for a finite class $\mathcal{F}$? This is not possible, since for a fixed graph F, the number of homomorphisms from F to a graph G is polynomial in the order of G. Hence, with a finite set $\mathcal{F}$ of graphs and a

graph G of order up to n, we can only realise polynomially many distinct homomorphism numbers, while the hardness of solving the equation $ax^2 + by = c$ stems from the fact that there is an exponential number of solution candidates. This implies that BHomRec($\mathcal{F}$) is sparse for every finite $\mathcal{F}$, which means that it cannot be NP-hard unless P = NP [39].

▶ **Theorem 16.** *If* BHomRec($\mathcal{F}$) *is* NP*-hard for a finite set of graphs* $\mathcal{F}$*, then* P = NP*.*

4.4 Subgraph Counts

While subgraphs counts can be expressed as linear combinations of homomorphism numbers via injective homomorphism numbers, cf. [10], adapting our reductions to subgraph counts is not as straightforward. There is no obvious way of adapting the reduction of Theorem 9 since non-injectivity is crucial to encode colourability of graphs. The reduction of Theorem 12 can partially be salvaged by producing individual constraints instead of only three constraints. Then, all constraint graphs are *colourful*, which means that their subgraph counts are homomorphism counts, which can be computed efficiently. However, the gadget construction used in Theorem 13 then produces constraint graphs of unbounded vertex-cover number. Hence, this reduction is uninteresting since the determining factor for tractability of subgraphs counts is the vertex-cover number [11].

The results for finite $\mathcal{F}$ transfer to subgraphs and are meaningful since subgraph counts can trivially be computed in polynomial time in this case. However, for subgraph counts, the reduction of Theorem 3 encodes the binomial equation $a\binom{x}{2} + by = c$ instead of $ax^2 + by = c$. Luckily, the problem BPoly of solving this equation is still NP-complete, cf. the full version [4]. Moreover, BSubRec($\mathcal{F}$) is still sparse for every finite class of graphs $\mathcal{F}$.

▶ **Theorem 17.** *There is a finite set* $\mathcal{F}$ *of graphs such that* BPoly $\leq_p$ SubRec($\mathcal{F}$)*. If* BSubRec($\mathcal{F}$) *is* NP*-hard for a finite set of graphs* $\mathcal{F}$*, then* P = NP*.*

5 Towards Tractability: Reconstructing Clique Counts

In this section, we show that BHomRec($\mathcal{F}$) is tractable when the only constraint graph is a clique and the constraints are in a certain range. The proofs rely on number-theoretic insights and tailored combinatorial constructions. Using a number-theoretic result by Kamke [29], we show that for almost all sensible $n, h, k \in \mathbb{N}$ there exists a graph G on slightly more than n vertices containing h copies of the k-vertex clique K_k as a subgraph.

▶ **Theorem 18** (Kamke [29], cf. [42, Theorem 11.10])**.** *There exists a function* $\gamma\colon \mathbb{N} \to \mathbb{N}$ *such that for every* $k \geq 1$ *and* $n \geq 1$ *there exist* $a_1, \ldots, a_{\gamma(k)} \in \mathbb{N}$ *such that* $n = \sum_{i=1}^{\gamma(k)} \binom{a_i}{k}$*.*

Nečaev [43] showed that $\gamma(k)$ can be chosen to be of order at most $O(k \log k)$ and gave a similar lower bound in [44]. Specific values of γ include $\gamma(1) = 1$, Gauß' Eureka Theorem, cf. [47], stating $\gamma(2) = 3$ and the unproven Tetrahedral Numbers Conjecture of Pollock [48] asserting $\gamma(3) = 5$.

▶ **Theorem 4.** *There exists a function* $\gamma\colon \mathbb{N} \to \mathbb{N}$ *such that for every* $k \geq 2$*,* $n \geq 1$*,* $h \leq \binom{n}{k}$*, there exists a graph* G *on* $n + \gamma(k-1) - 1$ *vertices such that* $\mathrm{sub}(K_k, G) = h$*.*

Proof. Let γ denote the function from Theorem 18. For every fixed k, the proof is by induction on n. For $n \leq k$, the claim is trivial. Suppose subsequently that $n > k$. Inductively, we may suppose that there exists a graph G on $n - 1 + \gamma(k-1) - 1$ vertices with $h \leq \binom{n-1}{k}$ copies of K_k. One may add an isolated vertex to obtain a graph on $n + \gamma(k-1) - 1$ vertices with h copies of K_k, as desired.

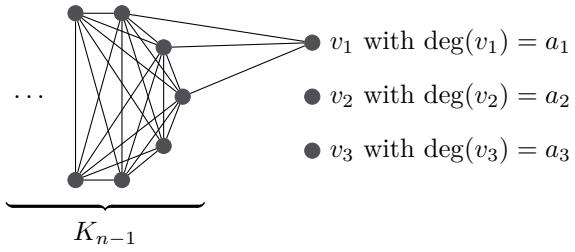


Figure 5 Example for $k = 3$, building a graph G with $n - 1 + \gamma(k-1)$ vertices and $\mathrm{sub}(\triangle, G) = h > \binom{n-1}{3}$. Since $\gamma(2) = 3$, there are a_1, a_2, a_3 such that $h - \binom{n-1}{3} = \binom{a_1}{2} + \binom{a_2}{2} + \binom{a_3}{2}$. Connecting a fresh vertex v_i with a_i vertices from the K_{n-1}, adds $\binom{a_i}{2}$ subgraphs $\triangle$ to G.

Thus, it remains to construct a graph with $\binom{n-1}{k} < h \leq \binom{n}{k}$ copies of K_k and $n+\gamma(k-1)-1$ vertices. Write $h' \coloneqq h - \binom{n-1}{k} \leq \binom{n-1}{k-1}$. By Theorem 18, there exist non-negative integers $a_1, \ldots, a_{\gamma(k-1)}$ such that $h' = \sum_{i=1}^{\gamma(k-1)} \binom{a_i}{k-1}$. It can be easily seen that $\binom{h}{k} > \binom{n}{k}$ for all integers $h > n \geq k \geq 1$. Hence, $\binom{a_i}{k-1} \leq h' \leq \binom{n-1}{k-1}$ implies that $a_i \leq n-1$ for all $1 \leq i \leq \gamma(k-1)$.

Define the graph G by taking the disjoint union of a clique K_{n-1} and fresh vertices $v_1, \ldots, v_{\gamma(k-1)}$. For $1 \leq i \leq \gamma(k-1)$, the vertex v_i is connected to an arbitrary selection of a_i many vertices of the clique. Note that this adds $\binom{a_i}{k-1}$ copies of K_k to the graph. The resulting graph on $n-1+\gamma(k-1)$ vertices satisfies $\mathrm{sub}(K_k, G) = \binom{n-1}{k} + \sum_{i=1}^{\gamma(k-1)} \binom{a_i}{k-1} = h$. For an example with $k = 3$, see Figure 5. ◀

For the special case of $k = 3$, i.e. $\triangle$, we can do slightly better than in Theorem 4. While Theorem 4 requires $\gamma(2) - 1 = 2$ extra vertices to realise any sensible h, we show in Theorem 19 that for large n one additional vertex suffices. While Theorem 4 builds on an $(n-1)$-vertex clique to which new vertices and edges are added, for Theorem 19 we start with an $(n+1)$-vertex clique and remove edges to realise the precise subgraph count. The proof of Theorem 19 can be found in the full version [4].

▶ **Theorem 19.** *For every $n \geq 130$ and $h \leq \binom{n}{3}$, there is a graph G on $n+1$ vertices such that* $\mathrm{sub}(\triangle, G) = h$.

6 Parametrised Complexity

For graph classes $\mathcal{F}$ and $\mathcal{G}$, we consider the parametrised version p-HomRec$(\mathcal{F}, \mathcal{G})$ of the homomorphism reconstructability problem. For an instance $(F_1, h_1), \ldots, (F_m, h_m)$, the parameter is $k \coloneqq \sum_{i=1}^{m} |V(F_i)|$. We aim for an fpt-algorithm, i.e. an algorithm that runs in $f(k)\,\mathrm{polylog}(h_1, \ldots, h_m)$ for some computable function f. By Theorem 15, p-HomRec$(\mathcal{F}, \mathcal{G})$ is para-NP-hard and thus we cannot expect to obtain an fpt-algorithm unless P = NP, cf. [17, Corollary 2.13], which means that we have to restrict the problem in some way. Surprisingly, it turns out that a certain restriction of p-HomRec$(\mathcal{F}, \mathcal{G})$ and of the analogous p-SubRec$(\mathcal{F}, \mathcal{G})$ are in FPT. Curiously, in FPT, one cannot even count homomorphisms or subgraphs from arbitrary graphs [12, 11]. Our algorithm has to make do with the integers from the input and cannot construct the graph G explicitly.

Given constraint graphs $\mathcal{I} \subseteq \mathcal{F}$, the overall strategy is to compute in time only depending on $\mathcal{I}$ a data structure representing the (infinite) set of all reconstructable vectors of homomorphism counts

$$R(\mathcal{I}) \coloneqq \left\{ \hom(\mathcal{I}, G) \in \mathbb{N}^{\mathcal{I}} \mid G \in \mathcal{G} \right\} \tag{1}$$

or analogously of subgraph counts. This data structure is required to admit a polynomial-time procedure for testing whether a given vector $h \in \mathbb{N}^{\mathcal{I}}$ is in this set. We identify various combinatorial conditions sufficient for guaranteeing the feasibility of this approach. To start with, we impose the following conditions on the graph classes $\mathcal{F}$ and $\mathcal{G}$:

▶ **Proviso 20.**
- **(i)** *membership in $\mathcal{G}$ is decidable,*
- **(ii)** *$\mathcal{G}$ is closed under taking induced subgraphs,*
- **(iii)** *$\mathcal{G}$ is closed under disjoint unions,*
- **(iv)** *all $F \in \mathcal{F}$ are connected.*

Items (iii) and (iv) imply that the set $R(\mathcal{I})$ of all reconstructable vectors is closed under addition. Indeed, for all connected graphs F and graphs G and H it holds that $\hom(F, G + H) = \hom(F, G) + \hom(F, H)$ and $\mathrm{sub}(F, G + H) = \mathrm{sub}(F, G) + \mathrm{sub}(F, H)$. Thus, we can use the vectors realised by small graphs, i.e. those which can be inspected in FPT time, to construct vectors realised by bigger graphs. More formally, writing

$$S(\mathcal{I}) \coloneqq \left\{ \hom(\mathcal{I}, G) \in \mathbb{N}^{\mathcal{I}} \;\middle|\; G \in \mathcal{G} \text{ with } |V(G)| \leq \max_{I \in \mathcal{I}} |V(I)| \right\}, \tag{2}$$

it holds that the set of all finite linear combinations of elements in $S(\mathcal{I})$ with coefficients from $\mathbb{N}$ is contained in $R(\mathcal{I})$, i.e. $\mathbb{N}S(\mathcal{I}) \subseteq R(\mathcal{I})$. The challenge is to ensure that all reconstructable vectors can be constructed in this way.

We require item (ii) to relate vectors realised by large graphs to those realised by small graphs, cf. Lemmas 22 and 24. However, this assumption is not sufficient to yield an fpt-algorithm. To that end, we make further assumptions which ensure that the set of realised vectors is in a sense linear and thus admits the aforementioned desired data structure. Combinatorially, these assumptions mean that the various constraints may not interact non-trivially.

6.1 Equi-Size Subgraph Constraints

The restriction we impose on p-SubRec$(\mathcal{F}, \mathcal{G})$ to put it in FPT is that all constraint graphs are of the same size:

▶ **Theorem 21.** *For graph classes $\mathcal{F}$, $\mathcal{G}$ as in Proviso 20, the following problem is in* FPT*:*

p-EquiSizeSubRec$(\mathcal{F}, \mathcal{G})$
Input Pairs $(F_1, h_1), \ldots, (F_m, h_m) \in \mathcal{F} \times \mathbb{N}$ with $\lvert V(F_1)\rvert = \cdots = \lvert V(F_m)\rvert =: k$
Parameter km
Question Is there a $G \in \mathcal{G}$ such that $\mathrm{sub}(F_i, G) = h_i$ for every $i \in [m]$?

Given an instance $\mathcal{I} \subseteq \mathcal{F}$, define $R(\mathcal{I})$ and $S(\mathcal{I})$ as in Equations (1) and (2) but with sub instead of hom. As argued in the previous section, we have $\mathbb{N}S(\mathcal{I}) \subseteq R(\mathcal{I})$. In the context of p-EquiSizeSubRec$(\mathcal{F}, \mathcal{G})$, the following Lemma 22 yields that $\mathbb{N}S(\mathcal{I}) = R(\mathcal{I})$.

▶ **Lemma 22.** *Let F and G be graphs. Then*

$$\mathrm{sub}(F, G) = \sum_{H \text{ s.t. } |V(H)|=|V(F)|} \mathrm{sub}(F, H)\, \mathrm{indsub}(H, G)$$

where the sum ranges over all isomorphism types of graphs H on $|V(F)|$ vertices.

Indeed, by Lemma 22, every vector $\mathrm{sub}(\mathcal{I}, G) \in R(\mathcal{I})$ is an $\mathbb{N}$-linear combination of the vectors $\mathrm{sub}(\mathcal{I}, H)$ where H has exactly k vertices. It is crucial that all graphs in $\mathcal{I}$ have the same number of vertices since otherwise any statement akin to Lemma 22 would involve negative coefficients stemming from Inclusion–Exclusion. In virtue of Lemma 22, testing membership in $R(\mathcal{I})$ reduces to solving a system of linear equations over $\mathbb{N}$.

▶ **Example 23.** Let $p, c \geq 0$ be integers. There exists a graph G with $\mathrm{sub}(\triangle\!\!-\!\!\cdot, G) = p$ and $\mathrm{sub}(\triangle, G) = c$ if and only if $p \geq 3c$.

Proof. By Lemma 22, there exists a graph G with $\mathrm{sub}(\triangle\!\!-\!\!\cdot, G) = p$ and $\mathrm{sub}(\triangle, G) = c$ if and only if the system $\binom{p}{c} = \left(\begin{smallmatrix} 1 & 3 \\ 0 & 1 \end{smallmatrix}\right)\binom{x}{y}$ has solutions $x, y \in \mathbb{N}$, i.e. $\mathrm{sub}(\triangle\!\!-\!\!\cdot, y\triangle + x\triangle\!\!-\!\!\cdot) = x + 3y$ and $\mathrm{sub}(\triangle, y\triangle + x\triangle\!\!-\!\!\cdot) = y$. The columns of this matrix correspond to the two graphs on three vertices which have a subgraph $\triangle\!\!-\!\!\cdot$ or $\triangle$, namely $\triangle\!\!-\!\!\cdot$ and $\triangle$. Solving this system yields that $y = c$ and $x = p - 3c$, as desired. ◀

The matrix constructed in Example 23 via Lemma 22 can clearly be computed in FPT. It remains to solve its system of linear equations, which is also possible in FPT [13].

Proof of Theorem 21. Let $\mathcal{I} \subseteq \mathcal{F}$ denote the instance. As observed above, $\mathbb{N}S(\mathcal{I}) = R(\mathcal{I})$. Write $\mathcal{H}$ for the set of all isomorphism types of graphs on k vertices. Write $A \in \mathbb{N}^{\mathcal{I} \times \mathcal{H}}$ for the matrix with entries $\mathrm{sub}(F, H)$ for $(F, H) \in \mathcal{I} \times \mathcal{H}$. This matrix can be computed in FPT. Then $b = (h_1, \ldots, h_m) \in R(\mathcal{I})$ if and only if $Ax = b$ has a solution over the non-negative integers. Testing the latter condition can be done in FPT by [13]. ◀

6.2 A Single Homomorphism Constraint

For p-HomRec$(\mathcal{F}, \mathcal{G})$, the restriction to ensure fixed parameter-tractability is that there is only one (connected) constraint graph.

▶ **Theorem 5.** *For graph classes $\mathcal{F}$, $\mathcal{G}$ as in Proviso 20, the following problem is in* FPT*:*

p-SingleHomRec$(\mathcal{F}, \mathcal{G})$
Input a graph $F \in \mathcal{F}$, an integer $h \in \mathbb{N}$ given in binary
Parameter $\lvert V(F)\rvert$
Question Does there exist a graph $G \in \mathcal{G}$ such that $\hom(F, G) = h$?

Define $R(F)$ and $S(F)$ as in Equations (1) and (2) replacing $\mathcal{I}$ by the singleton set $\{F\}$. As before, $\mathbb{N}S(F) \subseteq R(F)$. Dealing with p-SingleHomRec$(\mathcal{F}, \mathcal{G})$ is more complicated than tackling p-EquiSizeSubRec$(\mathcal{F}, \mathcal{G})$ in the sense that we will not be able to prove that $\mathbb{N}S(F) = R(F)$. In fact, these sets only coincide for large enough numbers. The key combinatorial identity is the following, whose proof is deferred to the full version [4]:

▶ **Lemma 24.** *Let F be a graph on k vertices. Then for all graphs G on more than k vertices,*

$$\hom(F, G) = \sum_{H \text{ s.t. } |V(H)| \leq k} \hom(F, H)\,\mathrm{indsub}(H, G)(-1)^{k-|V(H)|}\binom{|V(G)| - |V(H)| - 1}{k - |V(H)|},$$

where the sum ranges over all isomorphism types of graphs H on at most k vertices.

Lemma 24 yields that $R(F) \subseteq \mathbb{Z}S(F)$, i.e. every realised number is a linear coefficient of numbers in $S(F)$ with (not necessarily non-negative) integer coefficients. What allows us to obtain Theorem 5 is the purely number-theoretic observation that $\mathbb{N}S(F)$ and $\mathbb{Z}S(F)$ coincide on sufficiently large numbers. Lemma 25 is based on Bézout's identity and proven in the full version [4].

▶ **Lemma 25.** *Given $y_1, \dots, y_n \in \mathbb{N}$, one can compute integers y and N such that*

$$X \cap \{N, N+1, \dots\} = y\mathbb{N} \cap \{N, N+1, \dots\}$$

where $X \coloneqq \mathbb{N}\{y_1, \dots, y_n\}$.

▶ **Example 26.** Let $y_1 = 6$ and $y_2 = 16$. Their greatest common divisor is 2. The set of their $\mathbb{N}$-linear combinations is $X = \mathbb{N}\{6, 16\} = 2\mathbb{N} \setminus \{2, 4, 8, 10, 14, 20\}$, i.e. the set of all even numbers except $2, 4, 8, 10, 14, 20$.

This concludes the preparations for the proof of Theorem 5:

Proof of Theorem 5. The algorithm operates as follows: Compute $S(F)$ in time only depending on $|V(F)|$. Let y denote the greatest common divisor of the numbers in $S(F)$. By Lemmas 24 and 25, there exists a number N only depending on $|V(F)|$ such that for every $h \geq N$ there exists a graph G with $\hom(F, G) = h$ if and only if h is a multiple of y. This settles the question for all $h \geq N$. It remains to consider the case $h < N$. By Lemma 6, it suffices to consider graphs G of size bounded in $|V(F)|$ to conclude. ◀

To illustrate our algorithm, we include an example:

▶ **Example 27.** Consider the constraint graph $F =$  . Enumerating all graphs on at most 4 vertices yields that $S($  $) = \{0, 6, 16, 48\}$. For example, $\hom($  ,  $) = 6$, $\hom($, $) = 16$, and $\hom($, $) = 48$. Hence, $R($$)$ is a subset of the set in Example 26. It remains to check the finitely many exceptions $2, 4, 8, 10, 14, 20$. By Lemma 6, this can be done by inspecting graphs on at most $20 \cdot 4 = 80$ vertices.

7 Conclusion

This paper provides the first systematic study of the homomorphism reconstructability problem. Our results show that this deceivingly simple-to-state problem generally is hard – not only in terms of its computational complexity but also in terms of finding efficient algorithms for the simplest of cases, being subject to intricate phenomena from combinatorics and number theory. The following questions remain open and warrant further investigation:

- Is BHomRec($\mathcal{F}$) $\mathsf{NP}^{\#\mathsf{P}}$-complete for every class of unbounded treewidth $\mathcal{F}$, analogous to the #W[1]-completeness of #Hom($\mathcal{F}$) [12]? Is HomRec($\mathcal{F}$) NEXP-complete for the class of all graphs $\mathcal{F}$? Is there a graph class $\mathcal{F}$ for which HomRec($\mathcal{F}$) is not only NP-hard but actually NP-complete?
- While p-SingleHomRec($\mathcal{F}, \mathcal{G}$) is fpt, Theorem 3 implies that there exists a constant C such that p-HomRec($\mathcal{F}, \mathcal{G}$) restricted to instances with $\leq C$ constraints is para-NP-hard. What is the minimal such C? Is p-HomRec($\mathcal{F}, \mathcal{G}$) with two connected constraints fpt? Is there a sharp threshold from fixed-parameter tractability to para-NP-hardness?
- The proof of Theorem 12 suggests that in some cases the number of constraints can be decreased by increasing the number of connected components of the constraint graphs. Notably, a crucial ingredient in Proviso 20 is that the constraint graphs are connected. How do the parameters number of constraints and number of connected components affect the complexity of HomRec($\mathcal{F}, \mathcal{G}$)? What does the complexity hierarchy under these two parameters look like?
- In [18, 37], the functions $f\colon \mathcal{G} \to \mathbb{N}$ which are of the form $f = \hom(-, G)$ for some graph G were characterised. Here, $\mathcal{G}$ denotes the class of all graphs. For which finite graph classes $\mathcal{I}$ does a characterisation of functions $f\colon \mathcal{I} \to \mathbb{N}$ of this form exist? Our Theorem 3 implies that in some cases deciding whether a given f is of this form is NP-hard.

- Are there non-trivial examples of combinations of constraint graphs for which reconstructability is tractable? Is there an effective description of the yellow area in Figure 1?
- Is $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$ self-reducible [52]? That is, can we efficiently construct a graph G that realises the given constraints if we have access to an oracle for $\textsc{HomRec}(\mathcal{F}, \mathcal{G})$?
- What is the computational complexity of deciding whether homomorphism constraints are *approximately* reconstructable?
- More generally, one could study the complexity of questions about the set of graphs satisfying a given set of homomorphism constraints – such as computing its cardinality.
- How can one sample graphs satisfying homomorphism constraints uniformly at random?

References

1 Samson Abramsky, Tomáš Jakl, and Thomas Paine. Discrete Density Comonads and Graph Parameters. In Helle Hvid Hansen and Fabio Zanasi, editors, *Coalgebraic Methods in Computer Science*, pages 23–44, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-10736-8_2.

2 Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, April 1989. doi:10.1016/0166-218X(89)90031-0.

3 Albert Atserias, Phokion G. Kolaitis, and Wei-Lin Wu. On the Expressive Power of Homomorphism Counts. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13, 2021. doi:10.1109/LICS52264.2021.9470543.

4 Jan Böker, Louis Härtel, Nina Runde, Tim Seppelt, and Christoph Standke. The complexity of homomorphism reconstructibility. *CoRR*, abs/2310.09009, 2023. doi:10.48550/ARXIV.2310.09009.

5 Angela Bonifati, Irena Holubová, Arnau Prat-Pérez, and Sherif Sakr. Graph Generators: State of the Art and Open Challenges. *ACM Computing Surveys*, 53(2):1–30, March 2021. doi:10.1145/3379445.

6 Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2023. doi:10.1109/TPAMI.2022.3154319.

7 Surajit Chaudhuri and Moshe Y. Vardi. Optimization of Real Conjunctive Queries. In Catriel Beeri, editor, *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 25-28, 1993, Washington, DC, USA*, pages 59–70. ACM Press, 1993. doi:10.1145/153850.153856.

8 Yijia Chen, Jörg Flum, Mingjun Liu, and Zhiyang Xun. On Algorithms Based on Finitely Many Homomorphism Counts. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2022.32.

9 Radu Curticapean. Parity Separation: A Scientifically Proven Method for Permanent Weight Loss. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2016.47.

10 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*, STOC 2017, pages 210–223. Association for Computing Machinery, 2017. event-place: Montreal, Canada. doi:10.1145/3055399.3055502.

11 Radu Curticapean and Dániel 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*, FOCS '14, pages 130–139, USA, October 2014. IEEE Computer Society. doi:10.1109/FOCS.2014.22.

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

13 Peter Damaschke. Sparse solutions of sparse linear systems: Fixed-parameter tractability and an application of complex group testing. *Theoretical Computer Science*, 511:137–146, 2013. Exact and Parameterized Computation. doi:10.1016/j.tcs.2012.07.001.

14 Anuj Dawar, Tomás Jakl, and Luca Reggio. Lovász-Type Theorems and Game Comonads. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470609.

15 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász Meets Weisfeiler and Leman. *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, pages 40:1–40:14, 2018. doi:10.4230/LIPICS.ICALP.2018.40.

16 Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, August 2010. doi:10.1002/jgt.20461.

17 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2006. doi:10.1007/3-540-29953-X.

18 Michael Freedman, László Lovász, and Alexander Schrijver. Reflection positivity, rank connectivity, and homomorphism of graphs. *Journal of the American Mathematical Society*, 20:37–51, 2007. doi:10.1090/S0894-0347-06-00529-7.

19 M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, February 1976. doi:10.1016/0304-3975(76)90059-1.

20 Roman Glebov, Andrzej Grzesik, Ping Hu, Tamás Hubai, Daniel Král', and Jan Volec. Densities of 3-vertex graphs, April 2017. URL: http://arxiv.org/abs/1610.02446.

21 Martin Grohe. Counting Bounded Tree Depth Homomorphisms. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, pages 507–520, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3373718.3394739.

22 Martin Grohe. word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In Dan Suciu, Yufei Tao, and Zhewei Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 1–16. ACM, 2020. doi:10.1145/3375395.3387641.

23 Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism Tensors and Linear Equations. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 70:1–70:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 1868-8969. doi:10.4230/LIPIcs.ICALP.2022.70.

24 Xiaojie Guo and Liang Zhao. A Systematic Survey on Deep Generative Models for Graph Generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2022. doi:10.1109/TPAMI.2022.3214832.

25 Hamed Hatami and Serguei Norine. Undecidability of linear inequalities in graph homomorphism densities. *Journal of the American Mathematical Society*, 24(2):547–547, May 2011. doi:10.1090/S0894-0347-2010-00687-X.

26 Lane A. Hemaspaandra and Heribert Vollmer. The satanic notations: Counting classes beyond #P and other definitional adventures. *ACM SIGACT News*, 26(1):2–13, March 1995. doi:10.1145/203610.203611.

27 Hao Huang, Nati Linial, Humberto Naves, Yuval Peled, and Benny Sudakov. On the 3-Local Profiles of Graphs: On the 3-Local Profiles of Graphs. *Journal of Graph Theory*, 76(3):236–248, July 2014. doi:10.1002/jgt.21762.

28 Yannis E. Ioannidis and Raghu Ramakrishnan. Containment of Conjunctive Queries: Beyond Relations as Sets. *ACM Trans. Database Syst.*, 20(3):288–324, September 1995. Place: New York, NY, USA Publisher: Association for Computing Machinery. doi:10.1145/211414.211419.

29 E. Kamke. Verallgemeinerungen des Waring-Hilbertschen Satzes. *Mathematische Annalen*, 83(1-2):85–112, March 1921. doi:10.1007/BF01464230.

30 G. Katona. A theorem of finite sets. Theory of Graphs, Proc. Colloquium Tihany, Hungary 1966, 187-207, 1968.

31 Ker-I Ko and Wen-Guey Tzeng. Three Σ_2^P-complete problems in computational learning theory. *computational complexity*, 1(3):269–310, September 1991. doi:10.1007/BF01200064.

32 Swastik Kopparty and Benjamin Rossman. The homomorphism domination exponent. *European Journal of Combinatorics*, 32(7):1097–1114, 2011. Homomorphisms and Limits. doi:10.1016/j.ejc.2011.03.009.

33 Joseph B. Kruskal. The Number of Simplices in a Complex. In Richard Bellman, editor, *Mathematical Optimization Techniques*, pages 251–278. University of California Press, December 1963. doi:10.1525/9780520319875-014.

34 Jarosław Kwiecień, Jerzy Marcinkowski, and Piotr Ostropolski-Nalewaja. Determinacy of Real Conjunctive Queries. The Boolean Case. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 347–358. Association for Computing Machinery, 2022. doi:10.1145/3517804.3524168.

35 László Lovász. Coverings and colorings of hypergraphs. *Proc. 4th Southeastern Conference of Combinatorics, Graph Theory, and Computing*, pages 3–12, 1973. URL: https://zbmath.org/0322.05114.

36 László Lovász. *Combinatorial problems and exercises*. Akadémiai Kiado Elsevier Pub. Co, Budapest Amsterdam London, 2nd éd edition, 1993.

37 László Lovász and Alexander Schrijver. Semidefinite Functions on Categories. *Electron. J. Comb.*, 16(2), 2009. doi:10.37236/80.

38 László Lovász. *Large networks and graph limits*. Number volume 60 in American Mathematical Society colloquium publications. American Mathematical Society, Providence, Rhode Island, 2012. doi:10.1090/coll/060.

39 Stephen R. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, October 1982. doi:10.1016/0022-0000(82)90002-2.

40 Laura Mančinska and David E. Roberson. Quantum isomorphism is equivalent to equality of homomorphism counts from planar graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–672, 2020. doi:10.1109/FOCS46700.2020.00067.

41 Kenneth Manders and Leonard Adleman. NP-complete decision problems for quadratic polynomials. In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, STOC '76, pages 23–29, New York, NY, USA, May 1976. Association for Computing Machinery. doi:10.1145/800113.803627.

42 Melvyn B Nathanson. *Elementary Methods in Number Theory*, volume 195 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 2000. doi:10.1007/b98870.

43 V. I. Nečaev. On the representation of natural numbers as a sum of terms of the form $x(x+1)\cdots(x+n-1)/n!$. *Izvestiya Akad. Nauk SSSR. Ser. Mat.*, pages 485–498, 1953. URL: https://www.mathnet.ru/eng/im3481.

44 V. I. Nečaev. On the question of representing natural numbers by a sum of terms of the form $x(x+1)\dots(x+n-1)/n!$. *Trudy Mat. Inst. Steklov.*, 142:195–197, 270, 1976. Number theory, mathematical analysis and their applications. URL: https://www.mathnet.ru/eng/tm2571.

45 Hoang Nguyen and Takanori Maehara. Graph homomorphism convolution. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7306–7316. PMLR, 2020. URL: http://proceedings.mlr.press/v119/nguyen20c.html.

46 P. V. O'Neil. Ulam's conjecture and graph reconstructions. *The American Mathematical Monthly*, 77(1):35–43, 1970. doi:10.2307/2316851.

47 Ken Ono, Sinai Robins, and Patrick T. Wahl. On the representation of integers as sums of triangular numbers. *aequationes mathematicae*, 50(1):73–94, August 1995. doi:10.1007/BF01831114.

48 Jonathan Frederick Pollock. On the extension of the principle of Fermat's theorem of the polygonal numbers to the higher orders of series whose ultimate differences are constant. With a new theorem proposed, applicable to all the orders. *Abstracts of the Papers Communicated to the Royal Society of London*, 5:922–924, December 1851. doi:10.1098/rspl.1843.0223.

49 Alexander A. Razborov. On the Minimal Density of Triangles in Graphs. *Combinatorics, Probability and Computing*, 17(4):603–618, July 2008. doi:10.1017/S0963548308009085.

50 David E. Roberson. Oddomorphisms and homomorphism indistinguishability over graphs of bounded degree, June 2022. URL: http://arxiv.org/abs/2206.10321.

51 David E. Roberson and Tim Seppelt. Lasserre Hierarchy for Graph Isomorphism and Homomorphism Indistinguishability. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 101:1–101:18, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISSN: 1868-8969. doi:10.4230/LIPIcs.ICALP.2023.101.

52 Claus-Peter Schnorr. Optimal algorithms for self-reducible problems. In S. Michaelson and Robin Milner, editors, *Third International Colloquium on Automata, Languages and Programming, University of Edinburgh, UK, July 20-23, 1976*, pages 322–337. Edinburgh University Press, 1976.

53 Tim Seppelt. Logical Equivalences, Homomorphism Indistinguishability, and Forbidden Minors. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2023.82.

54 Janos Simon. *On Some Central Problems in Computational Complexity*. PhD thesis, Cornell University, USA, 1975.

55 Seinosuke Toda. PP is as Hard as the Polynomial-Time Hierarchy. *SIAM Journal on Computing*, 20(5):865–877, October 1991. doi:10.1137/0220053.

56 Jacobo Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38(3):752–773, July 1991. doi:10.1145/116825.116858.

57 Klaus W. Wagner. Some observations on the connection between counting and recursion. *Theoretical Computer Science*, 47:131–147, January 1986. doi:10.1016/0304-3975(86)90141-6.

58 Hinrikus Wolf, Luca Oeljeklaus, Pascal Kühner, and Martin Grohe. Structural Node Embeddings with Homomorphism Counts, August 2023. doi:10.48550/arXiv.2308.15283.