The Complexity of Approximately Counting Retractions

JACOB FOCKE, LESLIE ANN GOLDBERG, and STANISLAV ŽIVNÝ, University of Oxford, United Kingdom

Let G be a graph that contains an induced subgraph H. A *retraction* from G to H is a homomorphism from G to H that is the identity function on H. Retractions are very well-studied: Given H, the complexity of deciding whether there is a retraction from an input graph G to H is completely classified, in the sense that it is known for which H this problem is tractable (assuming P ≠ NP). Similarly, the complexity of (exactly) counting retractions from G to H is classified (assuming FP ≠ #P). However, almost nothing is known about approximately counting retractions. Our first contribution is to give a complete trichotomy for approximately counting retractions to graphs without short cycles. The result is as follows: (1) Approximately counting retractions to a graph H of girth at least 5 is in FP if every connected component of H is a star, a single looped vertex, or an edge with two loops. (2) Otherwise, if every component is an irreflexive caterpillar or a partially bristled reflexive path, then approximately counting retractions to H is equivalent to approximately counting the independent sets of a bipartite graph — a problem which is complete in the approximate counting complexity class $\mathrm{RH}\Pi_1$. (3) Finally, if none of these hold, then approximately counting retractions to H is equivalent to approximately counting the satisfying assignments of a Boolean formula.

Our second contribution is to locate the retraction counting problem for each H in the complexity landscape of related approximate counting problems. Interestingly, our results are in contrast to the situation in the exact counting context. We show that the problem of approximately counting retractions is separated both from the problem of approximately counting homomorphisms and from the problem of approximately counting list homomorphisms — whereas for exact counting all three of these problems are interreducible. We also show that the number of retractions is at least as hard to approximate as both the number of surjective homomorphisms and the number of compactions. In contrast, exactly counting compactions is the hardest of all of these exact counting problems.

CCS Concepts: • **Theory of computation → Approximation algorithms analysis**; **Problems, reductions and completeness**; • **Mathematics of computing → Combinatorics**.

Additional Key Words and Phrases: approximate counting, counting complexity, graph homomorphisms, retractions, surjective homomorphisms, graph compactions

ACM Reference Format:
Jacob Focke, Leslie Ann Goldberg, and Stanislav Živný. 2020. The Complexity of Approximately Counting Retractions. *ACM Trans. Comput. Theory* 1, 1, Article 1 (January 2020), 42 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A *homomorphism* from a graph G to a graph H is a function $h\colon V(G) \to V(H)$ such that, for all $\{u, v\} \in E(G)$, we have $\{h(u), h(v)\} \in E(H)$. For example, suppose that H is a path a, b, c. Then a homomorphism from G to H is a 3-colouring of G in which the colour classes $\{a, c\}$ and $\{b\}$ induce

Authors' address: Jacob Focke, jacob.focke@cs.ox.ac.uk; Leslie Ann Goldberg, leslie.goldberg@seh.ox.ac.uk; Stanislav Živný, standa.zivny@cs.ox.ac.uk, University of Oxford, Oxford, United Kingdom.

1942-3454/2020/1-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

a bipartition of G. Suppose that G itself contains a path A, B, C. Then a *retraction* from G to H is a homomorphism from G to H that maps A to a, B to b and C to c. In general, let G and H be graphs such that G contains a fixed copy of H as an induced subgraph. Then a retraction from G to H is a homomorphism from G to H that is the identity function on this fixed copy of H in G.

Retractions have been studied over a long period of time [25, 26, 31, 39]. In particular, the computational decision problem of determining whether there is a retraction from G to H is well-studied [13, 29, 45, 47, 50]. Retractions have also been studied under the name of *one-or-all list homomorphisms*, *pre-colouring extensions* or simply *extensions*, see, e.g., [1, 10–12, 35, 37, 43]. See Hell and Nešetřil's review article [30] for a more extensive list of such work.

Homomorphism *counting* problems have been researched extensively as well [3, 5, 6, 9, 14–16, 19, 20, 28, 33]. The problem of exactly counting retractions has been studied recently and a complete complexity classification is given in [14]. However, very little is known about *approximately* counting retractions. In this work we do two things. First we give a complexity trichotomy for approximately counting retractions for the class of graphs that have girth at least 5. Second we relate the complexity of approximately counting retractions to other approximate counting graph homomorphism problems.

1.1 First Contribution: A Trichotomy for Approximately Counting Retractions to Graphs of Girth at least 5

A *(self-)loop* is an edge from a vertex to itself. A *cycle* is a walk $w_0 w_1 \cdots w_k w_0$ where $k > 1$ and all vertices in $\{w_0, \dots, w_k\}$ are distinct. The *length* of the cycle is $k + 1$. We sometimes refer to length-3 cycles as "triangles" and to length-4 cycles as "squares". The *girth* of a graph H is the length of a shortest cycle in H. If H is acyclic (that is if H is a forest with possibly some loops) then its girth is infinity. In this work we give a complete complexity classification for the problem of approximately counting retractions to graphs that have a girth of at least 5 (Theorem 1.1). Thus, our classification applies to all graphs H except to those that contain 3-cycles or 4-cycles. We now informally introduce some notation and concepts in order to state this result.

Given a graph H, we use $\#\textsc{Ret}(H)$ to denote the problem of counting retractions to H, given as input a graph G containing a fixed copy of H.

To investigate the complexity of approximate counting problems, Dyer, Goldberg, Greenhill and Jerrum [5] introduce the concept of an approximation-preserving reduction (AP-reduction). Intuitively, an AP-reduction from a problem A to a problem B is an algorithm that is a "good" approximation to A if it has oracle access to a "good" approximation to B. We write $A \leq_{\mathrm{AP}} B$ if such an AP-reduction exists. Two problems that are studied in this paper appear frequently as benchmark problems in this line of research. #SAT is the problem of counting the satisfying assignments of a Boolean formula. This problem is complete for #P with respect to AP-reductions. #BIS is the problem of counting the independent sets of a bipartite graph. This problem is complete for the approximate counting complexity class $\mathrm{RH}\Pi_1$ (with respect to AP-reductions). While it is not believed that there is an efficient approximation algorithm for #BIS, it is also not believed that it is complete for #P with respect to AP-reductions.

While, in general, the vertices of H may or may not have loops, we will consider two special cases. We say that a graph is *irreflexive* if it does not contain any loops. We say that it is *reflexive* if every vertex has a loop. A *tree* may be irreflexive, reflexive, or neither, but it may not have any cycles. A *caterpillar* is an irreflexive tree which contains a path P such that all vertices outside of P have degree 1. A *partially bristled reflexive path* (formally defined in Definition 1.12) is a tree consisting of a reflexive path P, together with a (possibly empty) set of unlooped "bristle" vertices U and a matching connecting all of the vertices of U to "internal" vertices of P (vertices of P that are not endpoints of the path). A more formal definition, along with an example, is given in Section 1.5.

Theorem 1.1. *Let H be a graph of girth at least* 5.

i) If every connected component of H is an irreflexive star, a single looped vertex, or an edge with two loops, then #Ret(H) *is in* FP.
ii) Otherwise, if every connected component of H is an irreflexive caterpillar or a partially bristled reflexive path, then #Ret(H) *is approximation-equivalent to #BIS.*
iii) Otherwise, #Ret(H) *is approximation-equivalent to #SAT.*

Since there has been prior work on the problem of counting homomorphisms to trees [19], it is worth noting the special case of Theorem 1.1 where H is an irreflexive tree. In this case, #Ret(H) is in FP if H is a star. If H is a caterpillar but not a star, then #Ret(H) $\equiv_{\mathrm{AP}}$ #BIS. For all other irreflexive trees H, the problem #Ret(H) is #SAT-equivalent under AP-reductions. The special case where H is a reflexive tree is also easy to state. In this case, #Ret(H) is in FP if H is a single looped vertex or an edge with two loops. If H is a reflexive path with at least three vertices, then #Ret(H) $\equiv_{\mathrm{AP}}$ #BIS. Otherwise, #Ret(H) is #SAT-equivalent with respect to AP-reductions.

The special case of our classification where H is irreflexive is given in Theorem 2.3. It is slightly more general than what is given in Theorem 1.1 since it covers all irreflexive square-free graphs H, including those that have 3-cycles. The proof of Theorem 1.1 is given in Section 2.3.

1.2 Second Contribution: Locating #Ret(H) in the Approximate Counting Landscape

We locate the retraction counting problem in the complexity landscape of related homomorphism counting problems. Interestingly, it turns out that the complexity landscape for approximate counting looks very different from the one for exact counting.

We use $\mathcal{H}(G,H)$ to denote the set of homomorphisms from G to H and $N(G \to H)$ to denote the size of $\mathcal{H}(G,H)$. The following is the well-known homomorphism counting problem.

Name: #Hom(H).
Input: An irreflexive graph G.
Output: $N(G \to H)$.

Note that, in the problem #Hom(H), the input graph G is required to be irreflexive. This is standard in the field, and the reason for it is to make results stronger — typically it is the hardness results that are most challenging. In the problem #Ret(H), as we have informally defined it, it does not make sense to force G to be irreflexive, since it contains an induced copy of H, which may have loops. However, we can insist that G have no loops outside of the induced copy of H. Theorem 1.1 is still true under this restriction, and we incorporate this restriction into our formal definitions below. In order to give formal definitions, it is more natural to re-cast the retraction problem in terms of list homomorphisms, so we define these next.

Let $\mathrm{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$ be a set of "lists" indexed by the vertices of G. Each list S_v is a subset of $V(H)$. We say that a function $h\colon V(G) \to V(H)$ is a homomorphism from (G, S) to H (also called a *list homomorphism*) if h is a homomorphism from G to H and, for each vertex v of G, we have $h(v) \in S_v$. We use $\mathcal{H}((G,\mathrm{S}),H)$ to denote the set of homomorphisms from (G,S) to H and we use $N((G,\mathrm{S}) \to H)$ to denote the size of $\mathcal{H}((G,\mathrm{S}),H)$. We will be interested in the following generalisation of #Hom(H).

Name: #LHom(H).
Input: An irreflexive graph G and a collection of lists $\mathrm{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$.
Output: $N((G,\mathrm{S}) \to H)$.

As noted earlier, we will find it convenient to formally define the computational problem #Ret(H) in terms of list homomorphisms.

$$\begin{tikzcd}
 & \#\textsc{SHom}(H) \arrow[dr, "\text{Thm. 1.5}"] & & \#\textsc{LSHom}(H) \arrow[d, <->, "\text{Thm. 1.6}"] \\
\#\textsc{Hom}(H) \arrow[ur, "{[14, \text{Lem.37}]}"] \arrow[dr, "{[14, \text{Lem.38}]}"'] \arrow[rr, "/" marking, "\text{Obs. 1.2}", "\text{Cor. 1.4}"'] & & \#\textsc{Ret}(H) \arrow[r, "/" marking, "\text{Obs. 1.2}", "\text{Cor. 1.3}"'] & \#\textsc{LHom}(H) \\
 & \#\textsc{Comp}(H) \arrow[ur, "\text{Thm. 1.5}"'] & & \#\textsc{LComp}(H) \arrow[u, <->, "\text{Thm. 1.6}"']
\end{tikzcd}$$

Fig. 1. Approximate counting complexity landscape. An arrow from a problem A to a problem B means that there exists an AP-reduction from A to B. A struck through arrow corresponds to a reduction with a separation. The references for the reduction and the separation are given above and below the arrow, respectively.

Name: #Ret(H).
Input: An irreflexive graph G and a collection of lists $\mathbf{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$ such that, for all $v \in V(G)$, $|S_v| \in \{1, |V(H)|\}$.
Output: $N\big((G, \mathbf{S}) \to H\big)$.

The polynomial-time interreducibility between the problem #Ret(H) which we defined informally (with the restriction on loops in G) and the one defined here, is demonstrated by Feder and Hell [10, Theorem 4.1] who give a parsimonious reduction between them. (A reduction is parsimonious if it preserves the number of solutions.) This reduction also shows that the corresponding decision problems are polynomial-time interreducible.

We consider two more related counting problems, namely #SHom(H), the problem of counting vertex-surjective homomorphisms, and #Comp(H), the problem of counting edge-surjective homomorphisms, which are called *compactions*. We give their formal definitions at the beginning of Section 3. Both problems are well-studied in the decision setting [2, 21–23, 36, 44, 46, 48, 49]. All three of the problems #SHom(H), #Comp(H) and #Ret(H) can be interpreted as problems requiring one to count homomorphisms with some kind of surjectivity constraint. #LSHom(H) and #LComp(H) are the corresponding list homomorphism problems and these are also formally defined in Section 3.

A *separation* between two homomorphism-counting problems A and B is given by a parameter H for which A and B are of different complexity, subject to some complexity-theory assumptions.

Before stating our results we give an overview of the approximate counting complexity landscape in Figure 1. The results summarised in this figure are consistent with the results that are known concerning the corresponding decision problems, as surveyed by Bodirsky, Kára and Martin [2], but they are in contrast to the situation in the exact counting world. For exact counting, #Hom(H), #Ret(H) and #LHom(H) are interreducible. Also, all of the exact counting problems that we have mentioned reduce to #Comp(H) and #LComp(H) [14], as depicted in Figure 2. Moreover, #Comp(H) and #LComp(H) are separated from the remaining problems.

We now give our results in more detail. We start off with the following simple observation which follows immediately from the problem definitions.

Observation 1.2. *Let H be a graph. Then #Hom(H) $\leq_{\mathrm{AP}}$ #Ret(H) $\leq_{\mathrm{AP}}$ #LHom(H).*

As we will see later, the complexity of approximately counting homomorphisms is still open (despite a lot of work on the problem) — even if restricted to trees H. The complexity of approximately counting list homomorphisms is known, due to Galanis, Goldberg and Jerrum [16]. Thus,

$$\begin{tikzcd}
\boxed{\begin{array}{c}\#\textsc{Hom}(H),\ \#\textsc{SHom}(H),\ \#\textsc{Ret}(H),\\ \#\textsc{LHom}(H),\ \#\textsc{LSHom}(H)\end{array}} \arrow[r, "{[14]}", "/" marking] & \boxed{\#\textsc{Comp}(H),\ \#\textsc{LComp}(H)}
\end{tikzcd}$$

Fig. 2. Exact counting complexity landscape. All problems in the same box are interreducible with respect to polynomial-time Turing reductions. The arrow means that each problem in the box on the left-hand side reduces to each problem on the right-hand side using a polynomial-time Turing reduction. The arrow is struck through as there exists a separation between each problem on the left and each problem on the right.

Observation 1.2 indicates that #Ret(H) is an important intermediate problem, between the solved #LHom(H) and the wide-open #Hom(H).

The first interesting consequence of Theorem 1.1 is a separation between #Ret(H) and #LHom(H).

Corollary 1.3. *#Ret(H) and #LHom(H) are separated subject to the assumption that #BIS and #SAT are not AP-interreducible. In particular, if H is a partially bristled reflexive path with at least one unlooped vertex, then #Ret(H) $\equiv_{\mathrm{AP}}$ #BIS, whereas #LHom(H) $\equiv_{\mathrm{AP}}$ #SAT.*

The fact that #Ret(H) $\equiv_{\mathrm{AP}}$ #BIS for partially bristled reflexive paths follows from Theorem 1.1. The fact that #LHom(H) $\equiv_{\mathrm{AP}}$ #SAT is from [16], see Theorem 1.8 in the Related Work section.

As a second consequence, Theorem 1.1 separates #Ret(H) from #Hom(H), but in a different sense. For $q \geq 3$ let J_q be the irreflexive graph obtained from the q-leaf star by subdividing each edge. The graph J_3 is depicted in Figure 4 on page 10. From Goldberg and Jerrum [19] it is known that the problem #Hom(J_q) is AP-interreducible with the task of computing the partition function of the q-state ferromagnetic Potts model [40] — a well-studied model from statistical physics. Despite extensive work on this problem [17–19] it is only known to be #BIS-hard but is not known to be #BIS-easy or to be #SAT-hard (with respect to AP-reductions).

Corollary 1.4. *Let q be an integer with $q \geq 3$. #Hom(H) and #Ret(H) are separated subject to the assumption that approximately computing the partition function of the q-state ferromagnetic Potts model is not #SAT-hard. In particular, it follows from Theorem 1.1 that #SAT $\leq_{\mathrm{AP}}$ #Ret(J_q).*

In addition to these separations, we show that approximately counting retractions is at least as hard as approximately counting surjective homomorphisms and also at least as hard as approximately counting compactions. The latter is surprising as it is in contrast to known results for the corresponding exact counting problems (see Figure 2). Our proof uses an interesting Monte Carlo approach to AP-reductions and more details on this method are given in Section 1.3. The approach gives analogous reductions for the list versions of these problems for free.

Theorem 1.5. *Let H be a graph. Then #SHom(H) $\leq_{\mathrm{AP}}$ #Ret(H) and #Comp(H) $\leq_{\mathrm{AP}}$ #Ret(H).*

Theorem 1.6. *Let H be a graph. Then #LSHom(H) $\equiv_{\mathrm{AP}}$ #LHom(H) and #LComp(H) $\equiv_{\mathrm{AP}}$ #LHom(H).*

Using Theorem 1.5 and Corollary 1.3 we can deduce that #SHom(H) and #LHom(H) are also separated subject to the assumption that #BIS and #SAT are not AP-interreducible. The same holds for #Comp(H) and #LHom(H). Moreover, from Theorem 1.6 it follows that we can replace the problem #LHom(H) with #LSHom(H) or #LComp(H) in these separations.

Our reductions #SHom(H) $\leq_{\mathrm{AP}}$ #Ret(H) and #Comp(H) $\leq_{\mathrm{AP}}$ #Ret(H) allow us to state new #BIS-easiness results which are not limited to graphs of girth at least 5, namely the #BIS-easiness results in the following corollary.

Corollary 1.7. *Let H be one of the following:*

- *A reflexive proper interval graph but not a complete graph.*

- *An irreflexive bipartite permutation graph but not a complete bipartite graph.*

Then #SHom(H), #Comp(H) and #Ret(H) are #BIS-equivalent.

The #BIS-easiness results in Corollary 1.7 come from our Theorem 1.5 together with Observation 1.2 and the #BIS-easiness results for #LHom(H) given in Theorem 1.8 on page 7. The corresponding #BIS-hardness comes from [14, Theorem 35]).

The proof of Theorem 1.1 is given in Section 2.3. Theorem 1.5 is proved in Section 3.1 in the form of Corollaries 3.4 and 3.6. The proof of Theorem 1.6 is in Section 3.2.

1.3 Methods

In the proof of Theorem 1.1 we use several different techniques. In the #BIS-easiness proof for partially bristled reflexive paths (Lemma 2.4) we build upon a technique that was introduced by Dyer et al. [5] and extended by Kelk [33] to reduce the problem of approximately counting homomorphisms to the problem of approximately counting the downsets of a partial order. In order to obtain more general results, we formalise this technique and use it in the context of the constraint satisfaction framework. This framework is convenient for generating #BIS-easiness results, not only for counting homomorphisms but also for counting retractions, both in the setting of undirected graphs (as used in this work) and even in the setting of directed graphs.

In order to obtain the #SAT-hardness part of Theorem 1.1, we analyse, and classify, different local structures in graphs. The most difficult part is Lemma 2.30 which analyses distance-2 neighbourhood structures as defined in Section 1.5. This lemma requires intricate gadgets (based on simpler versions used in [5] and [33]), rather careful analysis, and classifying homomorphisms by type. Other #SAT-hardness results are easier to come by. For instance, we use modifications of, and a more careful analysis of, a gadget from [19] to prove the #SAT-hardness for irreflexive square-free graphs that have an induced subgraph J_3 (Lemma 2.2). Some hardness results are also based on NP-completeness results for the retraction decision problem that carry over to the approximate counting version.

The algorithms captured by the reductions #SHom$(H) \leq_{\mathrm{AP}}$ #Ret(H) and #Comp$(H) \leq_{\mathrm{AP}}$ #Ret(H) are based on a Monte Carlo approach (Lemma 3.3). We will discuss this approach here in the context of counting surjective homomorphisms to H. The details, and the related approach for counting compactions, are described in Section 3.1. The Monte Carlo approach is applicable for a reduction from #SHom(H) to #Ret(H) because #Ret(H) is a so-called self-reducible problem. Recall that the output of #Ret(H), given a graph G and lists $\mathbf{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$ such that, for all $v \in V(G)$, $|S_v| \in \{1, |V(H)|\}$, is the number of homomorphisms from $(G, \mathbf{S})$ to H. Using an oracle for approximating this number, it is also possible to *sample* a homomorphism from $(G, \mathbf{S})$ to H approximately uniformly at random (following the general method of Jerrum, Valiant, and Vazirani [32]). A naive approach for sampling surjective homomorphisms (and hence for approximately counting them) is as follows: Start with an input G to #SHom(H). Let $\mathbf{S}$ be the trivial set of lists $\mathbf{S} = \{S_v = V(H) \mid v \in V(G)\}$. Using the oracle for #Ret(H), obtain a random homomorphism from $(G, \mathbf{S})$ to H (which is just a random homomorphism from G to H). Reject (and repeat) if this homomorphism is not surjective. Eventually, we obtain a random surjective homomorphism from G to H, as required. While this approach is certainly straightforward, it does not lead to an *efficient* algorithm for sampling surjective homomorphisms because the number of surjective homomorphisms might be very small compared to the total number of homomorphisms.

Our method to shrink the sample space is based on the following fact. For every surjective homomorphism h from G to H there exists a constant-size set of vertices $U \subseteq V(G)$ such that the restriction of h to U is already surjective. We can enumerate all these constant-size sets U and use single vertex lists to fix their images. Consequently we obtain a (polynomial) number of instances $(G, \mathbf{S}^1), \ldots, (G, \mathbf{S}^k)$ of the problem #Ret(H). For $i \in \{1, \ldots, k\}$ let R_i be the set of homomorphisms from $(G, \mathbf{S}^i)$ to H. Then the set of surjective homomorphisms from G to H is the union $R = \bigcup_{i=1}^{k} R_i$. The final building block of our reduction is the idea that we can sample the union R by first sampling from the disjoint union $R^+ = \bigcup_{i=1}^{k}\{(h, i) \mid h \in R_i\}$. This idea is explained more generally, for instance, in [38, Section 11.2.2]. The point is, that we can sample uniformly from R^+ by using a #Ret(H) oracle, and the union R is relatively dense in the disjoint union R^+ (its size is at least $|R^+|/k$). So we can obtain a sample from R. Then the samples can be combined to obtain, with high probability, an approximate count. A lot of AP-reductions are based on the use of gadgets and we have not seen the use of Monte Carlo algorithms in AP-reductions before.

1.4 Related Work

Let H be a graph. It is well-known that the complexity of #LHom(H) is determined by the maximum complexity #LHom(C) for a connected component C of H. In the connected case the complexity is determined by the following theorem by Galanis et al. [16].

Theorem 1.8 ([16]). *Let H be a connected graph.*

(i) If H is an irreflexive complete bipartite graph or a reflexive complete graph, then #LHom(H) is in FP.
(ii) Otherwise, if H is an irreflexive bipartite permutation graph or a reflexive proper interval graph, then #LHom(H) is #BIS-equivalent under AP-reductions.
(iii) Otherwise, #LHom(H) is #SAT-equivalent under AP-reductions.

From our Theorem 1.1 and Theorem 1.8 we immediately obtain the separation between #Ret(H) and #LHom(H) given in Corollary 1.3. Note that when restricting to irreflexive or reflexive graphs, the classification of #Ret(H) for graphs of girth at least 5 is identical to the corresponding classification of the problem #LHom(H). A separation between these problems only occurs for graphs with at least one looped and one unlooped vertex.

The complexity of approximately counting homomorphisms in the absence of lists is still far from being resolved. Galanis, Goldberg and Jerrum [15] give a dichotomy for the problem in terms of #BIS.

Theorem 1.9 ([15]). *Let H be a connected graph. If H is a reflexive complete graph or an irreflexive complete bipartite graph, then #Hom(H) admits an FPRAS. Otherwise, #BIS $\leq_{\mathrm{AP}}$ #Hom(H).*

Surprisingly, even for the subclass of problems where H is an irreflexive tree, the complexity of approximately counting homomorphisms is not completely classified. The following partial classification, originally due to Goldberg and Jerrum [19], follows from Theorems 1.8 and 1.9.

Theorem 1.10 ([19]). *Let H be an irreflexive tree.*

i) If H is a star, then #Hom(H) is in FP.
ii) Otherwise, if H is a caterpillar, then #Hom(H) is #BIS-equivalent under AP-reductions.
iii) Otherwise, #Hom(H) is #BIS-hard under AP-reductions.

Note that, in general, for irreflexive trees H that are neither stars nor caterpillars it is open whether approximately counting homomorphisms is #BIS-equivalent, #SAT-hard or even none of the two. It is only known that #BIS AP-reduces to #Hom(H). However, there exist trees for which #Hom(H) is #SAT-equivalent with respect to AP-reductions (see [19, Section 5]).

The decision version of the retraction problem is formally defined as follows.[1]

[1]The literature is slightly inconsistent in the sense that the decision problem Ret(H) is often defined without the restriction that G is irreflexive. It is easy to see that the two versions (with and without the restriction) are polynomial-time interreducible since a looped vertex v of G with list S_v can be replaced with an irreflexive clique of size $|V(H)| + 1$ (all of whose members

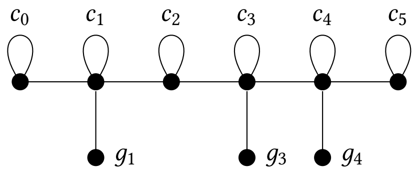

Fig. 3. Partially bristled reflexive path with $Q = 4$ and $S = \{1, 3, 4\}$.

Name: Ret(H).
Input: An irreflexive graph G and a collection of lists $\mathbf{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$ such that, for all $v \in V(G)$, $|S_v| \in \{1, |V(H)|\}$.
Output: Is $N\big((G, \mathbf{S}) \to H\big)$ positive?

Ret(H) is completely classified as a result of the recent proof of the CSP dichotomy conjecture [4, 52]. However, these proofs do not give a graph-theoretical characterisation. Feder, Hell, Jonsson, Krokhin and Nordh [13, Corollary 4.2, Theorem 5.1] give the following graph-theoretical characterisation for pseudotrees, where a pseudotree is a graph with at most one cycle. A graph H is called *loop-connected* if, for every connected component C of H, the looped vertices in C induce a connected subgraph of C.

Theorem 1.11 ([13]). *Let H be a pseudotree. Then Ret(H) is* NP*-complete if any of the following hold:*

- *H is not loop-connected,*
- *H contains a cycle of size at least* 5*,*
- *H contains a reflexive cycle of size* 4 *or*
- *H contains an irreflexive cycle of size* 3*.*

Otherwise Ret(H) is in P.

Finally, the complexity of exactly counting retractions is completely classified. It is in FP if every connected component of H is a reflexive complete graph or an irreflexive complete bipartite graph and #P-complete otherwise [14].

1.5 Preliminaries

For a positive integer n let $[n] = \{1, \ldots, n\}$. As partially bristled reflexive paths appear in a number of our results we give a more formal definition of this class of graphs. We also give an example in Figure 3.

Definition 1.12. A *partially bristled reflexive path* is a reflexive path, or a tree with the following form. Let Q be a positive integer and let S be a non-empty subset of $[Q]$. Then $V(H) = \{c_0, \ldots, c_{Q+1}\} \cup \bigcup_{i \in S}\{g_i\}$ and $E(H) = \bigcup_{i=0}^{Q}\{c_i, c_{i+1}\} \cup \bigcup_{i=0}^{Q+1}\{c_i, c_i\} \cup \bigcup_{i \in S}\{c_i, g_i\}$.

For a graph H and a vertex $u \in V(H)$ we define the *(distance-*1*) neighbourhood of u* as $\Gamma(u) = \{v \in V(H) \mid \{v, u\} \in E(H)\}$. Similarly, the *distance-*2 *neighbourhood of u* is defined as $\Gamma^2(u) = \{v \in V(H) \mid \exists w \in V(H) : \{v, w\}, \{w, u\} \in E(H)\}$. Let U be a subset of $V(H)$. Then $\Gamma(U) = \bigcap_{u \in U} \Gamma(u)$ is the *set of common neighbours* of the vertices in U. The set of vertices that have a neighbour in S is denoted by $\Phi(S) = \bigcup_{v \in S} \Gamma(v)$.

have list S_v) without changing whether or not there is a homomorphism to H. Thus, results stated for one version apply to the other.

We will also use the concept of *induced graphs*. Given a subset U of $V(H)$, the graph $H[U] = (U, \{\{u_1, u_2\} \in E(H) \mid u_1, u_2 \in U\})$ is called *the subgraph of H induced by U*. The graph $H' = (V', E')$ is a *subgraph* of $H = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$.

For the complexity theory part of our work we restate some standard definitions taken from [38, Definitions 11.1, 11.2, Exercise 11.3]. A randomised algorithm gives an (ε, δ)*-approximation* for the value V if the output X of the algorithm satisfies $\Pr(|X - V| \leq \varepsilon V) \geq 1 - \delta$. Slightly overloading the notation, an (ε, δ)*-approximation* for a problem V is a randomised algorithm which, given an input x and parameters $\varepsilon,\ \delta \in (0, 1)$, outputs an (ε, δ)-approximation for $V(x)$. A *randomised approximation scheme* (RAS) for a problem V is a $(\varepsilon, 1/4)$-approximation of V. A RAS is called *fully polynomial* (FPRAS) if it runs in time that is polynomial in $1/\varepsilon$ and the size of the input x.

Intuitively, an *approximation-preserving reduction* (AP-reduction) from a problem A to a problem B is an algorithm that yields an FPRAS for A if it has access to an FPRAS for B. We state the technical definition from [5]. An approximation-preserving reduction from a problem A to a problem B is a probabilistic oracle Turing machine M which takes as input an instance x of A and a parameter $\varepsilon \in (0, 1)$, and satisfies the following three properties: 1) Every oracle call made by M is of the form (y, δ), where y is an instance of B and $\delta \in (0, 1)$ with $1/\delta \in \text{poly}(|x|, 1/\varepsilon)$ specifies the precision of approximation. 2) The Turing machine M is a RAS for A whenever the oracle is a RAS for B. 3) The runtime of M is polynomial in $|x|$ and $1/\varepsilon$.

Sometimes it is useful to switch between different notions of accuracy. To this end we use the following observation which follows immediately from the Taylor expansion of the exponential function.

Observation 1.13. *Let ε be in $(0, 1)$. Then $1 + \varepsilon \leq e^{\varepsilon} \leq 1 + 2\varepsilon$ and $1 - \varepsilon \leq e^{-\varepsilon} \leq 1 - \varepsilon/2$.*

We conclude this section with some simple remarks regarding the connectivity of graphs when investigating the complexity of counting retractions. We will show that in the context of approximately counting retractions we can restrict to connected graph without loss of generality. We define the following problem which restricts the input to connected graphs.

Name: $\#\text{Ret}(H)^{\text{conn}}$.

Input: An irreflexive *connected* graph G and a collection of lists $\mathbf{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$ such that, for all $v \in V(G)$, $|S_v| \in \{1, |V(H)|\}$.

Output: $N\big((G, \mathbf{S}) \to H\big)$.

The following observation is well known.

Observation 1.14. *Let H be a graph. Then $\#\text{Ret}(H) \equiv_{\text{AP}} \#\text{Ret}(H)^{\text{conn}}$.*

Proof. The fact that $\#\text{Ret}(H) \geq_{\text{AP}} \#\text{Ret}(H)^{\text{conn}}$ is trivial. We now show that $\#\text{Ret}(H) \leq_{\text{AP}} \#\text{Ret}(H)^{\text{conn}}$. Let $(G, \mathbf{S})$ be an instance of $\#\text{Ret}(H)$ and let $\varepsilon \in (0, 1)$ be the desired precision. Let $C_1, \dots, C_k$ be the connected components of G. For each $i \in [k]$, let $\mathbf{S}_i = \{S_v \mid v \in V(C_i)\}$. Then $N\big((G, \mathbf{S}) \to H\big) = \prod_{i=1}^{k} N\big((C_i, \mathbf{S}_i) \to H\big)$. The algorithm which, for each $i \in [k]$, makes a $\#\text{Ret}(H)^{\text{conn}}$ oracle call with precision $\delta = \varepsilon/k$ and input $(C_i, \mathbf{S}_i)$, and returns the product of outputs, approximates $N\big((G, \mathbf{S}) \to H\big)$ with the desired precision. □

Remark 1.15. Let H be a graph with connected components $H_1, \dots, H_k$ and let $(G, \mathbf{S})$ be an input to $\#\text{Ret}(H)^{\text{conn}}$. For $j \in [k]$ let $S_v^j = S_v \cap V(H_j)$ and let $\mathbf{S}^j = \{S_v^j \mid v \in V(G)\}$. Then, as G is connected, it holds that

$$N\big((G, \mathbf{S}) \to H\big) = \sum_{j \in [k]} N\big((G, \mathbf{S}^j) \to H_j\big).$$

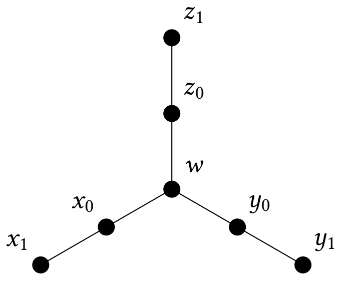

Fig. 4. The graph J_3.

Therefore, given an oracle for $\#\textsc{Ret}(H_j)^{\text{conn}}$ for each $j \in [k]$, we obtain an algorithm for $\#\textsc{Ret}(H)^{\text{conn}}$. By Observation 1.14 this means that given an oracle for $\#\textsc{Ret}(H_j)$ for each $j \in [k]$, we obtain an algorithm for $\#\textsc{Ret}(H)$.

In the opposite direction, it is straightforward to see that for each $j \in [k]$ we have that $\#\textsc{Ret}(H_j)^{\text{conn}} \leq_{\text{AP}} \#\textsc{Ret}(H)^{\text{conn}}$ (and therefore $\#\textsc{Ret}(H_j) \leq_{\text{AP}} \#\textsc{Ret}(H)$). The details are as follows: Let $(G, \mathbf{S})$ be an input to $\#\textsc{Ret}(H_j)^{\text{conn}}$. If all lists in $\mathbf{S}$ have size 1, computing $N\big((G, \mathbf{S}) \to H_j\big)$ is trivial. Otherwise we fix some vertex $v \in V(G)$ with $S_v = V(H_j)$. For each $u \in V(H)$ and $w \in V(G)$ we define

$$S_w^u = \begin{cases} \{u\}, & \text{if } w = v \\ S_w, & \text{if } |S_w| = 1 \\ V(H), & \text{otherwise.} \end{cases}$$

and $\mathbf{S}^u = \{S_w^u \mid w \in V(G)\}$. As G is connected, a homomorphism from G to H maps all vertices of G to the same connected component of H. Therefore, $N\big((G, \mathbf{S}) \to H_j\big) = \sum_{u \in V(H_j)} N\big((G, \mathbf{S}^u) \to H\big)$. This shows that $\big|V(H_j)\big|$ calls to a $\#\textsc{Ret}(H)^{\text{conn}}$ oracle are sufficient to approximate the number of retractions to a component H_j of H.

2 APPROXIMATELY COUNTING RETRACTIONS TO GRAPHS WITHOUT SHORT CYCLES

First we study the complexity of approximately counting retractions to graphs of girth at least 5. We start off by restricting to irreflexive graphs in Section 2.1. The corresponding classification (Theorem 2.3) is for irreflexive square-free graphs. Subsequently, in Section 2.2, we consider graphs that have at least one loop.

2.1 Irreflexive Square-free Graphs

The goal of this section is to prove Theorem 2.3. The most difficult part is Lemma 2.2, which shows that, if H is a square-free graph containing an induced J_3, then $\#\text{SAT} \leq_{\text{AP}} \#\textsc{Ret}(H)$.

The proof of Lemma 2.2 generalises ideas from the proof of Lemma 3.6 of [19], so we start with some definitions from there. A *multiterminal cut* of a graph G with distinguished vertices α, β and γ (called *terminals*) is a set of edges $E' \subseteq E(G)$ that disconnects the terminals (i.e. ensures that there is no path in $(V(G), E(G) \setminus E')$ that connects any two distinct terminals). The *size* of a multiterminal cut is its cardinality. We consider the following computational problem.

Name: $\#\textsc{MultiterminalCut}(3)$.
Input: A connected irreflexive graph G with 3 distinct terminals $\alpha,\ \beta,\ \gamma \in V(G)$ and a positive integer B. The input has the property that every multiterminal cut has size at least B.
Output: The number of size-B multiterminal cuts of G with terminals α, β and γ.

For motivation we consider the case where H is a tree. Suppose that H is an irreflexive tree with an induced J_3, labelled as in Figure 4. Lemma 3.6 of [19] gives an AP-reduction from the problem #MULTITERMINALCUT(3) to the problem of counting "weighted" homomorphisms to H. In fact, the weights used in the proof are Boolean values, so the proof actually reduces #MULTITERMINALCUT(3) to the problem of counting list homomorphisms to H. Given an input G, α, β, γ of the problem #MULTITERMINALCUT(3), a homomorphism instance is created in which lists ensure that the terminals α, β and γ are mapped to the vertices x_0, y_0 and z_0 of J_3, respectively. Lists also ensure that all other vertices of G are mapped to $\{x_0, y_0, z_0\}$. Finally, lists ensure that all remaining vertices of the homomorphism instance are mapped to the vertices $\{w, x_1, y_1, z_1\}$ of J_3. Our proof shows how to refine the gadgets so that lists have size 1 or size $|V(H)|$. Thus, our reduction is to the more refined problem #RET(H). We also show how to handle graphs H that are not trees (as long as they are square-free). The details are given in the proof of Lemma 2.2. We use the following technical lemma, known as Dirichlet's approximation lemma, which bounds the extent to which reals can be approximated by integers. Using this lemma is a standard technique in this line of research (see for instance [15]).

LEMMA 2.1 ([41, P. 34]). *Let $\lambda_1, \dots, \lambda_d > 0$ be real numbers and N be a natural number. Then there exist positive integers $p_1, \dots, p_d, r$ with $r \leq N$ such that $|r\lambda_i - p_i| \leq 1/N^{1/d}$ for every $i \in [d]$.*

Using Lemma 2.1, we can prove our main lemma.

LEMMA 2.2. *Let H be a square-free graph that contains an induced J_3. Then #SAT $\leq_{\mathrm{AP}}$ #RET(H).*

PROOF. Suppose that H is a square-free graph with q vertices and an induced J_3, which we label as shown in Figure 4. The problem #MULTITERMINALCUT(3) is shown in [19, Lemma 3.5] to be equivalent to #SAT with respect to AP-reductions. We will give a reduction from #MULTITERMINALCUT(3) to #RET(H).

Let $G, \alpha, \beta, \gamma, B$ be an instance of #MULTITERMINALCUT(3) with $n = |V(G)|$ and let ε be an error bound in $(0, 1)$. From this instance we construct an input $(J, \mathbf{S})$ to #RET(H) as follows. Each of the terminals α, β and γ will be vertices of J. J will also have a vertex ω which is distinct from α, β and γ. Let s_α, s_β and s_γ be positive integers (we will give their precise values later). For every edge $e = \{u, v\} \in E(G)$ we define the set of vertices

$$V'(e) = \{(e, \alpha, 1), \dots, (e, \alpha, s_\alpha), (e, \beta, 1), \dots, (e, \beta, s_\beta), (e, \gamma, 1), \dots, (e, \gamma, s_\gamma)\}.$$

The label "α" in the name of the vertex (e, α, i) indicates that, in the instance $(J, \mathbf{S})$, this vertex will be adjacent to α. The labels "β" and "γ" are similar. The label "e" in the name of the vertex (e, α, i) indicates that this vertex is in $V'(e)$.

For each edge $e = \{u, v\} \in E(G)$ we then define a graph $J(e)$ with vertex set $V(J(e)) = V'(e) \cup \{\alpha, \beta, \gamma, \omega\}$. Note that the vertices in $V'(e)$ are distinct for each edge e whereas α, β, γ and ω are identical for all e. The edge set $E(J(e))$ of the graph $J(e)$ is defined as shown in Figure 5.

Then we define $J = \left(V(G) \cup \{\omega\} \cup \bigcup_{e \in E(G)} V'(e), \bigcup_{e \in E(G)} E(J(e))\right)$. Intuitively, J is constructed from the graph G by replacing each edge $e \in E(G)$ with the corresponding graph $J(e)$. Since G is connected, every vertex $v \in V(G)$ is a member of some edge in $E(G)$. This ensures that $\{v, \omega\}$ is an edge of J.

Next we define the set of lists $\mathbf{S}$ in the instance $(J, \mathbf{S})$. We set $S_\omega = \{w\}$, $S_\alpha = \{x_0\}$, $S_\beta = \{y_0\}$, $S_\gamma = \{z_0\}$ and $S_v = V(H)$ for all $v \in V(J) \setminus \{\alpha, \beta, \gamma, \omega\}$. Then we define $\mathbf{S} = \{S_v \subseteq V(H) \mid v \in V(J)\}$. This is depicted in Figure 6.

We now show how a multiterminal cut of G, α, β, γ corresponds to a certain set of homomorphisms from $(J, \mathbf{S})$ to H. Every multiterminal cut E' of G, α, β, γ induces a partition of $V(G)$ into connected components. These are the connected components of $(V(G), E(G) \setminus E')$. Let $\kappa(E')$ be the number of

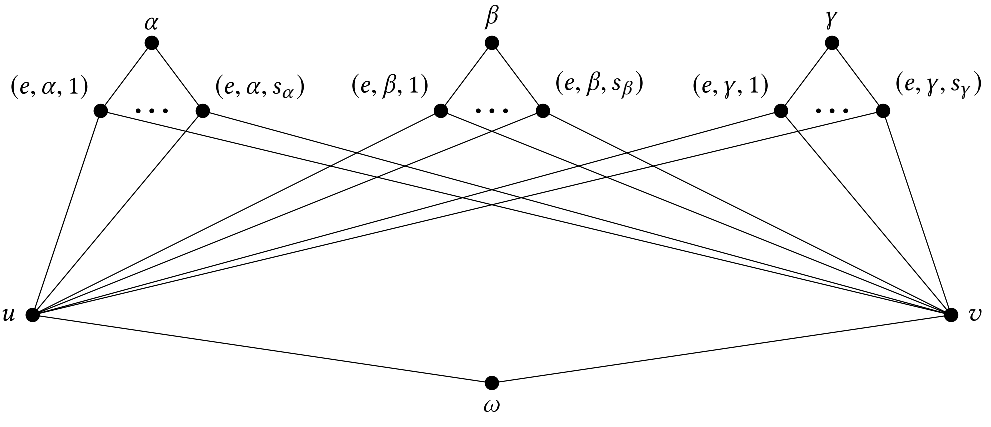

Fig. 5. The graph $J(e)$ for $e = \{u, v\}$.

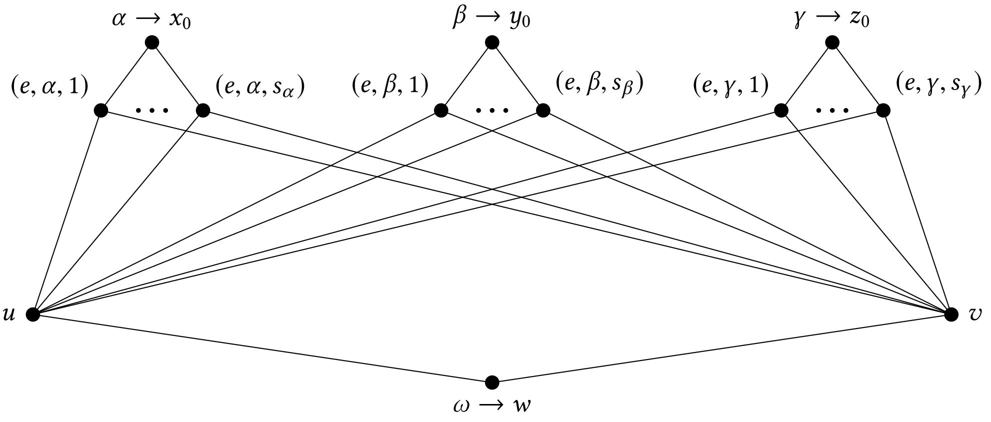

Fig. 6. The graph $J(e)$ for $e = \{u, v\}$. A label of the form $a \longrightarrow b$ means that the vertex $a \in V(G)$ is pinned to $b \in V(H)$ since $S_a = \{b\}$.

these components. Let $\Gamma(w)$ be the set of neighbours of vertex w in H and let $d_w \geq 3$ be the degree of w in H. Let $\Psi(E')$ be the set of functions $\psi\colon V(G) \longrightarrow \Gamma(w)$ such that

- ψ maps the vertices of the components containing the terminals α, β, and γ to x_0, y_0 and z_0, respectively, and
- the set of bichromatic edges $\{\{u, v\} \in E(G) \mid \psi(u) \neq \psi(v)\}$ is exactly the cut E'.

Then

$$|\Psi(E')| = {d_w}^{\kappa(E')-3}. \tag{1}$$

Now, for every $\psi \in \Psi(E')$, let $X(\psi) = \{\{u, v\} \in E(G) \mid \psi(u) = \psi(v) = x_0\}$. Note that $X(\psi)$ is the set of monochromatic edges in G whose endpoints are mapped to x_0. Similarly, let $Y(\psi) = \{\{u, v\} \in E(G) \mid \psi(u) = \psi(v) = y_0\}$ and $Z(\psi) = \{\{u, v\} \in E(G) \mid \psi(u) = \psi(v) = z_0\}$.

Given a multiterminal cut E' of G, α, β, γ and a map $\psi \in \Psi(E')$, we say that a homomorphism $\sigma \in \mathcal{H}((J, \mathrm{S}), H)$ *agrees* with ψ if, for all $v \in V(G)$, we have $\sigma(v) = \psi(v)$. Let $\Sigma(\psi)$ be the set of all $\sigma \in \mathcal{H}((J, \mathrm{S}), H)$ that agree with ψ. Given a multiterminal cut E' of G, α, β, γ we say that a homomorphism $\sigma \in \mathcal{H}((J, \mathrm{S}), H)$ *agrees* with E' if there is a $\psi \in \Psi(E')$ such that σ agrees with ψ. Let $Z_{E'} = \sum_{\psi \in \Psi(E')} |\Sigma(\psi)|$ be the number of homomorphisms from (J, S) to H that agree with the cut E'.

Now consider a multiterminal cut E' of G, α, β, γ. We will bound $Z_{E'}$ by considering two cases. Recall that B is a positive integer and part of the instance of #MULTITERMINALCUT(3).

Case 1: $|E'| = B$.

If $\kappa(E') \geq 4$ then, since G is connected, the input G, α, β, γ has a multiterminal cut of size less than B, which contradicts the definition of #MULTITERMINALCUT(3). Hence, it must be the case that $\kappa(E') = 3$, which means that $|\Psi(E')| = 1$. For the single $\psi \in \Psi(E')$ we have $|X(\psi)| + |Y(\psi)| + |Z(\psi)| = |E(G)| - B$. We will consider the possible homomorphisms $\sigma \in \Sigma(\psi)$.

Let $d_x, d_y, d_z \geq 2$ be the degrees of x_0, y_0 and z_0 in H, respectively.

- Consider any edge e of G that is not in the cut E'. Then, as $|E'| = B$, e has to be in $X(\psi)$, $Y(\psi)$ or $Z(\psi)$.
  - Suppose that e is in $X(\psi)$. Consider a vertex (e, β, i) of $J(e)$. This vertex is adjacent to the terminal β, which is mapped to y_0 by ψ and also to its endpoints, which are mapped to x_0 by ψ. Thus, σ has to map (e, β, i) to a mutual neighbour (in H) of x_0 and y_0. Since H is square-free, the only possibility is vertex w. Similarly, σ has to map each vertex (e, γ, i) of $J(e)$ to w. There is more choice concerning each vertex (e, α, i) of $J(e)$ — the homomorphism σ can map this vertex to any of the d_x neighbours of x_0 in H. Putting this together, the edge e contributes a factor of ${d_x}^{s_\alpha}$ to the number of homomorphisms in $\Sigma(\psi)$.
  - Suppose that e is in $Y(\psi)$. Similarly, the edge e contributes a factor of ${d_y}^{s_\beta}$ to $|\Sigma(\psi)|$.
  - Suppose that e is in $Z(\psi)$. Similarly, the edge e contributes a factor of ${d_z}^{s_\gamma}$ to $|\Sigma(\psi)|$.
- Consider any edge $e = \{u, v\}$ of G that is in the cut E'. Then a homomorphism $\sigma \in \Sigma(\psi)$ has to map all vertices of $V'(e)$ to a common neighbour of $\psi(u)$ and $\psi(v)$ in H. Since $\psi(u) \neq \psi(v)$ and H is square-free, the only possibility is to map all vertices of $V'(e)$ to w. Thus, the edge e contributes a factor of 1 to $|\Sigma(\psi)|$.

Putting all of this together, we have

$$Z_{E'} = {d_x}^{s_\alpha |X(\psi)|} {d_y}^{s_\beta |Y(\psi)|} {d_z}^{s_\gamma |Z(\psi)|}.$$

Now our goal is to choose s_α, s_β and s_γ so that $Z_{E'}$ depends only on the size of E' rather than on the sizes of $X(\psi)$, $Y(\psi)$ and $Z(\psi)$. (Intuitively, we want to design our graph $J(e)$ in such a way that it balances out the weights that are induced by the different degrees of x_0, y_0 and z_0.) We are limited by the fact that s_α, s_β and s_γ have to be integers. We use Lemma 2.1 to get around this. We set $\delta' = \log_q e^{\varepsilon/42}$ which we will use in the error bound of the Dirichlet approximation (the reasons behind our choice of δ' will become clear at the end of the proof). Note that $1/\delta' \in \mathrm{poly}(\varepsilon^{-1})$. Further, let $s = 2 + |E(G)| + \lceil \log_2 q \rceil |V(G)|$. We use Lemma 2.1 to approximate the real values $\lambda_1 = \log_{d_x}(2^s)$, $\lambda_2 = \log_{d_y}(2^s)$ and $\lambda_3 = \log_{d_z}(2^s)$ and obtain positive integers p_1, p_2, p_3 and r with $r \leq (n^2/\delta')^3 \in \mathrm{poly}(n, \varepsilon^{-1})$ such that for all $i \in \{1, 2, 3\}$ we have $|r\lambda_i - p_i| \leq \delta'/n^2$. Note that $p_1, p_2, p_3 \in \mathrm{poly}(n, \varepsilon^{-1})$. We set $s_\alpha = p_1$, $s_\beta = p_2$ and $s_\gamma = p_3$ to obtain

$$\begin{aligned}
Z_{E'} &= {d_x}^{p_1 |X(\psi)|} {d_y}^{p_2 |Y(\psi)|} {d_z}^{p_3 |Z(\psi)|} \\
&\leq {d_x}^{(r\lambda_1 + \delta'/n^2)|X(\psi)|} {d_y}^{(r\lambda_2 + \delta'/n^2)|Y(\psi)|} {d_z}^{(r\lambda_3 + \delta'/n^2)|Z(\psi)|} \\
&\leq 2^{sr(|X(\psi)| + |Y(\psi)| + |Z(\psi)|)} q^{\delta'/n^2 (|X(\psi)| + |Y(\psi)| + |Z(\psi)|)}
\end{aligned}$$

where we used the fact that $d_x, d_y, d_z \leq q$. Since $|X(\psi)| + |Y(\psi)| + |Z(\psi)| = |E(G)| - B \leq n^2$ it holds that

$$Z_{E'} \leq q^{\delta'} 2^{sr(|E(G)| - B)}.$$

Analogously we obtain $q^{-\delta'} 2^{sr(|E(G)|-B)} \le Z_{E'}$.

Let $Z^* = 2^{sr(|E(G)|-B)}$ (this value will be used later in the proof). Then

$$q^{-\delta'} Z^* \le Z_{E'} \le q^{\delta'} Z^*. \tag{2}$$

(End of Case 1.)

Case 2: $|E'| > B$.

In this case we have $\kappa(E') \ge 3$. For any $\psi \in \Psi(E')$, as in Case 1, each edge in $X(\psi)$ contributes a factor of $d_x^{s_\alpha}$ to $|\Sigma(\psi)|$, each edge in $Y(\psi)$ contributes a factor of $d_y^{s_\beta}$, and each edge in $Z(\psi)$ contributes a factor of $d_z^{s_\gamma}$.

Consider any $\psi \in \Psi(E')$ and let $E'' = E(G) \setminus (X(\psi) \cup Y(\psi) \cup Z(\psi))$. Then E'' consists of edges in E' and edges in $\{\{u, v\} \in E(G) \mid \psi(u) = \psi(v) \text{ and } \psi(u) \notin \{x_0, y_0, z_0\}\}$. Any edge e in E'' contributes a factor of 1 to $|\Sigma(\psi)|$ as every vertex in $V'(e)$ has to be mapped to w. Putting all of this together and simplifying as in Case 1, we have

$$\begin{aligned} Z_{E'} &= \sum_{\psi \in \Psi(E')} {d_x}^{p_1|X(\psi)|} {d_y}^{p_2|Y(\psi)|} {d_z}^{p_3|Z(\psi)|} \\ &\le \sum_{\psi \in \Psi(E')} q^{\delta'} 2^{sr(|X(\psi)|+|Y(\psi)|+|Z(\psi)|)}. \end{aligned}$$

We can analogously derive a lower bound for $Z_{E'}$ to obtain

$$q^{-\delta'} \sum_{\psi \in \Psi(E')} 2^{sr(|X(\psi)|+|Y(\psi)|+|Z(\psi)|)} \le Z_{E'} \le q^{\delta'} \sum_{\psi \in \Psi(E')} 2^{sr(|X(\psi)|+|Y(\psi)|+|Z(\psi)|)} \tag{3}$$

(End of Case 2.)

Let $\mathcal{M}$ denote the set of multiterminal cuts of G with terminals α, β and γ and let T be the number of multiterminal cuts in $\mathcal{M}$ with size B. We would like to show how to estimate T using an approximation for the number of homomorphisms from (J, S) to H. Towards this end, define Z as follows.

$$Z = TZ^* + \sum_{E' \in \mathcal{M}: |E'| > B} \; \sum_{\psi \in \Psi(E')} 2^{sr(|X(\psi)|+|Y(\psi)|+|Z(\psi)|)}.$$

The proof is in two parts.

Part 1: We show that $Z/Z^* \in [T, T + 1/4]$.

Since $|X(\psi)| + |Y(\psi)| + |Z(\psi)| \le |E(G)| - |E'|$, we have

$$Z \le TZ^* + \sum_{E' \in \mathcal{M}: |E'| > B} \; \sum_{\psi \in \Psi(E')} 2^{sr(|E(G)|-|E'|)}.$$

Using $|\Psi(E')| = {d_w}^{\kappa(E')-3}$ (from (1)) and the definition of Z^* (just before (2)), we obtain

$$Z \le TZ^* + \sum_{E' \in \mathcal{M}: |E'| > B} {d_w}^{\kappa(E')-3} \frac{Z^*}{2^{sr(|E'|-B)}}.$$

Then, in the following expression, the first inequality follows from the definition of Z and the second inequality follows from the fact that there are at most $2^{|E(G)|}$ multiterminal cuts and from the bounds ${d_w}^{\kappa(E')-3} \le q^n$, $|E'| - B \ge 1$ and $r \ge 1$. The third inequality follows from the choice of s.

$$T \le \frac{Z}{Z^*} \le T + \frac{2^{|E(G)|} q^n}{2^s} \le T + 1/4.$$

We have verified that $Z/Z^* \in [T, T + 1/4]$.

Part 2: We show that we can obtain a close approximation to Z/Z^* using an oracle for approximating #Ret(H).

Recall that $N\big((J,\mathrm{S}) \to H\big)$ is the number of homomorphisms from (J,S) to H and note that

$$N\big((J,\mathrm{S}) \to H\big) = \sum_{E' \in \mathcal{M}: |E'|=B} Z_{E'} + \sum_{E' \in \mathcal{M}: |E'|>B} Z_{E'}.$$

Using Inequalities (2) and (3), and the fact that G, α, β, γ has T multiterminal cuts of size B, we have

$$q^{-\delta'} Z \le N\big((J,\mathrm{S}) \to H\big) \le q^{\delta'} Z.$$

Let $\hat{Q}$ be a solution returned by the #Ret(H) oracle when called with input $\big((J,\mathrm{S}), \varepsilon/42\big)$. Then

$$e^{-\varepsilon/42} q^{-\delta'} Z \le e^{-\varepsilon/42} N\big((J,\mathrm{S}) \to H\big) \le \hat{Q} \le e^{\varepsilon/42} N\big((J,\mathrm{S}) \to H\big) \le e^{\varepsilon/42} q^{\delta'} Z.$$

The choice of $\delta' = \log_q e^{\varepsilon/42}$ yields $e^{-\varepsilon/21} \frac{Z}{Z^*} \le \frac{\hat{Q}}{Z^*} \le e^{\varepsilon/21} \frac{Z}{Z^*}$. Note that Z^* is easy to compute. The fact that this precision in the approximation of Z suffices to obtain the required accuracy of the output $\hat{Q}/Z^*$ as an approximation of T is derived in [5, Proof of Theorem 3]. □

We can now give a classification of #Ret(H) for irreflexive square-free graphs.

Theorem 2.3. *Suppose that H is an irreflexive square-free graph.*

(i) If every connected component of H is a star, then #Ret(H) is in FP.
(ii) Otherwise, if every connected component of H is a caterpillar, then #Ret(H) approximation-equivalent to #BIS.
(iii) Otherwise, #Ret(H) approximation-equivalent to #SAT.

Proof. We first give the classification assuming that H is a connected graph. Then we use Remark 1.15 to recover the full classification.

Suppose that H is a connected irreflexive square-free graph. We have #Hom(H) $\le_{\mathrm{AP}}$ #Ret(H) and #Ret(H) $\le_{\mathrm{AP}}$ #LHom(H) by Observation 1.2. Therefore, #Ret(H) inherits hardness results from #Hom(H) hardness results and it inherits easiness results from #LHom(H) easiness results. Thus, since a star is a complete bipartite graph, item (i) follows from Theorem 1.8. Since a square-free graph that is not a star cannot be a complete bipartite graph, the #BIS-hardness part of item (ii) follows from Theorem 1.9. It is known that a caterpillar is a bipartite permutation graph [34] (see also [16, Appendix A]), so the #BIS-easiness part of item (ii) follows again from Theorem 1.8. Theorem 1.8 also implies that #Ret(H) is always #SAT-easy, giving the easiness result in item (iii).

It remains to show the hardness result in item (iii). If H is not a caterpillar, then it contains either a cycle or an induced J_3 [24, Theorem 1].

Case 1: H contains an induced J_3. The fact that #SAT $\le_{\mathrm{AP}}$ #Ret(H) follows from Lemma 2.2. **End of Case 1.**

Case 2: H contains a cycle.

- Suppose that H contains a cycle of odd length. Then H is not bipartite and even the problem of deciding whether there exists a homomorphism to H is NP-complete due to Hell and Nešetřil [27]. This homomorphism decision problem reduces to the retraction decision problem Ret(H) [2] and therefore Ret(H) is NP-hard as well. Then, NP-hardness of Ret(H) implies #SAT-hardness of the corresponding approximate counting problem #Ret(H) by [5, Theorem 1].

- Suppose that H contains exactly one cycle of even length. Then H is a pseudotree and, as H is square-free, the cycle has length at least 6. Therefore $\textsc{Ret}(H)$ is NP-complete by Theorem 1.11. Then, as before, it follows that $\#\textsc{Ret}(H)$ is #SAT-hard under AP-reductions by [5, Theorem 1].
- Suppose that H contains at least 2 cycles and all cycles in H have even length. We will show that H contains an induced J_3 and therefore is covered by Case 1. Let C be a shortest cycle in H. As there are at least two cycles in H and H is connected, there exists a path $P_1 = w, z_0, z_1$ such that w is in C and z_0 is not in C. As C has length at least 6, there exists a path P_2 in C of the form x_0, x_1, w, y_0, y_1. As z_0 is not in C it does not coincide with any of the vertices of P_2. Further, as C is a shortest cycle, z_1 cannot coincide with any of the vertices of P_2. Therefore the vertices of P_1 and P_2 form a graph J_3 as shown in Figure 4. This subgraph J_3 is induced as H does not contain any cycles of length less than 6.

End of Case 2.

The theorem now follows easily by Remark 1.15. If every connected component is easy, so is H. If any connected component is hard, so is H. □

2.2 Graphs with Loops

In this section we consider graphs that are not irreflexive.

2.2.1 #BIS-Easiness Results for Graphs with Loops. The point of this section is to prove the following lemma.

Lemma 2.4. *Let H be a partially bristled reflexive path with at least 3 vertices. Then $\#\textsc{Ret}(H) \equiv_{\mathrm{AP}}$ #BIS.*

This lemma builds on Kelk [33, Appendix A.8], who shows that $\#\textsc{Hom}(H) \equiv_{\mathrm{AP}}$ #BIS for partially bristled reflexive paths H. Thus, our work in this section is generalising Kelk's work from homomorphism-counting to retraction-counting. For us, the main interest is actually that we manage to classify *all* graphs of girth at least 5, rather than that we show that these particular graphs are #BIS-equivalent. Nevertheless, partially bristled reflexive paths allow us to explore some interesting ideas, providing a convenient setting for generalising useful techniques.

In particular, in order to reduce $\#\textsc{Ret}(H)$ to #BIS, we generalise a technique that was introduced by Dyer et al. [5, Lemma 8] in order to reduce homomorphism-counting problems to #BIS. Although the graphs H that we consider in this work are undirected, we show that the technique also applies to directed graphs. We expect this to be useful for future work.[2] A homomorphism from a digraph G to a digraph H is simply a function $h\colon V(G) \to V(H)$ such that, for all $(u, v) \in E(G)$, the image $(h(u), h(v))$ is in $E(H)$. A homomorphism from $(G, \mathbf{S})$ to H must satisfy $h(v) \in S_v$, as in the undirected case. As for undirected graphs, we use $N\big((G, \mathbf{S}) \to H\big)$ to denote the number of homomorphisms from $(G, \mathbf{S})$ to H. Thus, we consider the following directed retraction problem.

Name: $\#\textsc{Dir-Ret}(H)$.
Input: An irreflexive digraph G and a collection of lists $\mathbf{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$ such that, for all $v \in V(G)$, $|S_v| \in \{1, |V(H)|\}$.
Output: $N\big((G, \mathbf{S}) \to H\big)$.

The main method used in the literature to prove #BIS-easiness of approximate homomorphism-counting problems is to reduce them to the problem of counting the downsets of a partial order, which is known to be #BIS-equivalent [5]. In order to obtain more general results, we formalise

[2]The technique also applies if the input G to $\#\textsc{Ret}(H)$ is allowed to have loops. This is the main observation needed to show that Theorem 1.1 extends to the setting where G might have loops.

the technique introduced in the proof of [5, Lemma 8] and expanded by Kelk [33], and use it in the context of the constraint satisfaction framework. Let $\mathcal{L}$ be a set of Boolean relations (called a constraint language). The counting constraint satisfaction problem (CSP) with parameter $\mathcal{L}$ is defined as follows.

Name: #CSP($\mathcal{L}$).

Input: A set of variables X and a set of constraints C, where each constraint applies a relation from $\mathcal{L}$ to a list of variables from X.

Output: The number of assignments $\sigma\colon X \to \{0,1\}$ that satisfy all constraints in C.

The constraint language that we will use consists of the two unary Boolean relations $\delta_0 = \{(0)\}$ and $\delta_1 = \{(1)\}$ and the arity-two Boolean relation $\mathrm{Imp} = \{(0,0),(0,1),(1,1)\}$. Note that the constraint $\delta_0(x)$ forces a satisfying assignment to assign the value 0 to the variable x and the constraint $\delta_1(x)$ forces a satisfying assignment to assign the value 1 to x. The constraint $\mathrm{Imp}(x,y)$ ensures that, in any satisfying assignment σ, we have $\sigma(x) \implies \sigma(y)$ (that is, if $\sigma(x) = 1$ then $\sigma(y) = 1$). It is known that the counting constraint satisfaction problem is #BIS-equivalent when the constraint language contains (exactly) these three relations.

Lemma 2.5. *[7, Theorem 3] #CSP*($\{\mathrm{Imp},\delta_0,\delta_1\}$) $\equiv_{\mathrm{AP}}$ *#BIS.*

We now formalise the downsets reduction technique from [5, Lemma 8] and state it as a technique for reducing homomorphism-counting problems to #CSP($\{\mathrm{Imp},\delta_0,\delta_1\}$). We generalise the original technique in two ways. First, we allow size-1 and size-$|V(H)|$ lists in the input, so we obtain #BIS-easiness results for #Ret(H) and not merely for #Hom(H). Second, even though the main focus of this work is on undirected graphs, we set up the machinery to enable (stronger) #BIS-easiness results for the directed problem #Dir-Ret(H).

The main idea is as follows. Given *any* instances I_v, I_e, I_f and I_b of #CSP($\{\mathrm{Imp}\}$) on a variable set X we will define (Definition 2.6) an undirected graph $H_{I_\mathrm{v},I_\mathrm{e}}$ and (Definition 2.7) a digraph $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$. Then Lemma 2.8 will show that the problems #Ret($H_{I_\mathrm{v},I_\mathrm{e}}$) and #Dir-Ret($H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$) both reduce to the #BIS-easy problem #CSP($\{\mathrm{Imp},\delta_0,\delta_1\}$). Finally, to prove the #BIS-easiness of #Ret(H) when H is a partially bristled reflexive path (in order to achieve our goal of proving Lemma 2.4), we have to show, given a partially bristled reflexive path H, how to set up the corresponding instances I_v and I_e of #CSP($\{\mathrm{Imp}\}$) so that $H_{I_\mathrm{v},I_\mathrm{e}} = H$.

Before defining the graph $H_{I_\mathrm{v},I_\mathrm{e}}$ and the digraph $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$, it helps to explain the notation. The subscript "v" stands for "vertex" and the #CSP($\{\mathrm{Imp}\}$) instance I_v is used to define the vertices of the graph $H_{I_\mathrm{v},I_\mathrm{e}}$ and the vertices of the digraph $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$. The subscript "e" stands for "edge" and the CSP instance I_e is used to define the edges of $H_{I_\mathrm{v},I_\mathrm{e}}$. The instance I_f gives the "forward" constraints for each directed edge of $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$ and the instance I_b gives the corresponding "backward" constraints. We will use C_v, C_e, C_f, and C_b to denote the constraint sets of the instances I_v, I_e, I_f, and I_b, respectively.

Definition 2.6. Let $I_\mathrm{v} = (X, C_\mathrm{v})$ and $I_\mathrm{e} = (X, C_\mathrm{e})$ be instances of #CSP($\{\mathrm{Imp}\}$). We define the undirected graph $H_{I_\mathrm{v},I_\mathrm{e}}$ as follows. The vertices of $H_{I_\mathrm{v},I_\mathrm{e}}$ are the satisfying assignments of I_v. Given any assignments σ and σ' in $V(H_{I_\mathrm{v},I_\mathrm{e}})$, there is an edge $\{\sigma,\sigma'\}$ in $H_{I_\mathrm{v},I_\mathrm{e}}$ if and only if the following holds: For every constraint $\mathrm{Imp}(x,y)$ in I_e, we have $\sigma(x) \Rightarrow \sigma'(y)$ and $\sigma'(x) \Rightarrow \sigma(y)$.

The definition of the digraph $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$ is similar.

Definition 2.7. Let $I_\mathrm{v} = (X, C_\mathrm{v})$, $I_\mathrm{f} = (X, C_\mathrm{f})$ and $I_\mathrm{b} = (X, C_\mathrm{b})$ be instances of #CSP($\{\mathrm{Imp}\}$). We define the directed graph $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$ as follows. The vertices of $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$ are the satisfying assignments of I_v. Given any assignments σ and σ' in $V(H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}})$, there is a (directed) edge (σ,σ') in $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$ if and only if the following holds:

- For every constraint $\mathrm{Imp}(x, y)$ in I_f, we have $\sigma(x) \Rightarrow \sigma'(y)$, and
- for every constraint $\mathrm{Imp}(x, y)$ in I_b, we have $\sigma'(x) \Rightarrow \sigma(y)$.

Lemma 2.8. *Let $I_\mathrm{v} = (X, C_\mathrm{v})$, $I_\mathrm{e} = (X, C_\mathrm{e})$, $I_\mathrm{f} = (X, C_\mathrm{f})$ and $I_\mathrm{b} = (X, C_\mathrm{b})$ be instances of #CSP($\{\mathrm{Imp}\}$). Then #Ret($H_{I_\mathrm{v},I_\mathrm{e}}$) $\leq_\mathrm{AP}$ #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$) and #Dir-Ret($H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$) $\leq_\mathrm{AP}$ #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$).*

Proof. **Undirected case:** We first show the reduction from #Ret($H_{I_\mathrm{v},I_\mathrm{e}}$) to #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$) and extend this to the directed result afterwards. The reductions we show are parsimonious. From an instance $(G, \mathbf{S})$ of #Ret($H_{I_\mathrm{v},I_\mathrm{e}}$) we create an instance I of #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$) as follows. The set of variables of I is $V(G) \times X$ and the set of constraints C of I is constructed as follows.

(1) For each $v \in V(G)$ and each constraint $\mathrm{Imp}(x, y) \in I_\mathrm{v}$, we add the constraint $\mathrm{Imp}((v, x), (v, y))$ to C.
(2) For each edge $\{u, v\} \in E(G)$ and each constraint $\mathrm{Imp}(x, y) \in I_\mathrm{e}$, we add the constraints $\mathrm{Imp}((u, x), (v, y))$ and $\mathrm{Imp}((v, x), (u, y))$ to C.
(3) For each $v \in V(G)$ with $|S_v| = 1$ let τ be the (only) element of S_v. If $\tau(x) = 0$ then add the constraint $\delta_0((v, x))$ to C. Otherwise, add the constraint $\delta_1((v, x))$ to C.

To complete the reduction from #Ret($H_{I_\mathrm{v},I_\mathrm{e}}$) to #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$), we will show that there is a bijection between homomorphisms from $(G, \mathbf{S})$ to $H_{I_\mathrm{v},I_\mathrm{e}}$ and satisfying assignments of I. This bijection ensures that the number of satisfying assignments of I is equal to $N\big((G, \mathbf{S}) \to H_{I_\mathrm{v},I_\mathrm{e}}\big)$. Hence the approximation to $N\big((G, \mathbf{S}) \to H_{I_\mathrm{v},I_\mathrm{e}}\big)$ can be achieved using a single oracle call to #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$) with the desired accuracy ε.

To establish the bijection, we present an (invertible) map from satisfying assignments of I to homomorphisms from $(G, \mathbf{S})$ to $H_{I_\mathrm{v},I_\mathrm{e}}$. The map is constructed as follows. Let σ be any satisfying assignment of I.

- For every vertex $v \in V(G)$, define a function $\sigma_v \colon X \to \{0, 1\}$ as follows. For all $x \in X$, let $\sigma_v(x) = \sigma((v, x))$. The constraints added to C in item (1) ensure that, since σ is a satisfying assignment of I, the assignment σ_v is a satisfying assignment of I_v. Thus, σ_v is a vertex of $H_{I_\mathrm{v},I_\mathrm{e}}$.
- Next, we will argue that the function from $V(G)$ to $V(H_{I_\mathrm{v},I_\mathrm{e}})$ that maps every vertex $v \in V(G)$ to σ_v is a homomorphism from $(G, \mathbf{S})$ to $H_{I_\mathrm{v},I_\mathrm{e}}$.
  - Consider an edge $\{u, v\}$ of G. We must show that $\{\sigma_u, \sigma_v\}$ is an edge of $H_{I_\mathrm{v},I_\mathrm{e}}$. Using Definition 2.6, this is equivalent to showing that, for every constraint $\mathrm{Imp}(x, y)$ in I_e, we have $\sigma_u(x) \Rightarrow \sigma_v(y)$ and $\sigma_v(x) \Rightarrow \sigma_u(y)$. Using the construction of σ_u and σ_v, this is equivalent to showing that, for every constraint $\mathrm{Imp}(x, y)$ in I_e, we have $\sigma(u, x) \Rightarrow \sigma(v, y)$ and $\sigma(v, x) \Rightarrow \sigma(u, y)$. This is ensured by the fact that σ is a satisfying assignment of I, so it satisfies the constraints added in item (2).
  - Consider a vertex $v \in V(G)$ with $S_v = \{\tau\}$. We must show that $\sigma_v = \tau$. This is ensured by the constraints added in item (3).

Starting from the satisfying assignment σ of I, we produced a homomorphism from $(G, \mathbf{S})$ to $H_{I_\mathrm{v},I_\mathrm{e}}$, namely the homomorphism that maps every vertex $v \in V(G)$ to σ_v. To finish the proof, we need only note that this construction is invertible — given any homomorphism from $(G, \mathbf{S})$ to $H_{I_\mathrm{v},I_\mathrm{e}}$ we can let σ_v denote the image of v under this homomorphism. Given the collection $\{\sigma_v \mid v \in V(G)\}$, we construct an assignment σ from $V(G) \times X$ to $\{0, 1\}$ by inverting the above construction: For every $v \in V(G)$ and $x \in X$, let $\sigma((v, x)) = \sigma_v(x)$. We must then check that σ is satisfying.

- For each $v \in V(G)$, the assignment σ satisfies the relevant constraints added in item (1) because σ_v is a vertex of $H_{I_\mathrm{v},I_\mathrm{e}}$, hence a satisfying assignment of I_v.

- For each $\{u, v\} \in E(G)$ and each pair of constraints $\mathrm{Imp}((u, x), (v, y))$ and $\mathrm{Imp}((v, x), (u, y))$ added to C in item (2), σ satisfies the constraints because $\{\sigma_u, \sigma_v\}$ is an edge of $H_{I_\mathrm{v},I_\mathrm{e}}$ (so $\sigma_u(x) \implies \sigma_v(y)$ and $\sigma_v(x) \implies \sigma_u(y)$).
- Finally, for any $s \in \{0, 1\}$, consider a constraint $\delta_s((v, x))$ introduced in item (3). The procedure in item (3) ensures that, for some τ with $S_v = \{\tau\}$, we have $\tau(x) = s$. Our homomorphism has $\sigma_v = \tau$. Thus, the constraint $\sigma((v, x)) = s$ is satisfied by σ.

Directed Case: The reduction from #Dir-Ret($H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$) to #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$) is similar to the one given in the undirected case. Starting with an instance (G, S) of #Dir-Ret($H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$) we create an instance I of #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$) as follows. The set of variables of I is $V(G)\times X$, as in the undirected reduction. The set of constraints C of I is constructed in the same way as in the undirected reduction, except that item (2) is replaced with the following.

(2)' For each (directed) edge $(u, v) \in E(G)$, we add the following constraints to C. For each constraint $\mathrm{Imp}(x, y) \in I_\mathrm{f}$, we add the constraint $\mathrm{Imp}((u, x), (v, y))$ to C. For each constraint $\mathrm{Imp}(x, y) \in I_\mathrm{b}$, we add the constraint $\mathrm{Imp}((v, x), (u, y))$ to C.

As in the undirected case, we complete the proof by establishing a bijection from satisfying assignments of I to homomorphisms from (G, S) to $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$. Let σ be any satisfying assignment of I. The construction of σ_v from σ is the same as in the undirected case. Only one difference arises in the verification that the function from $V(G)$ to $V(H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}})$ that maps every vertex $v \in V(G)$ to σ_v is a homomorphism from (G, S) to $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$. Consider any directed edge (u, v) of G. We must show that (σ_u, σ_v) is an edge of $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$. Using Definition 2.7 and the construction of σ_u and σ_v, this is equivalent to showing

- For every constraint $\mathrm{Imp}(x, y)$ in I_f, we have $\sigma(u, x) \Rightarrow \sigma(v, y)$, and
- for every constraint $\mathrm{Imp}(x, y)$ in I_b, we have $\sigma(v, x) \Rightarrow \sigma(u, y)$.

This is ensured by the constraints added in item (2)'.

As in the undirected case, we next show that we have a bijection by starting with a homomorphism from (G, S) to $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$ and letting σ_v denote the image of v under this homomorphism. Given the collection $\{\sigma_v \mid v \in V(G)\}$ we construct an assignment σ from $V(G) \times X$ to $\{0, 1\}$ exactly as in the undirected case. We must check that σ is a satisfying assignment of I. This is the same as the undirected case except when checking that σ satisfies the constraints added in item (2)'. For $(u, v) \in E(G)$ and a constraint $\mathrm{Imp}((u, x), (v, y))$ added to C because $\mathrm{Imp}(x, y) \in I_\mathrm{f}$, note that, since (σ_u, σ_v) is an edge of $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$, by Definition 2.7, we have $\sigma_u(x) \implies \sigma_v(x)$, so the constraint is satisfied. Similarly, for a constraint $\mathrm{Imp}((v, x), (u, y))$ added to C because $\mathrm{Imp}(x, y) \in I_\mathrm{b}$ we again have $\sigma_v(x) \implies \sigma_u(y)$, so the constraint is satisfied.

So we have a bijection from satisfying assignments of I to homomorphisms from (G, S) to $H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$ and the reduction from #Dir-Ret($H_{I_\mathrm{v},I_\mathrm{f},I_\mathrm{b}}$) to #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$) follows. □

Although, to be general, we have presented undirected and directed reductions in Lemma 2.8, our goal in Lemma 2.4 is to prove #BIS-easiness of #Ret(H) for a partially bristled reflexive path, which is an undirected graph. So we will use the undirected reduction from Lemma 2.8 for this. The rough idea will be to take a partially bristled reflexive path H and show how to set up the corresponding instances I_v and I_e of #CSP($\{\mathrm{Imp}\}$) so that $H_{I_\mathrm{v},I_\mathrm{e}} = H$. Then Lemma 2.8 shows that #Ret(H) reduces to #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$), so #Ret($H$) is #BIS-easy. Unfortunately, we can't precisely achieve this goal, but we can set up the corresponding instances I_v and I_e so that $H_{I_\mathrm{v},I_\mathrm{e}}$ is equal to H, together with some additional small connected components, which turn out not to matter. The fact that these small connected components don't cause trouble was first observed by Kelk [33] in the context of counting homomorphisms. In Lemma 2.10 we show that this is also true when counting retractions. Lemma 2.9 states the well-known fact that subtracting polynomial-size entities does not spoil an AP-reduction, which is, for instance, pointed out in [33, Lemma 6.6]. For the sake of completeness we give a short proof.

Lemma 2.9. *Let H and H' be graphs. For any graph G, let $f(G) = N(G \to H) - N(G \to H')$. If $f(G)$ is non-negative and bounded from above by a polynomial in $|V(G)|$, and can be computed in polynomial time, then $\#\textsc{Hom}(H') \le_{\mathrm{AP}} \#\textsc{Hom}(H)$.*

Proof. Let G be an instance of $\#\textsc{Hom}(H')$ and let $\varepsilon \in (0,1)$ be the desired precision. To shorten notation, let $N = N(G \to H)$. From the definition of f in the statement of the lemma, $N(G \to H') = N - f(G)$. First, the algorithm computes $k = f(G)$ in polynomial time. If $k = 0$, then $N(G \to H') = N(G \to H)$, and the algorithm simply returns the result of a $\#\textsc{Hom}(H)$ oracle call with precision ε.

Suppose instead that $k \ge 1$. In this case, the algorithm makes a $\#\textsc{Hom}(H)$ oracle call with input G and precision $\delta \le \frac{\varepsilon}{16k}$. Let R be the integer solution returned by this oracle call (note that R is an approximation to N satisfying $e^{-\delta}N \le R \le e^{\delta}N$). The algorithm returns $R - k$. We show that this output approximates $N(G \to H')$ with the desired precision.

If $N(G \to H') = 0$ then $N = k$ and $e^{-\delta}k \le R \le e^{\delta}k$. By Observation 1.13 and the facts that $\varepsilon < 1$ and $k \ge 1$ this implies $R \in (k - 1/4, k + 1/4)$ and since R is integer this gives $R = k$. Thus, in this case the algorithm returns 0, which is the exact solution.

Suppose instead that $N(G \to H') \ge 1$. In this case, $N \ge k + 1$ and by Observation 1.13 we have

$$R - k \le e^{\delta}N - k \le (1 + 2\delta)N - k = (1 + 2\delta)(N - k) + 2k\delta.$$

Since $N \ge k + 1$ and $2\delta \le \varepsilon/8$ we have $2k\delta \le \varepsilon/8 \le \varepsilon/8 \cdot (N - k)$ and consequently

$$(1 + 2\delta)(N - k) + 2k\delta \le (1 + \varepsilon/4)(N - k).$$

Analogously, we obtain $R - k \ge (1 - \delta)(N - k) - k\delta \ge (1 - \varepsilon/8)(N - k)$. Finally, by Observation 1.13, this implies $e^{-\varepsilon}(N - k) \le R - k \le e^{\varepsilon}(N - k)$ and thus returning $R - k$ has the desired precision. □

Lemma 2.10. *Let H' be a graph and let H be the graph consisting of a connected component that is isomorphic to H' together with some additional connected components $C_1, \dots, C_k$. Suppose that, for each $i \in [k]$, C_i is one of the following graphs: a singleton vertex, with or without a loop, or an unlooped edge. Then $\#\textsc{Ret}(H') \le_{\mathrm{AP}} \#\textsc{Ret}(H)$.*

Proof. Recall the definition of $\#\textsc{Ret}(H)^{\mathrm{conn}}$ from Section 1.5. To prove the lemma we show

$$\#\textsc{Ret}(H') \le_{\mathrm{AP}} \#\textsc{Ret}(H')^{\mathrm{conn}} \le_{\mathrm{AP}} \#\textsc{Ret}(H)^{\mathrm{conn}} \le_{\mathrm{AP}} \#\textsc{Ret}(H). \tag{4}$$

The first and the trivial third reduction follow from Observation 1.14. It remains to show that $\#\textsc{Ret}(H')^{\mathrm{conn}} \le_{\mathrm{AP}} \#\textsc{Ret}(H)^{\mathrm{conn}}$. Let $(G, \mathbf{S}')$ be an input to $\#\textsc{Ret}(H')^{\mathrm{conn}}$ and let $\varepsilon \in (0,1)$ be the desired precision. From the problem definition, G is connected. Now define the lists S_v for $v \in V(G)$ as follows. If $S'_v = V(H')$ then let $S_v = V(H)$. Otherwise, let $S_v = S'_v$. Let $\mathbf{S} = \{S_v \mid v \in V(G)\}$

First, the algorithm tests whether there is a list $S'_v \in \mathbf{S}'$ with $|S'_v| = 1$. If there is such a list, then there is a particular component of H' with the property that every homomorphism from G to H' maps all vertices of G to this component, and every homomorphism from G to H maps all vertices of G to this component. Thus, $N((G, \mathbf{S}') \to H') = N((G, \mathbf{S}) \to H)$. So a single oracle call with precision ε gives the sought-for approximation.

If there is no list $S'_v \in \mathbf{S}'$ with $|S'_v| = 1$ then every list S'_v is equal to $V(H')$ and every list S_v is equal to $V(H)$. Thus, $N((G, \mathbf{S}') \to H') = N(G \to H')$ and $N((G, \mathbf{S}) \to H) = N(G \to H)$. As G is connected we also have $N(G \to H) = N(G \to H') + \sum_{i=1}^{k} N(G \to C_i)$. As $C_1, \dots C_k$ are either singleton vertices or unlooped edges, the algorithm can compute $\sum_{i=1}^{k} N(G \to C_i)$ efficiently. Also, for each $i \in [k]$, $N(G \to C_i) \le 2$. Setting $f(G) = \sum_{i=1}^{k} N(G \to C_i)$ in Lemma 2.9 gives the sought-for AP-reduction. □

As noted at the beginning of this section, approximately counting homomorphisms to partially bristled reflexive paths is shown to be #BIS-easy in [33, Appendix A.8]. Using the same construction and our Lemma 2.8 we can now prove Lemma 2.4, which is the generalisation for counting retractions. We restate the lemma and recast the construction in our setting for convenience.

LEMMA 2.4. *Let H be a partially bristled reflexive path with at least 3 vertices. Then #RET$(H) \equiv_{\text{AP}}$ #BIS.*

PROOF. The #BIS-hardness part of the statement is inherited from #HOM(H) using Theorem 1.9 and the reduction #HOM$(H) \leq_{\text{AP}}$ #RET(H) from Observation 1.2. We now show #BIS-easiness.

Matching the notation from Definition 1.12, the partially bristled reflexive path H can be described as follows. There exists a positive integer Q and a be a subset S of $[Q]$ such that $V(H) = \{c_0, \dots, c_{Q+1}\} \cup \bigcup_{i\in S}\{g_i\}$ and $E(H) = \bigcup_{i=0}^{Q}\{c_i, c_{i+1}\} \cup \bigcup_{i=0}^{Q+1}\{c_i, c_i\} \cup \bigcup_{i\in S}\{c_i, g_i\}$. Note that S can be empty.

Let $X = \{x_0, \dots, x_Q\}$. Define the instances $I_\text{v} = (X, C_\text{v})$ and $I_\text{e} = (X, C_\text{e})$ of #CSP({Imp}) as follows.

- For each $i \in [Q] \setminus S$, we add a constraint $\text{Imp}(x_i, x_{i-1})$ to C_v.
- For each pair (i, j) satisfying $0 \leq i < j \leq Q$, we add a constraint $\text{Imp}(x_j, x_i)$ to C_e.

We claim that the graph $H_{I_\text{v},I_\text{e}}$, as defined in Definition 2.6, has a connected component that is isomorphic to H and that all other connected components of $H_{I_\text{v},I_\text{e}}$ are singleton vertices (without loops). Given the claim, the reduction from #RET(H) to #BIS follows from Lemmas 2.10, 2.8 and 2.5 (applied in that order).

We conclude the proof by showing the claim. For each $i \in \{0, \dots, Q+1\}$ let $\sigma_i \colon X \longrightarrow \{0, 1\}$ be the following assignment of Boolean values to variables in X.

$$\sigma_i(x_j) = \begin{cases} 1, & \text{if } j < i \\ 0, & \text{otherwise.} \end{cases}$$

Note that σ_0 maps all arguments to 0 and σ_{Q+1} maps all arguments to 1.

The indices of $\sigma_0, \dots, \sigma_{Q+1}$ are chosen this way to match the indices of the vertices c_0 to c_{Q+1} of the graph H. Note that $\sigma_0, \dots, \sigma_{Q+1}$ are satisfying assignments of I_v and, therefore, they are vertices of $H_{I_\text{v},I_\text{e}}$. By the definition of I_e, these vertices are looped in $H_{I_\text{v},I_\text{e}}$. Also, for all $i \in [Q]$, we have $\{\sigma_i, \sigma_{i+1}\} \in E(H_{I_\text{v},I_\text{e}})$. Hence, the vertices $\sigma_0, \dots, \sigma_{Q+1}$ form a reflexive path in $H_{I_\text{v},I_\text{e}}$.

Now for each $i \in [Q]$ let $\sigma'_i \colon X \longrightarrow \{0, 1\}$ be the following assignment of Boolean values to variables in X.

$$\sigma'_i(x_j) = \begin{cases} 1, & \text{if } j \leq i \text{ and } j \neq i-1 \\ 0, & \text{otherwise.} \end{cases}$$

For every $i \in [Q]$, we have $\sigma'_i(x_{i-1}) = 0$ and $\sigma'(x_i) = 1$, so σ'_i is not equal to any $\sigma_{i'}$.

Consider a vertex c_i of H with $i \in [Q]$.

- If $i \in S$: In this case, σ'_i is a satisfying assignment of I_v. By the definition of I_e, σ'_i has degree 1 and is adjacent to σ_i in $H_{I_\text{v},I_\text{e}}$. Thus, the vertex σ'_i of $H_{I_\text{v},I_\text{e}}$ corresponds to the vertex g_i of H.
- If $i \notin S$: In this case, as $i \geq 1$, the constraint $\text{Imp}(x_i, x_{i-1})$ in I_v ensures that σ'_i is not a satisfying assignment of I_v and, therefore, σ'_i is not a vertex of $H_{I_\text{v},I_\text{e}}$.

We will next show that the edges that we have already described constitute all of the edges of $H_{I_\text{v},I_\text{e}}$. This means that the rest of the vertices of $H_{I_\text{v},I_\text{e}}$ have degree 0, so we are finished.

To this end, let σ be any function from X to $\{0, 1\}$. From the definition of I_e, we obtain the following necessary condition for σ to have a neighbour in $H_{I_\text{v},I_\text{e}}$: Let $i \in \{0, \dots, Q\}$ be the largest

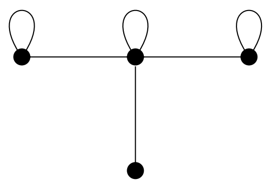
Fig. 7. The graph 2-Wrench.

index for which $\sigma(x_i) = 1$. If ψ is a neighbour of σ then, for all $j \leq i-1$, $\psi(x_j) = 1$ and hence, for all $j \leq i-2$, $\sigma(x_j) = 1$. Thus, for σ to have a neighbour in H_{I_v,I_e} it has to be of the form σ_i or σ_i'. □

Remark 2.11. One interesting feature of Lemma 2.4 is that it shows that there are graphs H for which #Ret(H) is #BIS-equivalent, whereas #LHom(H) is #SAT-hard. Thus, subject to the complexity assumption that #BIS is not #SAT-equivalent, there is a graph H for which the complexity of #Ret(H) differs from that of #LHom(H). The smallest example from the class of partially bristled reflexive paths for which this separation holds is the so-called 2-Wrench, depicted in Figure 7. The fact that #SAT $\leq_{\mathrm{AP}}$ #LHom(2-Wrench) follows from Theorem 1.8.

2.2.2 #SAT-Hardness Results for Graphs with Loops. The goal of this section is to prove the hardness results given in Lemmas 2.14, 2.15 and 2.30. In order to show #SAT-hardness results we will prove that certain neighbourhood structures induce hardness. To this end consider the following easy and well-known observation proved here for completeness.

Observation 2.12. *Let H be a graph and let u be a vertex of H. Then #Hom($H[\Gamma(u)]$) $\leq_{\mathrm{AP}}$ #Ret(H).*

Proof. Let G be an input to #Hom($H[\Gamma(u)]$) and let $v_1, \ldots, v_n$ be the vertices of G. Let w be a vertex distinct from the vertices in G. Then we construct the graph G' with vertices $V(G') = V(G) \cup \{w\}$ and edges $E(G') = E(G) \cup \{\{w, v_i\} \mid i \in [n]\}$. We set $S_w = \{u\}$ and $S_v = V(H)$ for all remaining vertices of G'. Let $\mathbf{S} = \{S_v \mid v \in V(G')\}$. Then $N\big(G \to H[\Gamma(u)]\big) = N\big((G', \mathbf{S}) \to H\big)$. □

First we combine some known results to show hardness that is derived from the analysis of distance-1 neighbourhoods (Lemmas 2.14 and 2.15). Then we show hardness results derived from the analysis of distance-2 neighbourhoods in the more difficult Lemma 2.30, which is the main result of this section.

For Lemmas 2.14 and 2.15 we use gadgets based on complete bipartite graphs where two states dominate (see, e.g., [5, Lemma 25], [20, Section 5] and [33, Lemma 5.1]). We use the version of Kelk [33]. Let $F(H) = \{u \in V(H) \mid \Gamma(u) = V(H)\}$. For a set of vertices S recall the set of common neighbours $\Gamma(S)$ from Section 1.5.

Lemma 2.13 ([33, Lemma 5.1]). *Let H be a graph with $\emptyset \subsetneq F(H) \subsetneq V(H)$. Suppose that, for every pair (S,T) with $\emptyset \subseteq S, T \subseteq V(H)$ satisfying $S \subseteq \Gamma(T)$ and $T \subseteq \Gamma(S)$, at least one of the following holds:*
(1) $S = F(H)$.
(2) $T = F(H)$.
(3) $|S| \cdot |T| < |F(H)| \cdot |V(H)|$.
Then #SAT $\leq_{\mathrm{AP}}$ #Hom(H).

Lemma 2.13 is not difficult to prove. A homomorphism from a complete bipartite graph to H will typically map one side to $F(H)$ and the other to $V(H)$. So it is easy to reduce from counting independent sets.

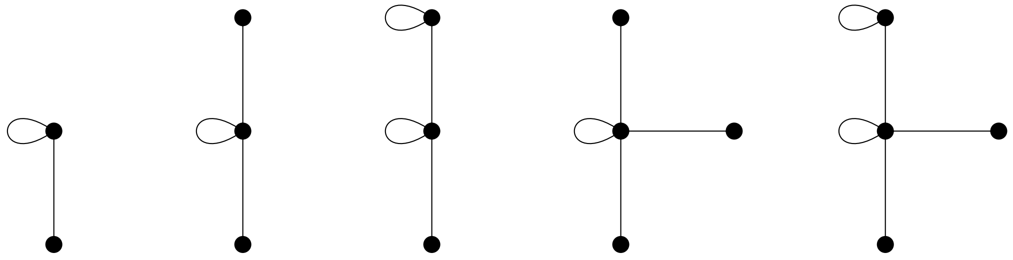
Fig. 8. Possible graphs H_b with at most 4 vertices.

Let WR_q be a reflexive star with q leaves. (The name is not relevant here but it comes from the Widom-Rowlinson model [51] from statistical physics.) The non-leaf vertex of WR_q is called its centre.

Lemma 2.14. *Let H be a graph that has a looped vertex b such that $H[\Gamma(b)]$ is isomorphic to WR_q for some $q \geq 3$. Then #SAT $\leq_{\mathrm{AP}}$ #Ret(H).*

Proof. The problem #Hom(WR_q) is the same as #Hom($H[\Gamma(b)]$), and by Observation 2.12 we obtain #Hom($H[\Gamma(b)]$) $\leq_{\mathrm{AP}}$ #Ret(H). For $q \geq 4$ Dyer et al. [5, Lemma 26] show #SAT $\leq_{\mathrm{AP}}$ #Hom(WR_q). For $q = 3$ this fact is due to Kelk [33, Section 2.3]. Summarising we obtain

$$\#\mathrm{SAT} \leq_{\mathrm{AP}} \#\mathrm{Hom}(\mathrm{WR}_q) \equiv_{\mathrm{AP}} \#\mathrm{Hom}(H[\Gamma(b)]) \leq_{\mathrm{AP}} \#\mathrm{Ret}(H).$$

□

Recall the 2-Wrench as given in Figure 7.

Lemma 2.15. *Let H be a triangle-free graph that has a looped vertex b which has an unlooped neighbour. If $H[\Gamma(b)]$ is not isomorphic to a 2-Wrench, then #SAT $\leq_{\mathrm{AP}}$ #Ret(H).*

Proof. By Observation 2.12 we know #Hom($H[\Gamma(b)]$) $\leq_{\mathrm{AP}}$ #Ret(H). We show #SAT $\leq_{\mathrm{AP}}$ #Hom($H[\Gamma(b)]$) to obtain #SAT $\leq_{\mathrm{AP}}$ #Ret(H).

To shorten the notation let $H_b = H[\Gamma(b)]$. We consider different cases depending on the graph H_b. By assumption the vertex b is looped and has at least one unlooped neighbour. First consider the case where H_b has at most 4 vertices. Since, by assumption, H_b is triangle-free and not isomorphic to a 2-Wrench, it has to be isomorphic to one of the graphs depicted in Figure 8. Approximately counting homomorphisms to the first graph in Figure 8 is well-known to be equivalent to #IS (the problem of approximately counting independent sets in a graph) which is #SAT-equivalent [5, Theorem 3]. The second and fourth graphs correspond to weighted versions of #IS which are known to be #SAT-equivalent [33, Lemma 2.3]. The third graph is the so-called 1-Wrench and the corresponding #SAT-hardness is shown in [5, Theorem 21]. Finally, approximately counting homomorphisms to the fifth graph in Figure 8 is shown to be #SAT-hard in [33, Section 2.3].

Now consider the case where H_b has 5 or more vertices. We claim that, under this assumption, Lemma 2.13 gives #SAT $\leq_{\mathrm{AP}}$ #Hom(H_b). To see this, note that $F(H_b) = \{b\}$ and $|F(H_b)||V(H_b)| \geq 5$. Consider any pair (S, T) with $\emptyset \subseteq S, T \subseteq V(H_b)$, $S \subseteq \Gamma(T)$ and $T \subseteq \Gamma(S)$. We distinguish between different cases depending on the cardinalities of S and T and show that in each case the conditions of Lemma 2.13 are fulfilled.

- If $|S| = 1$ then either item (1) or item (3) of Lemma 2.13 are satisfied.
- If $|T| = 1$ then either item (2) or item (3) of Lemma 2.13 are satisfied.

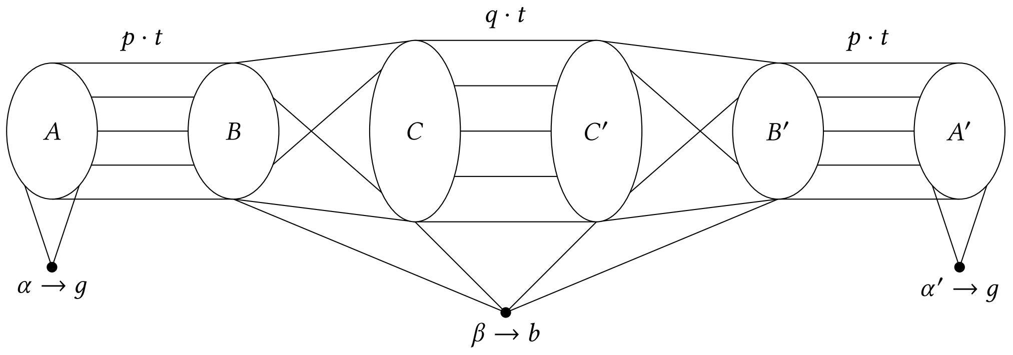
$p \cdot t$
$q \cdot t$
$p \cdot t$
A
B
C
C'
B'
A'
$\alpha \longrightarrow g$
$\alpha' \longrightarrow g$
$\beta \longrightarrow b$

Fig. 9. The graph J. A label of the form $v \longrightarrow u$ means that the vertex $v \in V(J)$ is pinned to $u \in V(H)$ since $S_v = \{u\}$.

- If $|S| \geq 3$ then $T = \{b\}$ since $T \subseteq \Gamma(S)$ and H is triangle-free. So $|T| = 1$.
- If $|T| \geq 3$ then $S = \{b\}$ since $S \subseteq \Gamma(T)$ and H is a triangle-free. So $|S| = 1$.
- If $|S| = |T| = 2$ then $|S| \cdot |T| = 4$ and item (3) of Lemma 2.13 is satisfied.

So Lemma 2.13 gives #SAT $\leq_{\text{AP}}$ #HOM(H_b). □

The goal of the remainder of this section is to show Lemma 2.30, in which we prove #SAT-hardness using distance-2 neighbourhoods of vertices in H. In order to show #SAT-hardness we use a reduction from counting large cuts in a graph G. We use graph gadgets to model these cuts. We replace each vertex of G by a graph J such that the number of homomorphisms from J to H is dominated by exactly two "types" of homomorphisms. These two types encode the two parts of a cut. In Table 1 we give all types that represent a significant share of the set of homomorphisms. In Lemma 44 we show how to choose parameters of the graph J to ensure that only 2 significant types remain. In the proof of Lemma 45 we verify another desired property, which is that the two types interact in an "anti-ferromagnetic" way to ensure that large cuts dominate.

At this point we introduce the gadget graph J and introduce some of its properties. Note that a similar but simpler gadget has been used in [5] and [33].

Definition 2.16. For sets X and Y we define $X \times Y = \{\{x, y\} \mid x \in X, y \in Y\}$ as an undirected version of the usual definition of the Cartesian product.

Definition 2.17. We now define the graph J, as visualised in Figure 9. Let p, q and t be positive integers — these are parameters of J. Let A, A', B and B' be independent sets of size $p \cdot t$ and let C and C' be independent sets of size $q \cdot t$. These six sets are pairwise disjoint. In addition, we introduce vertices α, α' and β that are distinct from each other and the remaining vertices. The vertex set of J is the union of $\{\alpha, \alpha', \beta\}$ and the sets A, A', B, B', C and C'. As displayed in Figure 9, the edge set of J is defined as follows. The set of edges $\mathcal{M}_1$ between the vertices of A and B forms a perfect matching (every vertex in A is adjacent to exactly one vertex in B and vice versa). The set of edges $\mathcal{M}_2$ between the vertices of C and C' and the edges $\mathcal{M}_3$ between the vertices of A' and B' form perfect matchings respectively. Then

$$\begin{aligned} E(J) = &\bigcup_{i \in [3]} \mathcal{M}_i \cup \Big(B \times C\Big) \cup \Big(B' \times C'\Big) \\ &\cup \Big(\{\alpha\} \times A\Big) \cup \Big(\{\alpha'\} \times A'\Big) \cup \Big(\{\beta\} \times (B \cup C \cup C' \cup B')\Big). \end{aligned}$$

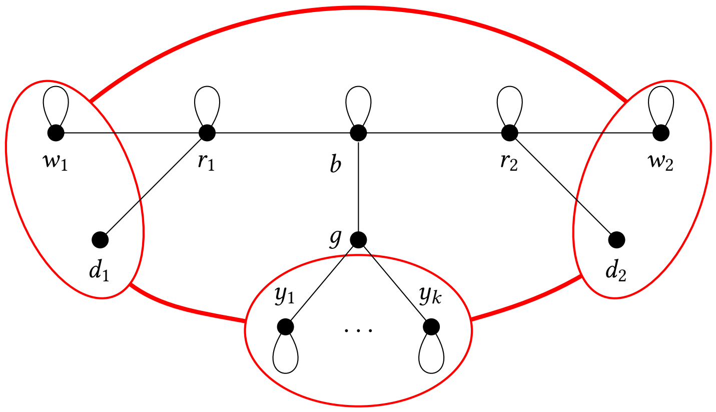

Fig. 10. The graph H_k. Circled sets of vertices are independent sets of possibly looped vertices. Sets of vertices that are connected by a thick red edge have a complete set of edges between them.

This completes the definition of the graph J.

For any positive integer k let H_k be the graph as shown in Figure 10. The vertex set of H_k is $\{w_1, d_1, r_1, w_2, d_2, r_2, b, g, y_1, \dots, y_k\}$. All of these vertices are looped except for d_1, d_2 and g. The non-loop edges of H_k are the edges in

$$\{\{w_1, r_1\}, \{w_2, r_2\}, \{d_1, r_1\}, \{d_2, r_2\}, \{r_1, b\}, \{r_2, b\}, \{b, g\}, \{g, y_1\}, \dots, \{g, y_k\}\},$$

together with those in $\{w_1, d_1\} \times \{w_2, d_2\}$, $\{w_1, d_1\} \times \{y_1, \dots, y_k\}$ and $\{w_2, d_2\} \times \{y_1, \dots, y_k\}$. The significance of this graph will become clear in the proof of Lemma 2.30.

For a graph J we define the vertex lists $S_\alpha = \{g\}$, $S_{\alpha'} = \{g\}$, and $S_\beta = \{b\}$. Also, for all $v \in V(J) \setminus \{\alpha, \alpha', \beta\}$, we define $S_v = V(H_k)$. Finally, we let $\mathrm{S}_J = \{S_v \mid v \in V(J)\}$. In order to investigate the number of homomorphisms from (J, S_J) to H_k, we set up the following notation. Suppose that U and V are subsets of $V(J)$ and that h is a homomorphism $h \in \mathcal{H}((J, \mathrm{S}_J), H_k)$. We define

- $h(V) = \{h(x) \mid x \in V\}$ and
- $h(U, V) = \{(h(x), h(y)) \mid x \in U, v \in V, \{x, y\} \in E(J)\}$.

We say that $(h(A, B), h(C, C'), h(B', A'))$ is the *type* of h. We will partition the set $\mathcal{H}((J, \mathrm{S}_J), H_k)$ into different classes by type. Formally, a type is a tuple $T = (T_1, T_2, T_3)$ where each T_i is a subset of $\{(x, y) \mid x \in V(H_k), y \in V(H_k), \{x, y\} \in E(H_k)\}$. The type of a homomorphism gives a lot of information. Given a type $T = (T_1, T_2, T_3)$, let $A(T) = \{x \mid \exists y\, (x, y) \in T_1\}$, $B(T) = \{y \mid \exists x\, (x, y) \in T_1\}$, $C(T) = \{x \mid \exists y\, (x, y) \in T_2\}$, $C'(T) = \{y \mid \exists x\, (x, y) \in T_2\}$, $B'(T) = \{x \mid \exists y\, (x, y) \in T_3\}$, and $A'(T) = \{y \mid \exists x\, (x, y) \in T_3\}$. If a homomorphism $h \in \mathcal{H}((J, \mathrm{S}_J), H_k)$ has type T then it is clear from the definition of J that $h(A) = A(T)$, $h(B) = B(T)$, $h(C) = C(T)$, $h(C') = C'(T)$, $h(B') = B'(T)$ and $h(A') = A'(T)$. A type T is called *non-empty* if there exists a homomorphism from (J, S_J) to H_k that has type T, otherwise it is called empty. The following observation follows from the definition of J.

Observation 2.18. *A type $T = (T_1, T_2, T_3)$ is non-empty if and only if*

(1) T_1, T_2 and T_3 are non-empty,
(2) $B(T) \cup C(T) \cup C'(T) \cup B'(T) \subseteq \Gamma(b)$,
(3) $A(T) \cup A'(T) \subseteq \Gamma(g)$,

(4) $B(T) \times C(T) \subseteq E(H_k)$ and $B'(T) \times C'(T) \subseteq E(H_k)$.

Given a type $T = (T_1, T_2, T_3)$ we define $N(T)$ to be the number of homomorphisms in $\mathcal{H}((J, \mathrm{S}_J), H_k)$ that have type T. We also set $\widehat{N}(T) = |T_1|^{pt}|T_2|^{qt}|T_3|^{pt}$. In Lemma 2.20 we show that, for non-empty T, $\widehat{N}(T)$ is a close approximation to $N(T)$.

We use the following technical fact. Let $\left\{ {a \atop b} \right\}$ be the Stirling number of the second kind, i.e. the number of surjective functions from a set of a elements to a set of b elements.

Lemma 2.19 ([5, Lemma 18]). *If a and b are positive integers and $a \geq 2b \ln b$, then*

$$b^a\left(1 - \exp\left(-\frac{a}{2b}\right)\right) \leq \left\{ {a \atop b} \right\} \leq b^a.$$

Lemma 2.20. *Let p and q be positive integers. There exists a positive integer t_0 such that for all $t \geq t_0$ and all non-empty types T of the corresponding graph J we have*

$$\frac{\widehat{N}(T)}{2} \leq N(T) \leq \widehat{N}(T).$$

Proof. Let $T = (T_1, T_2, T_3)$ be a non-empty type. Then

$$N(T) = \left\{ {p \cdot t \atop |T_1|} \right\} \cdot \left\{ {q \cdot t \atop |T_2|} \right\} \cdot \left\{ {p \cdot t \atop |T_3|} \right\}. \tag{5}$$

For fixed p and q and sufficiently large t_0 we know from Lemma 2.19 that for all $t \geq t_0$ we have

$$1 - \exp\left(-\frac{p \cdot t}{2|T_1|}\right) \geq (1/2)^{1/3},$$

an analogous bound holds for the other two factors in Equation (5). The statement of the lemma then directly follows from Lemma 2.19. □

Definition 2.21. We say that a type $T = (T_1, T_2, T_3)$ is *maximal* if it is non-empty and every type $T' = (T'_1, T'_2, T'_3)$ with $T' \neq T$, $T_1 \subseteq T'_1$, $T_2 \subseteq T'_2$ and $T_3 \subseteq T'_3$ is empty.

Using this definition of maximality we prove that the number of homomorphisms in $\mathcal{H}((J, \mathrm{S}_J), H_k)$ that have a maximal type is exponentially larger as a function of t than the number of homomorphisms that have non-maximal types. Note that the precise value of the fraction $\frac{31+12k}{32+12k}$ that appears in the following lemmas is not important, we only need it to be smaller than 1. This particular bound uses the fact that, for any type (T_1, T_2, T_3), the sets T_1, T_2 and T_3 have cardinality at most $2|E(H_k)| = 32 + 12k$.

Constraint 2.22. In our proofs we will need the fact that the parameters p and q of J are sufficiently large with respect to the number of edges in H_k. In particular, we require that $p, q \geq 2|E(H_k)| = 32 + 12k$.

Lemma 2.23. *Let T be a non-empty type that is not maximal. Then there exists a non-empty type T^* such that $\widehat{N}(T) \leq \left(\frac{31+12k}{32+12k}\right)^t \widehat{N}(T^*)$.*

Proof. Let $T = (T_1, T_2, T_3)$ be a non-empty type that is not maximal. Then there exists a non-empty type $T^* = (T^*_1, T^*_2, T^*_3)$ with $T^* \neq T$ and $T_i \subseteq T^*_i$ for $i \in [3]$. Since $T^* \neq T$ there exists an index $i \in [3]$ such that $T_i \subsetneq T^*_i$, i.e. $|T_i| \leq |T^*_i| - 1$. Then (using the fact that $p, q \geq 1$)

$$\frac{\widehat{N}(T)}{\widehat{N}(T^*)} = \frac{|T_1|^{pt}|T_2|^{qt}|T_3|^{pt}}{|T^*_1|^{pt}|T^*_2|^{qt}|T^*_3|^{pt}} \leq \left(\frac{|T^*_i| - 1}{|T^*_i|}\right)^t \leq \left(\frac{2|E(H_k)| - 1}{2|E(H_k)|}\right)^t \leq \left(\frac{31 + 12k}{32 + 12k}\right)^t.$$

□

Definition 2.24. Let $E = E(H_k)$. For all $X \subseteq V(H_k)$ and $Y \subseteq V(H_k)$ we set $E(X, Y) = \{(x, y) \mid x \in X, y \in Y, \{x, y\} \in E\}$.

For a set of vertices S in a graph H recall the definition of the set of common neighbours $\Gamma(S)$ and the set of all neighbours $\Phi(S)$ from Section 1.5.

Lemma 2.25. *Let $T = (T_1, T_2, T_3)$ be a maximal type. Then*

(1) $T_1 = E(A(T), B(T))$, $T_2 = E(C(T), C'(T))$ and $T_3 = E(B'(T), A'(T))$. Also,
(2) $C(T) = \Gamma\big(\Gamma(C(T)) \cap \Gamma(b)\big) \cap \Gamma(b)$ and $C'(T) = \Gamma\big(\Gamma(C'(T)) \cap \Gamma(b)\big) \cap \Gamma(b)$.
(3) $B(T) = \Gamma(C(T)) \cap \Gamma(b)$ and $B'(T) = \Gamma(C'(T)) \cap \Gamma(b)$.
(4) $A(T) = \Phi(B(T)) \cap \Gamma(g)$ and $A'(T) = \Phi(B'(T)) \cap \Gamma(g)$.

Proof. Let $T = (T_1, T_2, T_3)$ be a non-empty type.

Proof of (1): It is clear from the definitions that $T_1 \subseteq E(A(T), B(T))$, $T_2 \subseteq E(C(T), C'(T))$ and $T_3 \subseteq E(B'(T), A'(T))$. Suppose that T_2 is a strict subset of $E(C(T), C'(T))$. We will show that T is not maximal. To this end, consider the type $T^* = (T_1, E(C(T), C'(T)), T_3)$. Note that $A(T^*) = A(T)$, $B(T^*) = B(T)$, $C(T^*) = C(T)$, $C'(T^*) = C'(T)$, $B'(T^*) = B'(T)$ and $A'(T^*) = A'(T)$. Using Observation 2.18 and the fact that T is non-empty, we conclude that T^* is non-empty. Using the definition of maximality (comparing T to T^*) we conclude that T is not maximal. Similarly, if T_1 is a strict subset of $E(A(T), B(T))$ or if T_3 is a strict subset of $E(B'(T), A'(T))$ then T is not maximal.

Proof of (2): Let $X = \Gamma\big(\Gamma(C(T)) \cap \Gamma(b)\big)$ and $S = X \cap \Gamma(b)$. If $y \in \Gamma(C(T)) \cap \Gamma(b)$ then y is certainly adjacent to everything in $C(T)$, so $C(T) \subseteq X$. Since $C(T) \subseteq \Gamma(b)$ by Observation 2.18, we conclude that $C(T)$ is a subset of S. Similarly, defining $X' = \Gamma\big(\Gamma(C'(T)) \cap \Gamma(b)\big)$ and $S' = X' \cap \Gamma(b)$, we have $C'(T) \subseteq S'$. Thus, $T_2 \subseteq E(S, S')$. Consider the type $T^* = (T_1, E(S, S'), T_3)$.

- We first show that T^* is non-empty. Note that $A(T^*) = A(T)$, $B(T^*) = B(T)$, $A'(T^*) = A'(T)$ and $B'(T^*) = B'(T)$. Also, $C(T^*) \subseteq S \subseteq \Gamma(b)$ and $C'(T^*) \subseteq S' \subseteq \Gamma(b)$. Using Observation 2.18 and the fact that T is non-empty, we must check that $B(T) \times C(T^*) \subseteq E(H_k)$ and $B'(T) \times C'(T^*) \subseteq E(H_k)$. To do this, we will check that $B(T) \times S \subseteq E(H_k)$ and $B'(T) \times S' \subseteq E(H_k)$.
  We start with the first of these. Since T is non-empty, Observation 2.18 guarantees that $B(T) \subseteq \Gamma(C(T)) \cap \Gamma(b)$. So it suffices to show that $\big(\Gamma(C(T)) \cap \Gamma(b)\big) \times S \subseteq E(H_k)$, which follows from the definition of S. The proof that $B'(T) \times S' \subseteq E(H_k)$ is similar. We have shown that T^* is non-empty.
- We next show that $C(T^*) = S$. We have already established that $C(T^*) \subseteq S$. The vertex b is adjacent to everything in $\Gamma(b)$ so it is adjacent to everything in the subset $\Gamma(C'(T)) \cap \Gamma(b)$ hence $b \in X'$. Since b has a loop, this implies $b \in S'$. By the definition of T^* it follows that $S \subseteq C(T^*)$, and hence $C(T^*) = S$, as required. We can similarly show that $C'(T^*) = S'$.

Suppose that $C(T)$ is a strict subset of S. Comparing T to T^*, we find that T_2 is a strict subset of $E(S, S')$ so T is not maximal. Similarly, if $C'(T)$ is a strict subset of S' then T is not maximal.

Proof of (3): It is immediate from Observation 2.18 that $B(T) \subseteq \Gamma(C(T)) \cap \Gamma(b)$ and $B'(T) \subseteq \Gamma(C'(T)) \cap \Gamma(b)$.

Suppose that $B(T)$ is a strict subset of $\Gamma(C(T)) \cap \Gamma(b)$. We will show that T is not maximal. To this end, let v be any vertex in $\Gamma(C(T)) \cap \Gamma(b) \setminus B(T)$ and consider the type $T^* = (T_1 \cup \{(b, v)\}, T_2, T_3)$. Observation 2.18 shows that T^* is non-empty, so T is not maximal. Similarly, if $B'(T)$ is a strict subset of $\Gamma(C'(T)) \cap \Gamma(b)$ then T is not maximal.

Proof of (4): It is immediate from Observation 2.18 and the definition of a type that $A(T) \subseteq \Phi(B(T)) \cap \Gamma(g)$ and $A'(T) \subseteq \Phi(B'(T)) \cap \Gamma(g)$. If either of these subset inclusions is strict then, as in the proof of (3), it is straightforward to see that T is not maximal. □

Lemma 2.26. *Let T be a maximal type. Then $C(T)$ and $C'(T)$ are both in the set*

$$\{\{b\}, \{r_1, b\}, \{r_2, b\}\{r_1, r_2, b, g\}\}.$$

Proof. We will prove this for $C(T)$. The argument for $C'(T)$ is the same. From Observation 2.18, $C(T)$ is a (not necessarily strict) subset of $\Gamma(b) = \{r_1, r_2, b, g\}$ (and it is non-empty).

- If $g \in C(T)$ then $\Gamma(C(T)) \cap \Gamma(b) = \{b\}$ so, by item (2) of Lemma 2.25, $C(T) = \Gamma(b) = \{r_1, r_2, b, g\}$.
- If $r_1 \in C(T)$ and $r_2 \in C(T)$ then $\Gamma(C(T)) \cap \Gamma(b) = \{b\}$ so, again, $C(T) = \Gamma(b) = \{r_1, r_2, b, g\}$.
- If $C(T) = \{r_1\}$ then $\Gamma(C(T)) \cap \Gamma(b) = \{r_1, b\}$ so, by item (2) of Lemma 2.25, $C(T) = \{r_1, b\}$, which is a contradiction.
- Similarly, the case $C(T) = \{r_2\}$ gives a contradiction.

This covers all possible cases. □

Definition 2.27. For $i \in [6]$ let $X_i \subseteq V(H_k)$. We say that the types $(E(X_1, X_2), E(X_3, X_4), E(X_5, X_6))$ and $(E(X_6, X_5), E(X_4, X_3), E(X_2, X_1))$ are *symmetric* to each other.

Note that if T and T' are symmetric to each other it holds that $N(T) = N(T')$.

Table 1. Maximal types of the homomorphisms in $\mathcal{H}(J, \mathrm{S}_J), H_k)$.

	$A(T)$	$B(T)$	$C(T)$	$C'(T)$	$B'(T)$	$A'(T)$	$\widehat{N}(T)$
T_1	$\{b\} \cup \mathcal{Y}$	$\{r_1, r_2, b, g\}$	$\{b\}$	$\{b\}$	$\{r_1, r_2, b, g\}$	$\{b\} \cup \mathcal{Y}$	$(4+k)^{pt} \cdot 1^{qt} \cdot (4+k)^{pt}$
T_2	$\{b\} \cup \mathcal{Y}$	$\{r_1, r_2, b, g\}$	$\{b\}$	$\{r_1, b\}$	$\{r_1, b\}$	$\{b\}$	$(4+k)^{pt} \cdot 2^{qt} \cdot 2^{pt}$
T_3	$\{b\} \cup \mathcal{Y}$	$\{r_1, r_2, b, g\}$	$\{b\}$	$\{r_2, b\}$	$\{r_2, b\}$	$\{b\}$	$(4+k)^{pt} \cdot 2^{qt} \cdot 2^{pt}$
T_4	$\{b\} \cup \mathcal{Y}$	$\{r_1, r_2, b, g\}$	$\{b\}$	$\{r_1, r_2, b, g\}$	$\{b\}$	$\{b\}$	$(4+k)^{pt} \cdot 4^{qt} \cdot 1^{pt}$
T_5	$\{b\}$	$\{r_1, b\}$	$\{r_1, b\}$	$\{r_2, b\}$	$\{r_2, b\}$	$\{b\}$	$2^{pt} \cdot 3^{qt} \cdot 2^{pt}$
T_6	$\{b\}$	$\{r_1, b\}$	$\{r_1, b\}$	$\{r_1, b\}$	$\{r_1, b\}$	$\{b\}$	$2^{pt} \cdot 4^{qt} \cdot 2^{pt}$
T_7	$\{b\}$	$\{r_2, b\}$	$\{r_2, b\}$	$\{r_2, b\}$	$\{r_2, b\}$	$\{b\}$	$2^{pt} \cdot 4^{qt} \cdot 2^{pt}$
T_8	$\{b\}$	$\{r_1, b\}$	$\{r_1, b\}$	$\{r_1, r_2, b, g\}$	$\{b\}$	$\{b\}$	$2^{pt} \cdot 6^{qt} \cdot 1^{pt}$
T_9	$\{b\}$	$\{r_2, b\}$	$\{r_2, b\}$	$\{r_1, r_2, b, g\}$	$\{b\}$	$\{b\}$	$2^{pt} \cdot 6^{qt} \cdot 1^{pt}$
T_{10}	$\{b\}$	$\{b\}$	$\{r_1, r_2, b, g\}$	$\{r_1, r_2, b, g\}$	$\{b\}$	$\{b\}$	$1^{pt} \cdot 9^{qt} \cdot 1^{pt}$

Note: Recall that p, q and t are the parameters of J where p and q satisfy Constraint 2.22. Each line corresponds to a type $\Big(E\big(A(T), B(T)\big), E\big(C(T), C'(T)\big), E\big(B'(T), A'(T)\big)\Big)$. To shorten the notation we set $\mathcal{Y} = \{y_i \mid i \in [k]\}$.

Lemma 2.28. *All maximal types are listed in Table 1 (except for those that are symmetric to a listed type). Furthermore, for each listed type T we give the corresponding value for $\widehat{N}(T)$.*

Proof. First, Lemma 2.26 gives the 4 possibilities for $C(T)$ and $C'(T)$. Up to symmetry, this gives the 10 possibilities listed in the table.

Next, for each of the 10 possibilities, we use items (3) and (4) of Lemma 2.25 to compute the corresponding sets $A(T)$, $B(T)$, $B'(T)$ and $A'(T)$.

Now item (1) of Lemma 2.25 guarantees that $T_1 = E(A(T), B(T))$, $T_2 = E(C(T), C'(T))$ and $T_3 = E(B'(T), A'(T))$. So

$$\widehat{N}(T) = |E(A(T), B(T))|^{pt} |E(C(T), C'(T))|^{qt} |E(B'(T), A'(T))|^{pt}.$$

These quantities are all computed in the table. □

Let $T_1, \dots, T_{10}$ be the types as given in Table 1.

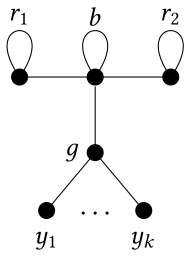

Fig. 11. The graph H'_k.

Lemma 2.29. *Let k be a positive integer. Then there is a $\gamma \in (0,1)$ and positive integers p and q that satisfy Constraint 2.22 such that, for all $i \in [10]$ except $i = 4$ and all positive integers t, we have $\widehat{N}(T_i) \leq \gamma^t \widehat{N}(T_4)$.*

Proof. We choose integers $p, q \geq 32 + 12k$ (p and q satisfy Constraint 2.22) such that

$$\log_4(4+k) < \frac{q}{p} < \log_{9/4}(4+k). \tag{6}$$

This is possible as $\log_4(4+k) < \log_{9/4}(4+k)$ for all $k > 0$. Suppose that T and T' are types listed in Table 1 which are distinct from T_4 and have the property that $\widehat{N}(T') < \widehat{N}(T)$. Then the sought-for bound automatically holds for T' if it holds for T.

We check the sought-for bound for each $i \in [10]$, $i \neq 4$:

T_1: $\frac{\widehat{N}(T_1)}{\widehat{N}(T_4)} = (4+k)^{pt}(1/4)^{qt} < \gamma^t$ is fulfilled for some sufficiently large $\gamma < 1$ if and only if $(4+k)^p/4^q < 1$. This is true as $\log_4(4+k) < \frac{q}{p}$ by (6).

T_2 (and T_3): $\frac{\widehat{N}(T_2)}{\widehat{N}(T_4)} = 2^{pt}(1/2)^{qt} < \gamma^t$ is fulfilled for some sufficiently large $\gamma < 1$ if and only if $2^p/2^q < 1$. This is true as $q > p$ by (6).

T_5: $\widehat{N}(T_5) < \widehat{N}(T_6)$.

T_6 (and T_7): $\frac{\widehat{N}(T_6)}{\widehat{N}(T_4)} = (4/(4+k))^{pt} < \gamma^t$ is fulfilled for $4/(4+k) \leq 4/5 < \gamma < 1$.

T_8 (and T_9): $\frac{\widehat{N}(T_8)}{\widehat{N}(T_4)} = (2/(4+k))^{pt}(3/2)^{qt} < \gamma^t$ is fulfilled for some sufficiently large $\gamma < 1$ if and only if $(2/(4+k))^p(3/2)^q < 1$. This is true as $\frac{q}{p} < \log_{9/4}(4+k) < \log_{3/2}((4+k)/2)$ by (6) and for all $k > 0$.

T_{10}: $\frac{\widehat{N}(T_{10})}{\widehat{N}(T_4)} = (1/(4+k))^{pt}(9/4)^{qt} < \gamma^t$ is fulfilled for some sufficiently large $\gamma < 1$ if and only if $(1/(4+k))^p(9/4)^q < 1$. This is true as $\frac{q}{p} < \log_{9/4}(4+k)$ by (6).

□

Now we have collected all properties of the gadget graph J that we need to prove Lemma 2.30. We will see that this lemma is the final piece to show the classification for graphs of girth at least 5 stated in Theorem 1.1. In the statement of the lemma we refer to the graph H'_k as depicted in Figure 11.

Lemma 2.30. *Let H be graph that has a looped vertex b such that, for some positive integer k, H'_k (see Figure 11) is a subgraph of $H[\Gamma^2(b)]$, and $H[\Gamma^2(b)]$ in turn is a subgraph of H_k (see Figure 10). Then #SAT $\leq_{\mathrm{AP}}$ #Ret(H).*

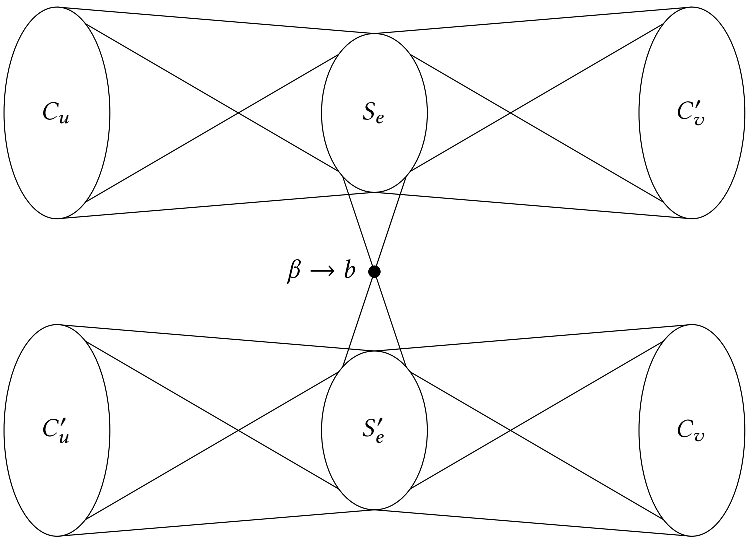

Fig. 12. The edge gadget for the edge $e = \{u, v\}$.

Proof. We use a reduction from #LargeCut, which is known to be #SAT-hard (see [5]). A *cut* of a graph G is a partition of $V(G)$ into two subsets (the order of this pair is ignored) and the size of a cut is the number of edges that have exactly one endpoint in each of these two subsets.

Name: #LargeCut.

Input: An integer $K \geq 1$ and a connected graph G in which every cut has size at most K.

Output: The number of size-K cuts in G.

Let G and K be an input to #LargeCut, n be the number of vertices of G and $\varepsilon \in (0, 1)$ be the parameter of the desired precision of approximation in the AP-reduction. From G we construct an input (G', S) to #Ret(H) by introducing vertex and edge gadgets. By the assumption of the lemma, the vertex b of H has $\Gamma(b) = \{b, r_1, r_2, g\}$ where b, r_1 and r_2 are looped and g is not and $\Gamma(g) = \{b, y_1, \dots, y_k\}$ with $k \geq 1$.

Let p, q be positive integers that are chosen such that they fulfil Constraint 2.22 and (6). Note that p and q only depend on k which is a parameter of the fixed graph H and therefore do not depend on the input G. We will define the parameter t of the gadget graph J to be $t = n^4$. We also define a new parameter $s = n + 1$.

For each vertex $v \in V(G)$ we introduce a vertex gadget G'_v which is a graph J with parameters p, q and t as given in Definition 2.17. We denote the corresponding sets A, B, C, C', B', A' by $A_v, B_v, C_v, C'_v, B'_v$ and A'_v, respectively. It is fine to keep the notation for the remaining vertices as α, α' and β as technically these vertices can be thought of as identical vertices over all gadgets because of their pinning. We say that two gadgets G'_u and G'_v are adjacent if u and v are adjacent in G.

We connect vertex gadgets as follows. For every edge $e = \{u, v\} \in E(G)$ we introduce an edge gadget as follows. We introduce two size-s independent sets, denoted by S_e and S'_e. We set $V'_e = S_e \cup S'_e$. As shown in Figure 12 we construct the set of edges

$$E'_e = (C_u \times S_e) \cup (C'_u \times S'_e) \cup (C_v \times S'_e) \cup (C'_v \times S_e) \cup (\{\beta\} \times S_e) \cup (\{\beta\} \times S'_e).$$

Putting the pieces together, G' is the graph with

$$V(G') = \bigcup_{v \in V(G)} V(G'_v) \cup \bigcup_{e \in E(G)} V'_e \qquad \text{and} \qquad E(G') = \bigcup_{v \in V(G)} E(G'_v) \cup \bigcup_{e \in E(G)} E'_e.$$

Finally, we define the vertex lists $S_\alpha = S_{\alpha'} = \{g\}$, $S_\beta = \{b\}$ and $S_v = V(H)$ for all $v \in V(G') \setminus \{\alpha, \alpha', \beta\}$. Then $\mathrm{S} = \{S_v \mid v \in V(G')\}$. This completes the definition of the instance (G', S).

The pinning of the vertex β (via the list S_β) ensures that every homomorphism from (G', S) to H is a homomorphism from (G', S) to $H[\Gamma^2(b)]$. By the assumption of the lemma, $H[\Gamma^2(b)]$ is a subgraph of H_k. We make a case distinction based on the graph $H[\Gamma^2(b)]$.

Case 1: $H[\Gamma^2(b)] = H_k$**.** Let h be a homomorphisms from (G', S) to H, v be some vertex of G and G'_v be the corresponding vertex gadget. Then by our definition of (G', S) we observe that $h|_{V(G'_v)}$ corresponds to a homomorphism from (J, S_J) to H_k and therefore has a type.

We say that a homomorphism from (G', S) to H is *full* if its restriction to each vertex gadget is either of type T_4 (from Table 1) or of its symmetric type (let us call it T'_4). Each full homomorphism h defines a cut as it partitions $V(G)$ into those vertices v for which $h|_{G'_v}$ has type T_4 and those for which $h|_{G'_v}$ has type T'_4. We say that a full homomorphism is K*-large* if the size of the corresponding cut is equal to K, otherwise we say that the homomorphism is K*-small.* Consider a full homomorphism h from (G', S) to H_k.

- For an edge $e = \{u, v\}$ of G suppose that $h|_{G'_u}$ has type T_4 and $h|_{G'_v}$ has type T'_4. Note that by the definition of the edge gadget, we have $h(S_e) \subseteq \Gamma(h(C_u)) \cap \Gamma(h(C'_v))$. Then the vertices in S_e can be mapped to any of the 4 neighbours of b, whereas all vertices in S'_e have to be mapped to b (since $h(S'_e) \subseteq \Gamma(h(C'_u)) \cap \Gamma(h(C_v))$ where $C'_u = C_v = \{r_1, r_2, b, g\}$ and b is the sole common neighbour of r_1, r_2, b and g).
- Suppose instead that $h|_{G'_u}$ and $h|_{G'_v}$ have the same type T_4 or T'_4. Then the homomorphism h has to map the vertices in both S_e and S_v to b.

Thus, every pair of adjacent gadgets of different types contributes a factor of 4^s to the number of full homomorphisms, whereas every pair of adjacent gadgets of the same type only contributes a factor of 1. Recall the definition of $N(T)$ as the number of homomorphisms from (J, S_J) to H_k that have type T. Then for $\ell \geq 1$ every size-ℓ cut of G arises in $2 \cdot N(T_4)^n \cdot 4^{s\ell}$ ways as a full homomorphism from (G', S) to H_k.

Let N be the number of solutions to #LargeCut with input G and K (our goal is to approximate this number). We partition the homomorphisms from (G', S) to H_k into three different sets. Z^* is the number of K-large (full) homomorphisms, Z_1 is the number of homomorphisms that are full but K-small and Z_2 is the number of non-full homomorphisms. Then we have $N = Z^*/(2N(T_4)^n 4^{sK})$ and $N\big((G', \mathrm{S}) \to H\big) = N\big((G', \mathrm{S}) \to H_k\big) = Z^* + Z_1 + Z_2$. Thus it remains to show that $(Z_1 + Z_2)/(2N(T_4)^n 4^{sK}) \leq 1/4$ for our choice of p, q, t and s. Under this assumption we then have $N\big((G', \mathrm{S}) \to H\big)/(2N(T_4)^n 4^{sK}) \in [N, N + 1/4]$ and a single oracle call to determine $N\big((G', \mathrm{S}) \to H\big)$ with precision $\delta = \varepsilon/21$ suffices to determine N with the sought-for precision as demonstrated in [5, Proof of Theorem 3].

Now we prove $(Z_1 + Z_2)/(2N(T_4)^n 4^{sK}) \leq 1/4$. As there are at most 2^n ways to assign a type T_4 or T'_4 to the n vertex gadgets in G' we have $Z_1 \leq 2^n \cdot N(T_4)^n \cdot 4^{s(K-1)}$. We next obtain the following bound since $s = n + 1$:

$$\frac{Z_1}{2N(T_4)^n 4^{sK}} \leq \frac{2^n N(T_4)^n 4^{s(K-1)}}{2N(T_4)^n 4^{sK}} = \frac{2^n}{2 \cdot 4^s} \leq \frac{1}{8}.$$

We obtain a similar bound for Z_2. From Lemmas 2.23, 2.28 and 2.29 we know that for our choice of p and q there exists $\gamma \in (0, 1)$ such that for every type T that is not T_4 or T'_4 we have $\widehat{N}(T) \leq \gamma^t \widehat{N}(T_4)$.

Using Lemma 2.20 this gives $N(T) \leq 2\gamma^t N(T_4)$ for sufficiently large t with respect to p, q and k (which only depend on H but not on the input G). Since $t = n^4$ we can assume that t is sufficiently large with respect to p and q as otherwise the input size is bounded by a constant (in which case we can solve #LargeCut in constant time).

For each type $T = (T_1, T_2, T_3)$, the cardinality of each set T_i is bounded above by $2|E(H_k)| = 32+12k$ and hence there are at most $\left(2^{32+12k}\right)^3$ different types. Furthermore, as H_k has $8 + k$ vertices, there are at most $(8 + k)^{2sn^2}$ possible functions from the at most $2sn^2$ vertices in $\bigcup_{e\in E(G)}(S_e \cup S'_e)$ to vertices in H_k. Since $t = n^4$ and $s = n + 1$ we obtain

$$\begin{aligned}\frac{Z_2}{2N(T_4)^n 4^{sK}} &\leq \frac{\left(2^{32+12k}\right)^{3n} \cdot N(T_4)^{n-1} \cdot 2\gamma^t N(T_4) \cdot (8 + k)^{2sn^2}}{2N(T_4)^n 4^{sK}} \\ &= \gamma^t \cdot \frac{\left(2^{32+12k}\right)^{3n}(8 + k)^{2sn^2}}{4^{sK}} \leq \frac{1}{8}.\end{aligned}$$

The last inequality holds for sufficiently large n as

$$\frac{\left(2^{32+12k}\right)^{3n}(8 + k)^{2sn^2}}{4^{sK}} \leq C^{n^3}$$

for some positive constant C that only depends on H, but not on the input G, whereas $t = n^4$. **(End of Case 1)**

Case 2: $H[\Gamma^2(b)] \neq H_k$. By the assumption of the lemma, $H[\Gamma^2(b)]$ is a subgraph of H_k. Let $\mathcal{H}'$ be the set of homomorphisms in $\mathcal{H}((J, \mathrm{S}_J), H_k)$ that are homomorphisms from J to $H[\Gamma^2(b)]$. Then for each type T the number of homomorphisms in $\mathcal{H}'$ of type T is at most the number of homomorphisms in $\mathcal{H}((J, \mathrm{S}_J), H_k)$ that have type T.

Note that the type T_4 (and its symmetric type) only uses vertices and edges from H'_k and we know that H'_k is a subgraph of $H[\Gamma^2(b)]$ by the assumption of the lemma. Therefore each homomorphism which is of type T_4 is also in $\mathcal{H}'$ (their number remains unchanged). The analysis is then analogous to that of Case 1. (The number of K-large and K-small homomorphisms stays the same whereas the number of non-full homomorphisms can only decrease as we only need to consider a subset of the previous types and the number of homomorphisms that have a particular type can only decrease.) **(End of Case 2)** □

2.3 Putting the Pieces together

Now finally we have all the tools at hand to prove the main classification result for counting retractions to graphs of girth at least 5, which we restate at this point.

Theorem 1.1. *Let H be a graph of girth at least* 5.

i) If every connected component of H is an irreflexive star, a single looped vertex, or an edge with two loops, then #Ret(H) is in FP.
ii) Otherwise, if every connected component of H is an irreflexive caterpillar or a partially bristled reflexive path, then #Ret(H) is approximation-equivalent to #BIS.
iii) Otherwise, #Ret(H) is approximation-equivalent to #SAT.

Proof. As in the proof of Theorem 2.3, the fact that the classification extends from connected graphs to graphs with multiple connected components follows from Remark 1.15. Now assume without loss of generality that H is a connected graph. If H is an irreflexive graph then the statement of the theorem follows from the slightly more general Theorem 2.3 (in the irreflexive case we only require H to be square-free).

Now suppose that H has at least one looped vertex. From Observation 1.2 we know that, in general, hardness results for #Hom(H) carry over to #Ret(H) and easiness results carry over from #LHom(H). Then, by Theorem 1.8, #Ret(H) is in FP if H is a single looped vertex or a single looped edge. Otherwise, since it is triangle-free, H cannot be a complete reflexive graph, so #Ret(H) is #BIS-hard with respect to AP-reductions by Theorem 1.9. The #BIS-easiness for partially bristled reflexive paths follows from our Lemma 2.4. Theorem 1.8 implies that #Ret(H) is always #SAT-easy.

It remains to show the #SAT-hardness result for graphs H that have at least one looped vertex but are not partially bristled reflexive paths. To this end we distinguish two disjoint cases:

(1) Suppose that every unlooped vertex in H has degree 1. Let H^* be the subgraph induced by the looped vertices of H. As all unlooped vertices have degree 1, the fact that H is connected implies that H^* is connected. Recall that WR_q is a reflexive star with q leaves. Then, in general, H^* is either a reflexive path, a reflexive cycle or it contains a subgraph WR_q for some $q \geq 3$.
  (a) Suppose that H^* is a reflexive path $u_1, \ldots, u_t$. By the fact that H is not a partially bristled reflexive path and all unlooped vertices have degree 1, it follows that either some u_i has more than one unlooped neighbour or at least one of the endpoints u_1 or u_t has an unlooped neighbour. Then #SAT-hardness follows from Lemma 2.15.
  (b) If H^* is a reflexive cycle, then by the fact that every unlooped vertex in H has degree 1, it holds that H^* is the only cycle in H. Then H is a pseudotree and, as H has girth at least 5, the reflexive cycle H^* has length at least 5. Therefore Ret(H) is NP-complete by Theorem 1.11 and it follows that #Ret(H) is #SAT-hard under AP-reductions by [5, Theorem 1].
  (c) If H^* contains a subgraph WR_q for some $q \geq 3$, then H contains a looped vertex with at least 3 looped neighbours apart from itself. As H is triangle-free, the subgraph WR_q is induced and #SAT-hardness follows either from Lemma 2.14 or Lemma 2.15.

(2) Suppose there exists an unlooped vertex in H that has degree at least 2. As H is connected and contains at least one looped vertex, it follows that there exists a looped vertex b with an unlooped neighbour g, which has degree $k + 1$ for some $k \geq 1$, i.e. has neighbours $y_1, \ldots, y_k$ apart from b. Then $H[\Gamma(b)]$ is isomorphic to a 2-Wrench, or otherwise hardness follows from Lemma 2.15. Therefore b has exactly 2 looped neighbours apart from itself. Let us call them r_1 and r_2. Then, as H has girth at least 5, the vertices $\{r_1, r_2, b, g, y_1, \ldots, y_k\}$ are distinct. This shows that $V(H'_k) \subseteq V(H[\Gamma^2(b)])$ and $E(H'_k) \subseteq E(H[\Gamma^2(b)])$, i.e. that H'_k (see Figure 11) is a subgraph of $H[\Gamma^2(b)]$.
  For $i = 1, 2$ the following hold:
  (a) Apart from b and r_i itself, the vertex r_i has at most 1 other looped neighbour, or otherwise hardness follows either from Lemma 2.14 or from Lemma 2.15.
  (b) If r_i has an unlooped neighbour, then $H[\Gamma(r_i)]$ is isomorphic to a 2-Wrench, or otherwise hardness follows from Lemma 2.15.
  From items 2a and 2b it follows that for $i = 1, 2$ the vertex r_i has at most one looped and one unlooped neighbour apart from b and r_i itself. (If they exist let us call the looped neighbour w_i and the unlooped neighbour d_i.) Therefore, $V(H[\Gamma^2(b)]) \subseteq \{w_1, d_1, r_1, w_2, d_2, r_2, b, g, y_1, \ldots, y_k\} \subseteq V(H_k)$.
  Note that d_1, d_2 and g are unlooped vertices in H. Furthermore, for $i = 1, 2$ we have shown the following
  - $E(H'_k) \subseteq E(H[\Gamma^2(b)])$.
  - $\{w_i, r_i\} \in E(H[\Gamma^2(b)])$ if $w_i \in V(H[\Gamma^2(b)])$.
  - $\{d_i, r_i\} \in E(H[\Gamma^2(b)])$ if $d_i \in V(H[\Gamma^2(b)])$.

  The edges $E(H'_k)$ together with $\{w_1, r_1\}$, $\{w_2, r_2\}$, $\{d_1, r_1\}$, $\{d_2, r_2\}$ (if these exist) form a tree on the vertices $\Gamma^2(b)$ (Recall that a tree might have loops but no cycles). By the fact that H

has girth at least 5, all named vertices are distinct, and it follows that $E(H[\Gamma^2(b)]) \subseteq E(H_k)$, which shows that $H[\Gamma^2(b)]$ is a subgraph of H_k.

Summarising, H'_k is a subgraph of $H[\Gamma^2(b)]$ and $H[\Gamma^2(b)]$ is a subgraph of H_k and we can apply Lemma 2.30 to obtain #SAT-hardness.

Items 1 and 2 cover all graphs H that have at least one looped vertex but are not partially bristled reflexive paths. (Note that item 1 includes the case where H is reflexive.) □

3 APPROXIMATELY COUNTING RETRACTIONS IS AS LEAST AS HARD AS COUNTING SURJECTIVE HOMOMORPHISMS OR COMPACTIONS

This section studies the place of the problem #Ret(H) within the landscape of a number of closely related counting problems. We now give formal definitions for these problems, which are parameterised by a graph H.

Let G be an irreflexive graph. A homomorphism $h\colon G \to H$ is said to be *surjective* if for every vertex $v \in V(H)$ there is a vertex $u \in V(G)$ such that $h(u) = v$. We use $N^{\text{sur}}(G \to H)$ to denote the number of surjective homomorphisms from G to H. Similarly, the homomorphism h is a *compaction* if it is surjective and for every non-loop edge $\{v_1, v_2\} \in E(H)$ there is is an edge $\{u_1, u_2\} \in E(G)$ such that $h(u_1) = v_1$ and $h(u_2) = v_2$. We use $N^{\text{comp}}(G \to H)$ to denote the number of compactions from G to H.

Name: #SHom(H).
Input: An irreflexive graph G.
Output: $N^{\text{sur}}(G \to H)$.

Name: #Comp(H).
Input: An irreflexive graph G.
Output: $N^{\text{comp}}(G \to H)$.

We also define the corresponding list versions of these two problems. We use $N^{\text{sur}}((G, \mathbf{S}) \to H)$ and $N^{\text{comp}}((G, \mathbf{S}) \to H)$ to denote the number of surjective homomorphisms from $(G, \mathbf{S})$ to H and the number of compactions from $(G, \mathbf{S})$ to H, respectively. Note that the list version of the problem #Ret(H) is simply the problem #LHom(H).

Name: #LSHom(H).
Input: An irreflexive graph G and a collection of lists $\mathbf{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$.
Output: $N^{\text{sur}}((G, \mathbf{S}) \to H)$.

Name: #LComp(H).
Input: An irreflexive graph G and a collection of lists $\mathbf{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$.
Output: $N^{\text{comp}}((G, \mathbf{S}) \to H)$.

Furthermore, we define a generalisation of the problems #Hom(H), #Ret(H) and #LHom(H). Let $2^{V(H)} = \{S \mid S \subseteq V(H)\}$ be the power set of $V(H)$. For a fixed graph H and a set $\mathcal{L} \subseteq 2^{V(H)}$ we define

Name: #Hom($H, \mathcal{L}$).
Input: An irreflexive graph G and a collection of lists $\mathbf{S} = \{S_v \in \mathcal{L} \mid v \in V(G)\}$.
Output: $N((G, \mathbf{S}) \to H)$.

As a measure of distance between two distributions π and π' on a finite universe Ω we use the *total variation distance* $\mathrm{d_{TV}}(\pi, \pi') = \frac{1}{2}\sum_{\omega\in\Omega}|\pi(\omega) - \pi'(\omega)|$. For a set $A \subseteq \Omega$, $\mathrm{Uni}(A)$ is the uniform distribution on A. Furthermore, $\mathrm{Be}(p)$ is the Bernoulli distribution with parameter p. In general, we write $X \sim D$ if a random variable X has distribution D.

3.1 Reductions using a Monte Carlo Approach

The main goal of this section is to prove Corollaries 3.4 and 3.6. Together they constitute Theorem 1.5 which states that both #SHom(H) and #Comp(H) are AP-reducible to #Ret(H).

In the following two lemmas we prove some necessary ingredients that we use in the proof of Lemma 3.3. From Section 1.5 recall that a RAS for $\#\textsc{Hom}(H,\mathcal{L})$ is an (ε,δ)-approximation for $\#\textsc{Hom}(H,\mathcal{L})$ with $\delta = 1/4$. First, we point out the well-known fact that this can be powered to obtain an (ε,δ)-approximation for smaller δ.

Lemma 3.1. *Let H be a graph and $\mathcal{L} \subseteq 2^{V(H)}$. There is an algorithm $\textsc{CountHom}_{H,\mathcal{L}}$ which uses oracle access to a RAS for $\#\textsc{Hom}(H,\mathcal{L})$ and has the following properties.*

- *It is given an input (G,S) to $\#\textsc{Hom}(H,\mathcal{L})$ together with accuracy parameters ε and δ in $(0,1)$.*
- *It returns a natural number X with $\Pr\left(e^{-\varepsilon} \le \frac{X}{N\left((G,\mathrm{S})\to H\right)} \le e^{\varepsilon}\right) \ge 1-\delta$.*
- *Its running time is bounded by a polynomial in ε^{-1}, $\log\delta^{-1}$, and the number of vertices of G.*

Proof. The lemma is basically the same as [32, Lemma 6.1] applied to the problem $\#\textsc{Hom}(H,\mathcal{L})$. The only difference is that [32, Lemma 6.1] gives a precision guarantee of the form

$$\Pr\left((1-\varepsilon) \le \frac{X}{N\left((G,\mathrm{S})\to H\right)} \le (1+\varepsilon)\right) \ge 1-\delta.$$

However, by Observation 1.13, using accuracy parameter $\varepsilon/2$ instead of ε in [32, Lemma 6.1] suffices to obtain the desired result. □

Second, we point out that if $\mathcal{L}$ contains the set of singletons $\{\{v\} \mid v \in V(H)\}$ then $\#\textsc{Hom}(H,\mathcal{L})$ is *self-reducible*. So the technique of Jerrum, Valiant and Vazirani [32] reduces the problem of approximately sampling homomorphisms with lists in $\mathcal{L}$ to the problem of approximately counting them. The original notion of self-reducibilty, due to Schnorr [42], relies on careful encodings of instances, so we use instead the more general *self-partitionability* notion of Dyer and Greenhill. Dyer and Greenhill show [8] that the technique of Jerrum, Valiant and Vazirani applies to every self-partitionable problem. Thus, we get the following lemma.

Lemma 3.2. *Let H be a graph and $\mathcal{L} \subseteq 2^{V(H)}$ such that $\{\{v\} \mid v \in V(H)\} \subseteq \mathcal{L}$. There is an algorithm $\textsc{SampleHom}_{H,\mathcal{L}}$ which uses oracle access to a RAS for $\#\textsc{Hom}(H,\mathcal{L})$ and has the following properties.*

- *It is given an input (G,S) to $\#\textsc{Hom}(H,\mathcal{L})$ together with an accuracy parameter $\varepsilon \in (0,1)$.*
- *The distribution D of its outputs satisfies $\mathrm{d_{TV}}\big(D, \mathrm{Uni}(\mathcal{H}((G,\mathrm{S}),H))\big) \le \varepsilon$.*
- *Its running time is bounded by a polynomial in $\log\varepsilon^{-1}$ and the number of vertices of G.*

Proof. Rather than repeating the (lengthy) formal definition of *self-partitionability* from [8], we state the (self-evident) relevant properties of $\#\textsc{Hom}(H,\mathcal{L})$ which imply that $\#\textsc{Hom}(H,\mathcal{L})$ is self-partitionable. The lemma follows immediately from [8].

Let (G,S) be an input to $\#\textsc{Hom}(H,\mathcal{L})$. If $v \in V(G)$ and $s \in S_v$, then let $\mathrm{S}^{v\to s} = \{S_u^{v\to s} \mid u \in V(G)\}$ be defined as follows.

$$S_u^{v\to s} = \begin{cases} \{s\} & \text{if } u = v \\ S_u & \text{otherwise.} \end{cases}$$

Note that $(G,\mathrm{S}^{v\to s})$ is a valid input to $\#\textsc{Hom}(H,\mathcal{L})$ as $\{\{v\} \mid v \in V(H)\} \subseteq \mathcal{L}$.

The relevant properties are

(1) If for all $v \in V(G)$ we have $S_v \in \{\{u\} \mid u \in V(H)\}$ then the function τ which maps each vertex $v \in V(G)$ to the single element in the corresponding list S_v is the only mapping from G to H that respects the lists. It is then easy to check whether τ is a homomorphism. Therefore, computing $N\left((G,\mathrm{S})\to H\right)$ and sampling from the set of list homomorphisms from (G,S) to H is trivial.

(2) If $v \in V(G)$ then

$$\mathcal{H}((G,\mathrm{S}),H) = \bigcup_{s \in S_v} \mathcal{H}((G,\mathrm{S}^{v\to s}),H). \tag{7}$$

Note that the right-hand-side of (7) is a union of disjoint sets since all of the homomorphisms in $\mathcal{H}((G,\mathrm{S}^{v\to s}),H)$ map v to s.

These properties imply that #Hom$(H,\mathcal{L})$ is self-partitionable in the sense of Dyer and Greenhill, thus the lemma follows from the technique of Jerrum, Valiant and Vazirani, as demonstrated in [8].

This completes the proof of the lemma, but for the reader who wants to relate the above properties to the notation of Dyer and Greenhill, we take the size of an instance (G,S) to be $|\{v \in V(G) \mid |S_v| > 1\}|$. The set of smaller instances $\Xi(G,\mathrm{S})$ considered in [8] can be constructed by fixing any $v \in V(G)$ with $|S_v| > 1$ and then setting $\Xi(G,\mathrm{S}) = \{(G,\mathrm{S}^{v\to s}) \mid s \in S_v\}$. The functions k_ξ mentioned in [8] can all be taken to be constant functions, with output 1. The injection $\phi_{(G,\mathrm{S}^{v\to s})}$ is the identity. Finally $W((G,\mathrm{S}),\tau)$ is just the indicator function that is 1 if τ is a homomorphism from (G,S) to H and 0 otherwise. □

Our first goal is Corollary 3.4 which is an AP-reduction from #Comp(H) to #Ret(H). The reduction uses a Monte Carlo Algorithm (Algorithm 1). The algorithm is presented more generally, with lists, so that we can also use it in the reductions of Corollaries 3.5, 3.6 and 3.7. The following observation provides the basis for the algorithm. Let H be a graph, G be an irreflexive graph and S be a corresponding set of lists. If there is a compaction from (G,S) to H then there exists a set $U \subseteq V(G)$ with $|U| \le |V(H)| + 2|E(H)|$ and a compaction τ from $G[U]$ to H. Accordingly, we define

$$\begin{aligned} T_{G,\mathrm{S}} = \{(U,\tau) \mid\ & U \subseteq V(G), |U| \le |V(H)| + 2|E(H)|, \\ & \tau \text{ is a compaction from } G[U] \text{ to } H \text{ such that } \forall u \in U, \tau(u) \in S_u\} \end{aligned} \tag{8}$$

and $t_{G,\mathrm{S}} = \left|T_{G,\mathrm{S}}\right|$. Let $(U_i,\tau_i)_{i\in[t_{G,\mathrm{S}}]}$ be an arbitrary indexing of the elements of $T_{G,\mathrm{S}}$. For $i \in [t_{G,\mathrm{S}}]$ we define

$$\Omega_{G,\mathrm{S},i} = \left\{\sigma \in \mathcal{H}((G,\mathrm{S}),H) \mid \quad \sigma|_{U_i} = \tau_i\right\}, \tag{9}$$

$$\Omega^+_{G,\mathrm{S}} = \left\{(i,\sigma) \mid i \in [t_{G,\mathrm{S}}] \text{ and } \sigma \in \Omega_{G,\mathrm{S},i}\right\}, \text{and} \tag{10}$$

$$\Omega_{G,\mathrm{S}} = \left\{(i,\sigma) \in \Omega^+_{G,\mathrm{S}} \mid \sigma \notin \bigcup_{k=1}^{i-1} \Omega_{G,\mathrm{S},k}\right\}. \tag{11}$$

Note that $\left|\Omega^+_{G,\mathrm{S}}\right| = \sum_{i\in[t_{G,\mathrm{S}}]}\left|\Omega_{G,\mathrm{S},i}\right|$. As every element of a set $\Omega_{G,\mathrm{S},i}$ is a compaction from (G,S) to H and every such compaction is contained in a set $\Omega_{G,\mathrm{S},i}$, we have

$$\begin{aligned} \left|\Omega_{G,\mathrm{S}}\right| &= \left|\bigcup_{i\in[t_{G,\mathrm{S}}]} \Omega_{G,\mathrm{S},i}\right| = \left|\{\sigma \in \mathcal{H}((G,\mathrm{S}),H) \mid \exists i \in [t_{G,\mathrm{S}}] \text{ such that } \sigma \in \Omega_{G,\mathrm{S},i}\}\right| \\ &= N^{\mathrm{comp}}\big((G,\mathrm{S}) \to H\big). \end{aligned}$$

It is clear from the definitions that $|\Omega_{G,\mathrm{S}}| \ge |\Omega^+_{G,\mathrm{S}}|/t_{G,\mathrm{S}}$. Thus,

$$N^{\mathrm{comp}}\big((G,\mathrm{S}) \to H\big) = \left|\Omega_{G,\mathrm{S}}\right| \ge \frac{\left|\Omega^+_{G,\mathrm{S}}\right|}{t_{G,\mathrm{S}}}. \tag{12}$$

Intuitively, for some fixed graph H and $\mathcal{L} \subseteq 2^{V(H)}$ we use this lower bound to construct a Monte Carlo algorithm (Algorithm 1) in the style of [38, Algorithm 11.2], which approximately samples from $\Omega^+_{G,\mathrm{S}}$ to approximately compute $\left|\Omega_{G,\mathrm{S}}\right| = N^{\mathrm{comp}}\big((G,\mathrm{S}) \to H\big)$. To this end the algorithm relies on access to a RAS oracle for #Hom$(H,\mathcal{L}^*)$ where $\mathcal{L}^* = \mathcal{L} \cup \{\{v\} \mid v \in V(H)\}$.

Algorithm 1 Approximate Computation of $|\Omega_{G,\mathrm{S}}|$. Let H be a fixed graph, $\mathcal{L} \subseteq 2^{V(H)}$ and $\mathcal{L}^* = \mathcal{L} \cup \{\{v\} \mid v \in V(H)\}$. Then $\textsc{CountHom}_{H,\mathcal{L}^*}$ and $\textsc{SampleHom}_{H,\mathcal{L}^*}$ are the routines from Lemma 3.1 and 3.2, respectively. Let $T_{G,\mathrm{S}}$, $t_{G,\mathrm{S}}$, $\Omega_{G,\mathrm{S},i}$, $\Omega^+_{G,\mathrm{S}}$ and $\Omega_{G,\mathrm{S}}$ be defined as in Equations (8)-(11). Note that (U_i, τ_i) is the i'th element of $T_{G,\mathrm{S}}$.

```
Input: Irreflexive graph $G$ with lists $\mathrm{S} = \{S_v \in \mathcal{L} \mid v \in V(G)\}$ and $\varepsilon,\ \delta \in (0,1)$.
  if $t_{G,\mathrm{S}} = 0$
      $Y = 0$.
  else
      $\varepsilon' = \frac{\varepsilon}{12}, \delta' = \frac{\delta}{2}, \delta'' = \frac{\delta'}{t_{G,\mathrm{S}}}$.
      for $i = 1, \dots, t_{G,\mathrm{S}}$
          For all $v \in V(G)$, if $v \in U_i$, set $S_v^i = \{\tau_i(v)\}$, otherwise set $S_v^i = S_v$.
          $\mathrm{S}^i = \{S_v^i \mid v \in V(G)\}$
          $\omega_i = \textsc{CountHom}_{H,\mathcal{L}^*}(G, \mathrm{S}^i, \varepsilon', \delta'')$.
      $\omega = \sum_{i=1}^{t_{G,\mathrm{S}}} \omega_i$.
      $m = \left\lceil 6 t_{G,\mathrm{S}} \cdot \frac{\ln(2/\delta')}{\varepsilon'^2} \right\rceil$.
      for $j = 1, \dots, m$
          Choose $i \in [t_{G,\mathrm{S}}]$ with probability $\frac{\omega_i}{\omega}$.
          $\sigma_j = \textsc{SampleHom}_{H,\mathcal{L}^*}(G, \mathrm{S}^i, \varepsilon'/(2|V(H)|^n))$.
          Let $X_j$ be 1 in the event $(i, \sigma_j) \in \Omega_{G,\mathrm{S}}$ and 0 otherwise.
      $Y = \frac{\omega}{m} \sum_{j=1}^{m} X_j$.
Output: $Y$
```

Lemma 3.3. *Algorithm 1 returns an (ε, δ)-approximation of $|\Omega_{G,\mathrm{S}}|$ if it has access to a RAS oracle for #Hom$(H, \mathcal{L}^*)$ and every list in* S *is an element of $\mathcal{L}$. For fixed δ, the algorithm runs in time polynomial in $n = |V(G)|$ and ε^{-1}.*

Proof. First note that given oracle access to a RAS for #Hom$(H, \mathcal{L}^*)$, the routines $\textsc{CountHom}_{H,\mathcal{L}^*}$ and $\textsc{SampleHom}_{H,\mathcal{L}^*}$ exist as shown in Lemmas 3.1 and 3.2 (by definition $\mathcal{L}^*$ contains $\{\{v\} \mid v \in V(H)\}$). Furthermore, the input to these routines is valid: A list $S_v^i \in \mathrm{S}^i$ is either of the form $\{\tau_i(v)\}$ or otherwise $S_v^i = S_v \in \mathcal{L}$. Therefore, in general, $S_v^i \in \mathcal{L}^*$. Thus Algorithm 1 is well-defined.

Next we show that the runtime condition is met. Assume δ to be fixed. Note that we can determine $T_{G,\mathrm{S}}$ exactly by enumerating all possible assignments of at most $|V(H)| + 2|E(H)|$ vertices of G to the vertices of H and checking whether the resulting assignment is a compaction. Checking can be done in polynomial time and $t_{G,\mathrm{S}} = |T_{G,\mathrm{S}}| \le \sum_{k=1}^{|V(H)|+2|E(H)|} n^k \in \mathrm{poly}(n)$. It follows that $m \in \mathrm{poly}(n, \varepsilon^{-1})$. The runtime of the routine $\textsc{CountHom}_{H,\mathcal{L}^*}(G, \mathrm{S}^i, \varepsilon', \delta'')$ is in $\mathrm{poly}(n, 1/\varepsilon')$ by Lemma 3.1. Finally, the runtime of $\textsc{SampleHom}_{H,\mathcal{L}^*}(G, \mathrm{S}^i, \varepsilon'/(2|V(H)|^n))$ is in $\mathrm{poly}(n, \log(1/\varepsilon'))$ by Lemma 3.2. It is essential here that the runtime of $\textsc{SampleHom}_{H,\mathcal{L}^*}$ has logarithmic dependence on the precision parameter as the precision we use is $\varepsilon'/(2|V(H)|^n)$, which is exponential in n.

If $|T_{G,\mathrm{S}}| = t_{G,\mathrm{S}} = 0$ then $|\Omega_{G,\mathrm{S}}| = 0$ and the algorithm returns an exact solution. To prove the correctness of the algorithm it remains to show that otherwise it is an (ε, δ)-approximation.

Note that by the definition of the S^i in the first part of the algorithm, $\Omega_{G,\mathrm{S},i} = \mathcal{H}((G, \mathrm{S}^i), H)$. Then, by Lemma 3.1 and the definition of δ'', $\textsc{CountHom}_{H,\mathcal{L}^*}(G, \mathrm{S}^i, \varepsilon', \delta'')$ returns a number ω_i with $\Pr(e^{-\varepsilon'}|\Omega_{G,\mathrm{S},i}| \le \omega_i \le e^{\varepsilon'}|\Omega_{G,\mathrm{S},i}|) \ge 1 - \delta'/t_{G,\mathrm{S}}$. By the union bound, with probability of at least $1-\delta'$, we have

$$e^{-\varepsilon'}\left|\Omega_{G,\mathrm{S},i}\right| \leq \omega_i \leq e^{\varepsilon'}\left|\Omega_{G,\mathrm{S},i}\right| \qquad (\forall i \in [t_{G,\mathrm{S}}]). \tag{13}$$

The following two subclaims are based on this assumption. Let $p = \left|\Omega_{G,\mathrm{S}}\right| / \left|\Omega^+_{G,\mathrm{S}}\right|$.

Subclaim 1: Assume that (13) holds. Then for all $j \in [m]$ we have $X_j \sim \mathrm{Be}(p')$ where $e^{-3\varepsilon'}p \leq p' \leq e^{3\varepsilon'}p$.

Proof of Subclaim 1: Consider fixed $\omega_1, \dots, \omega_{t_{G,\mathrm{S}}}$ satisfying (13). Note that the distribution of (i, σ_j), conditioned on these, does not depend on the index j. Let D be the distribution (conditioned on $\omega_1, \dots, \omega_{t_{G,\mathrm{S}}}$) such that for all $j \in [m]$ we have $(i, \sigma_j) \sim D$. By Lemma 3.2 and the fact that $\Omega_{G,\mathrm{S},k} = \mathcal{H}((G, \mathrm{S}^k), H)$ we have

$$D((k,\sigma)) = \Pr(\sigma_j = \sigma \mid i = k) \cdot \Pr(i = k) \leq \left(\frac{1}{\left|\Omega_{G,\mathrm{S},k}\right|} + \frac{\varepsilon'}{2|V(H)|^n}\right) \cdot \Pr(i = k)$$

First using $\left|\Omega_{G,\mathrm{S},k}\right| \leq |V(H)|^n$ and then using Observation 1.13 it follows

$$D((k,\sigma)) \leq \left(1 + \frac{\varepsilon'}{2}\right)\frac{1}{\left|\Omega_{G,\mathrm{S},k}\right|} \cdot \Pr(i = k) \leq e^{\varepsilon'}\frac{1}{\left|\Omega_{G,\mathrm{S},k}\right|} \cdot \Pr(i = k) = e^{\varepsilon'}\frac{1}{\left|\Omega_{G,\mathrm{S},k}\right|} \cdot \frac{\omega_k}{\sum_{i=1}^{t_{G,\mathrm{S}}} \omega_i}.$$

Using the assumption of this subclaim, we obtain

$$D((k,\sigma)) \leq e^{\varepsilon'}\frac{1}{\left|\Omega_{G,\mathrm{S},k}\right|} \cdot e^{2\varepsilon'}\frac{\left|\Omega_{G,\mathrm{S},k}\right|}{\sum_{i\in[t_{G,\mathrm{S}}]}\left|\Omega_{G,\mathrm{S},i}\right|} = e^{3\varepsilon'}\frac{1}{\left|\Omega^+_{G,\mathrm{S}}\right|}$$

and thus

$$p' = \Pr(X_j = 1) = \sum_{(i,\sigma)\in\Omega_{G,\mathrm{S}}} D((i,\sigma)) \leq e^{3\varepsilon'} \sum_{(i,\sigma)\in\Omega_{G,\mathrm{S}}} \frac{1}{\left|\Omega^+_{G,\mathrm{S}}\right|} = e^{3\varepsilon'}p.$$

Analogously we obtain the lower bound $e^{-3\varepsilon'}p \leq p'$. **(End of the proof of Subclaim 1.)**

Subclaim 2: Assume that (13) holds. Then $e^{-4\varepsilon'}\left|\Omega_{G,\mathrm{S}}\right| \leq \mathbf{E}[Y] \leq e^{4\varepsilon'}\left|\Omega_{G,\mathrm{S}}\right|$.

Proof of Subclaim 2: Consider fixed $\omega_1, \dots, \omega_{t_{G,\mathrm{S}}}$ satisfying (13). Conditioned on these, we have $X_j \sim \mathrm{Be}(p')$ and

$$\mathbf{E}[Y] = \frac{\sum_{i\in[t_{G,\mathrm{S}}]}\omega_i}{m} \sum_{j\in[m]} \mathbf{E}[X_j] = \sum_{i\in[t_{G,\mathrm{S}}]} \omega_i \cdot p'.$$

We now use (13) as well as Subclaim 1 to obtain

$$\mathbf{E}[Y] \leq e^{\varepsilon'} \sum_{i\in[t_{G,\mathrm{S}}]} \left|\Omega_{G,\mathrm{S},i}\right| \cdot e^{3\varepsilon'}p = e^{4\varepsilon'}\left|\Omega_{G,\mathrm{S}}\right|$$

and

$$\mathbf{E}[Y] \geq e^{-\varepsilon'} \sum_{i\in[t_{G,\mathrm{S}}]} \left|\Omega_{G,\mathrm{S},i}\right| \cdot e^{-3\varepsilon'}p = e^{-4\varepsilon'}\left|\Omega_{G,\mathrm{S}}\right|.$$

(End of the proof of Subclaim 2.)

Next we show that, conditioned on computing ω_i's that satisfy (13), the number of samples m is sufficiently large. First, by Subclaim 1, $X_1, \dots, X_m$ are independent indicator random variables that have distribution $\mathrm{Be}(p')$ and expected value p'. By Subclaim 1 and Observation 1.13 we have

$$(1 - 6\varepsilon')p \leq e^{-3\varepsilon'}p \leq p' \leq e^{3\varepsilon'}p \leq (1 + 6\varepsilon')p.$$

From the definition of ε' it follows that $|p' - p| \leq 6\varepsilon' p \leq \varepsilon p/2$ and consequently $p' \geq p/2$. Using this fact and taking into account that by Equation (12) we have $t_{G,\mathrm{S}} \geq 1/p$, it follows that

$$m = \left\lceil 6t_{G,\mathrm{S}} \cdot \frac{\ln(2/\delta')}{\varepsilon'^2} \right\rceil \geq 6\frac{\ln(2/\delta')}{\varepsilon'^2 p} \geq 3\frac{\ln(2/\delta')}{\varepsilon'^2 p'}.$$

Thus, we can use [38, Theorem 11.1] to obtain $\Pr(|Y - \mathbf{E}[Y]| \geq \varepsilon' \mathbf{E}[Y]) \leq \delta'$ which is conditioned on the fact that (13) holds. Now taking into account the fact that, with probability at least $1 - \delta'$, $\omega_1, \ldots, \omega_{t_{G,\mathrm{S}}}$ satisfy (13), we have shown that, with probability of at least $(1 - \delta')^2 \geq 1 - \delta$, we have

$$|Y - \mathbf{E}[Y]| \leq \varepsilon' \mathbf{E}[Y] = \frac{\varepsilon}{12}\mathbf{E}[Y].$$

By Subclaim 2 and Observation 1.13 we also know that

$$\left|\mathbf{E}[Y] - \left|\Omega_{G,\mathrm{S}}\right|\right| \leq 8\varepsilon' \left|\Omega_{G,\mathrm{S}}\right| = \frac{2\varepsilon}{3}\left|\Omega_{G,\mathrm{S}}\right|.$$

Summarising, with probability of at least $1 - \delta$, we have

$$\begin{aligned}\left|Y - \left|\Omega_{G,\mathrm{S}}\right|\right| &\leq |Y - \mathbf{E}[Y]| + \left|\mathbf{E}[Y] - \left|\Omega_{G,\mathrm{S}}\right|\right| \leq \frac{\varepsilon}{12}\mathbf{E}[Y] + \frac{2\varepsilon}{3}\left|\Omega_{G,\mathrm{S}}\right| \\ &\leq \frac{\varepsilon}{12}\left(\left|\Omega_{G,\mathrm{S}}\right| + \frac{2\varepsilon}{3}\left|\Omega_{G,\mathrm{S}}\right|\right) + \frac{2\varepsilon}{3}\left|\Omega_{G,\mathrm{S}}\right| \leq \varepsilon\left|\Omega_{G,\mathrm{S}}\right|.\end{aligned}$$

Hence, Y is an (ε, δ)-approximation of $\left|\Omega_{G,\mathrm{S}}\right|$. □

Corollary 3.4. *Let H be a graph. Then #Comp(H) $\leq_{\mathrm{AP}}$ #Ret(H).*

Proof. Let $\mathcal{L} = \{V(H)\}$. Then the problem #Hom($H, \mathcal{L}^*$) is identical to #Ret(H). Furthermore, given an irreflexive graph G and a set $\mathrm{S} = \{S_v \in \mathcal{L} \mid v \in G\}$, for this choice of $\mathcal{L}$ it holds that $N^{\mathrm{comp}}\big((G,\mathrm{S}) \to H\big) = N^{\mathrm{comp}}\big(G \to H\big)$.

Then, by Lemma 3.3, given a RAS oracle for #Ret(H), Algorithm 1 computes an (ε, δ)-approximation of $\left|\Omega_{G,\mathrm{S}}\right| = N^{\mathrm{comp}}\big((G,\mathrm{S}) \to H\big) = N^{\mathrm{comp}}\big(G \to H\big)$. If we choose $\delta = 1/4$ then the algorithm is an FPRAS for #Comp(H). □

Corollary 3.5. *Let H be a graph. Then #LComp(H) $\leq_{\mathrm{AP}}$ #LHom(H).*

Proof. Let $\mathcal{L} = 2^{V(H)}$. Then the problem #Hom($H, \mathcal{L}^*$) = #Hom($H, \mathcal{L}$) is identical to #LHom(H).

From Lemma 3.3 it follows that given a RAS oracle for #LHom(H), Algorithm 1 returns an (ε, δ)-approximation of $\left|\Omega_{G,\mathrm{S}}\right| = N^{\mathrm{comp}}\big((G,\mathrm{S}) \to H\big)$. In particular, as $\mathcal{L}$ is unrestricted, it does so for any valid input (G, S) of the problem #LComp(H). Thus, if we choose $\delta = 1/4$, the algorithm is an FPRAS for #LComp(H). □

To obtain Corollaries 3.4 and 3.5, the only property of compactions we use is the fact that for every compaction from G to H there exists a preimage U of polynomial size, i.e. a set $U \subseteq V(G)$ with $|U| \leq |V(H)| + 2|E(H)|$ and a compaction τ from $G[U]$ to H. (This is used in Equation (8).)

Similarly, for every surjective homomorphism from G to H there exists a set $U \subseteq V(G)$ with $|U| = |V(H)|$ such that there exists a surjective homomorphism τ from $G[U]$ to H. If we substitute

$$\begin{aligned}T_{G,\mathrm{S}} = \{(U, \tau) \mid\ & U \subseteq V(G), |U| = |V(H)|, \\ & \tau \text{ is a surjective homomorphism from } G[U] \text{ to } H \text{ such that } \forall u \in U, \tau(u) \in S_u\}\end{aligned}$$

for Equation (8), Lemma 3.3 still holds and now $\left|\Omega_{G,\mathrm{S}}\right| = N^{\mathrm{sur}}\big((G,\mathrm{S}) \to H\big)$.

Therefore, analogously to the previous two corollaries we obtain the following.

Corollary 3.6. *Let H be a graph. Then #SHom(H) $\leq_{\mathrm{AP}}$ #Ret(H).*

Corollary 3.7. *Let H be a graph. Then #LSHom(H) $\leq_{\mathrm{AP}}$ #LHom(H).*

Corollaries 3.4 and 3.6 together constitute Theorem 1.5.

3.2 Additional Reductions and Consequences

The following simple reductions complete our current knowledge of the complexity landscape given in Figure 1.

Lemma 3.8. *Let H be a graph. Then #LHom(H) $\leq_{\mathrm{AP}}$ #LSHom(H) and #LHom(H) $\leq_{\mathrm{AP}}$ #LComp(H).*

Proof. Let $v_1, \ldots, v_q$ be the vertices of H and let (G, S) be an input to #LHom(H). Further, let H' be a copy of H and let $u_1, \ldots, u_q$ be the vertices of H' ordered in the same way as $v_1, \ldots, v_q$. For $i \in [q]$ let $S_{u_i} = \{v_i\}$ and let $\mathrm{S}' = \mathrm{S} \cup \{S_{u_i} : i \in [q]\}$. Let G' be the disjoint union of G and H'. Then $N\big((G, \mathrm{S}) \to H\big) = N^{\mathrm{sur}}\big((G', \mathrm{S}') \to H\big) = N^{\mathrm{comp}}\big((G', \mathrm{S}') \to H\big)$. □

From Corollaries 3.5 and 3.7 as well as Lemma 3.8 we immediately obtain Theorem 1.6 which we restate at this point.

Theorem 1.6. *Let H be a graph. Then #LSHom(H) $\equiv_{\mathrm{AP}}$ #LHom(H) and #LComp(H) $\equiv_{\mathrm{AP}}$ #LHom(H).*

ACKNOWLEDGMENTS

A preliminary version of this paper (without the proofs) appeared in the proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019. This preliminary version stated a trichotomy for approximately counting retractions to trees rather than (more generally) to graphs of girth at least 5.

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828 and under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). Jacob Focke has received funding from the Engineering and Physical Sciences Research Council (grant ref: EP/M508111/1). Stanislav Živný was supported by a Royal Society University Research Fellowship. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

REFERENCES

[1] M. Bíró, M. Hujter, and Zs. Tuza. 1992. Precoloring extension. I. Interval graphs. *Discrete Math.* 100, 1-3 (1992), 267–279. DOI: http://dx.doi.org/10.1016/0012-365X(92)90646-W Special volume to mark the centennial of Julius Petersen's "Die Theorie der regulären Graphs", Part I.

[2] Manuel Bodirsky, Jan Kára, and Barnaby Martin. 2012. The complexity of surjective homomorphism problems—a survey. *Discrete Appl. Math.* 160, 12 (2012), 1680–1690. DOI: http://dx.doi.org/10.1016/j.dam.2012.03.029

[3] Christian Borgs, Jennifer Chayes, László Lovász, Vera T. Sós, and Katalin Vesztergombi. 2006. Counting graph homomorphisms. In *Topics in discrete mathematics*. Algorithms Combin., Vol. 26. Springer, Berlin, 315–371. DOI: http://dx.doi.org/10.1007/3-540-33700-8_18

[4] Andrei A. Bulatov. 2017. A Dichotomy Theorem for Nonuniform CSPs. In *58th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2017*. IEEE Computer Soc., Los Alamitos, CA, 319–330.

[5] Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. 2004. The Relative Complexity of Approximate Counting Problems. *Algorithmica* 38, 3 (2004), 471–500. DOI: http://dx.doi.org/10.1007/s00453-003-1073-y

[6] Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. 2004. Counting and sampling H-colourings. *Inform. and Comput.* 189, 1 (2004), 1–16. DOI: http://dx.doi.org/10.1016/j.ic.2003.09.001

[7] Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. 2010. An approximation trichotomy for Boolean ♯CSP. *J. Comput. System Sci.* 76, 3-4 (2010), 267–277. DOI: http://dx.doi.org/10.1016/j.jcss.2009.08.003

[8] Martin Dyer and Catherine Greenhill. 1999. Random walks on combinatorial objects. In *Surveys in combinatorics, 1999 (Canterbury) (London Math. Soc. Lecture Note Ser.)*, Vol. 267. Cambridge Univ. Press, Cambridge, 101–136.

[9] Martin E. Dyer and Catherine S. Greenhill. 2000. The complexity of counting graph homomorphisms. *Random Struct. Algorithms* 17, 3-4 (2000), 260–289.

[10] Tomas Feder and Pavol Hell. 1998. List homomorphisms to reflexive graphs. *J. Combin. Theory Ser. B* 72, 2 (1998), 236–250. DOI:http://dx.doi.org/10.1006/jctb.1997.1812

[11] Tomas Feder, Pavol Hell, and Jing Huang. 1999. List Homomorphisms and Circular Arc Graphs. *Combinatorica* 19, 4 (1999), 487–505. DOI:http://dx.doi.org/10.1007/s004939970003

[12] Tomas Feder, Pavol Hell, and Jing Huang. 2009. Extension problems with degree bounds. *Discrete Appl. Math.* 157, 7 (2009), 1592–1599. DOI:http://dx.doi.org/10.1016/j.dam.2008.04.006

[13] Tomás Feder, Pavol Hell, Peter Jonsson, Andrei Krokhin, and Gustav Nordh. 2010. Retractions to pseudoforests. *SIAM J. Discrete Math.* 24, 1 (2010), 101–112. DOI:http://dx.doi.org/10.1137/080738866

[14] Jacob Focke, Leslie Ann Goldberg, and Stanislav Živný. 2017. The Complexity of Counting Surjective Homomorphisms and Compactions. *CoRR* abs/1706.08786 (2017). http://arxiv.org/abs/1706.08786 A preliminary version of this work appeared in the Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1772-1781.

[15] Andreas Galanis, Leslie Ann Goldberg, and Mark Jerrum. 2016. Approximately counting H-colorings is #BIS-hard. *SIAM J. Comput.* 45, 3 (2016), 680–711. DOI:http://dx.doi.org/10.1137/15M1020551

[16] Andreas Galanis, Leslie Ann Goldberg, and Mark Jerrum. 2017. A Complexity Trichotomy for Approximately Counting List H-Colorings. *ACM Trans. Comput. Theory* 9, 2 (2017), Art. 9, 22. DOI:http://dx.doi.org/10.1145/3037381

[17] Andreas Galanis, Daniel Štefankovič, Eric Vigoda, and Linji Yang. 2016. Ferromagnetic Potts Model: Refined #BIS-Hardness and Related Results. *SIAM J. Comput.* 45, 6 (2016), 2004–2065. DOI:http://dx.doi.org/10.1137/140997580

[18] Leslie Ann Goldberg and Mark Jerrum. 2012. Approximating the Partition Function of the Ferromagnetic Potts Model. *J. ACM* 59, 5 (2012), Art. 25, 31. DOI:http://dx.doi.org/10.1145/2371656.2371660

[19] Leslie Ann Goldberg and Mark Jerrum. 2014. The Complexity of Approximately Counting Tree Homomorphisms. *ACM Trans. Comput. Theory* 6, 2 (2014), Art. 8, 31. DOI:http://dx.doi.org/10.1145/2600917

[20] Leslie Ann Goldberg, Steven Kelk, and Mike Paterson. 2004. The Complexity of Choosing an H-Coloring (Nearly) Uniformly at Random. *SIAM J. Comput.* 33, 2 (2004), 416–432. DOI:http://dx.doi.org/10.1137/S0097539702408363

[21] Petr A. Golovach, Matthew Johnson, Barnaby Martin, Daniël Paulusma, and Anthony Stewart. 2017. Surjective H-Colouring: New Hardness Results. In *Unveiling dynamics and complexity*. Lecture Notes in Comput. Sci., Vol. 10307. Springer, Cham, 270–281.

[22] Petr A. Golovach, Bernard Lidický, Barnaby Martin, and Daniël Paulusma. 2012a. Finding vertex-surjective graph homomorphisms. *Acta Inform.* 49, 6 (2012), 381–394. DOI:http://dx.doi.org/10.1007/s00236-012-0164-0

[23] Petr A. Golovach, Daniël Paulusma, and Jian Song. 2012b. Computing vertex-surjective homomorphisms to partially reflexive trees. *Theoret. Comput. Sci.* 457 (2012), 86–100. DOI:http://dx.doi.org/10.1016/j.tcs.2012.06.039

[24] Frank Harary and Allen Schwenk. 1971. Trees with Hamiltonian square. *Mathematika* 18 (1971), 138–140. DOI:http://dx.doi.org/10.1112/S0025579300008494

[25] Pavol Hell. 1973. *Retractions des graphes.* ProQuest LLC, Ann Arbor, MI. (no paging) pages. http://ezproxy-prd.bodleian.ox.ac.uk:2175/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqdiss&rft_dat=xri:pqdiss:0289365 Thesis (Ph.D.)–Universite de Montreal (Canada).

[26] Pavol Hell. 1974. Absolute retracts in graphs. (1974), 291–301. Lecture Notes in Math., Vol. 406.

[27] Pavol Hell and Jaroslav Nešetřil. 1990. On the complexity of H-coloring. *J. Combin. Theory Ser. B* 48, 1 (1990), 92–110. DOI:http://dx.doi.org/10.1016/0095-8956(90)90132-J

[28] Pavol Hell and Jaroslav Nešetřil. 2004a. Counting list homomorphisms for graphs with bounded degrees. In *Graphs, morphisms and statistical physics*. DIMACS Ser. Discrete Math. Theoret. Comput. Sci., Vol. 63. Amer. Math. Soc., Providence, RI, 105–112. DOI:http://dx.doi.org/10.1093/acprof:oso/9780198528173.001.0001

[29] Pavol Hell and Jaroslav Nešetřil. 2004b. *Graphs and Homomorphisms*. Oxford Lecture Series in Mathematics and its Applications, Vol. 28. Oxford University Press, Oxford. xii+244 pages. DOI:http://dx.doi.org/10.1093/acprof:oso/9780198528173.001.0001

[30] Pavol Hell and Jaroslav Nešetřil. 2008. Colouring, constraint satisfaction, and complexity. *Computer Science Review* 2, 3 (2008), 143–163. DOI:http://dx.doi.org/10.1016/j.cosrev.2008.10.003

[31] Pavol Hell and Ivan Rival. 1987. Absolute retracts and varieties of reflexive graphs. *Canad. J. Math.* 39, 3 (1987), 544–567. DOI:http://dx.doi.org/10.4153/CJM-1987-025-1

[32] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. 1986. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoret. Comput. Sci.* 43, 2-3 (1986), 169–188. DOI:http://dx.doi.org/10.1016/0304-3975(86)90174-X

[33] Steven Kelk. 2003. *On the relative complexity of approximately counting H-colourings.* Ph.D. Dissertation. Warwick University.

[34] Ekkehard G. Köhler. 1999. *Graphs without asteroidal triples.* Ph.D. Dissertation. Technische Universität Berlin.

[35] Jan Kratochvíl and András Sebő. 1997. Coloring precolored perfect graphs. *J. Graph Theory* 25, 3 (1997), 207–215. DOI: http://dx.doi.org/10.1002/(SICI)1097-0118(199707)25:3<207::AID-JGT4>3.0.CO;2-P

[36] Barnaby Martin and Daniël Paulusma. 2015. The computational complexity of disconnected cut and $2K_2$-partition. *J. Combin. Theory Ser. B* 111 (2015), 17–37. DOI: http://dx.doi.org/10.1016/j.jctb.2014.09.002

[37] Dániel Marx. 2006. Parameterized coloring problems on chordal graphs. *Theoret. Comput. Sci.* 351, 3 (2006), 407–424. DOI: http://dx.doi.org/10.1016/j.tcs.2005.10.008

[38] Michael Mitzenmacher and Eli Upfal. 2017. *Probability and Computing* (second ed.). Cambridge University Press, Cambridge. xx+467 pages. Randomization and Probabilistic techniques in Algorithms and Data Analysis.

[39] Erwin Pesch. 1988. *Retracts of graphs*. Mathematical Systems in Economics, Vol. 110. Athenäum Verlag GmbH, Frankfurt am Main. xii+220 pages.

[40] R. B. Potts. 1952. Some generalized order-disorder transformations. *Proc. Cambridge Philos. Soc.* 48 (1952), 106–109.

[41] Wolfgang M. Schmidt. 1991. *Diophantine approximations and Diophantine equations*. Lecture Notes in Mathematics, Vol. 1467. Springer-Verlag, Berlin. viii+217 pages. DOI: http://dx.doi.org/10.1007/BFb0098246

[42] Claus-Peter Schnorr. 1976. Optimal Algorithms for Self-Reducible Problems. In *Third International Colloquium on Automata, Languages and Programming, University of Edinburgh, UK, July 20-23, 1976*, S. Michaelson and Robin Milner (Eds.). Edinburgh University Press, 322–337.

[43] Zsolt Tuza. 1997. Graph colorings with local constraints—a survey. *Discuss. Math. Graph Theory* 17, 2 (1997), 161–228. DOI: http://dx.doi.org/10.7151/dmgt.1049

[44] Narayan Vikas. 2002. Computational Complexity of Compaction to Reflexive Cycles. *SIAM J. Comput.* 32, 1 (2002/03), 253–280. DOI: http://dx.doi.org/10.1137/S0097539701383522

[45] Narayan Vikas. 2004a. Compaction, Retraction, and Constraint Satisfaction. *SIAM J. Comput.* 33, 4 (2004), 761–782. DOI: http://dx.doi.org/10.1137/S0097539701397801

[46] Narayan Vikas. 2004b. Computational complexity of compaction to irreflexive cycles. *J. Comput. System Sci.* 68, 3 (2004), 473–496. DOI: http://dx.doi.org/10.1016/S0022-0000(03)00034-5

[47] Narayan Vikas. 2005. A complete and equal computational complexity classification of compaction and retraction to all graphs with at most four vertices and some general results. *J. Comput. System Sci.* 71, 4 (2005), 406–439. DOI: http://dx.doi.org/10.1016/j.jcss.2004.07.003

[48] Narayan Vikas. 2013. Algorithms for Partition of Some Class of Graphs under Compaction and Vertex-Compaction. *Algorithmica* 67, 2 (2013), 180–206. DOI: http://dx.doi.org/10.1007/s00453-012-9720-9

[49] Narayan Vikas. 2017. Computational complexity of graph partition under vertex-compaction to an irreflexive hexagon. In *42nd International Symposium on Mathematical Foundations of Computer Science*. LIPIcs. Leibniz Int. Proc. Inform., Vol. 83. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, Art. No. 69, 14.

[50] Narayan Vikas. 2018. Computational Complexity Relationship between Compaction, Vertex-Compaction, and Retraction. In *Combinatorial algorithms*. Lecture Notes in Comput. Sci., Vol. 10765. Springer, Cham, 154–166.

[51] Benjamin Widom and John S. Rowlinson. 1970. New Model for the Study of Liquid-Vapor Phase Transitions. *The Journal of Chemical Physics* 52, 4 (1970), 1670–1684. DOI: http://dx.doi.org/10.1063/1.1673203

[52] Dmitriy Zhuk. 2017. A Proof of CSP Dichotomy Conjecture. In *58th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2017*. IEEE Computer Soc., Los Alamitos, CA, 331–342.

Received October 2018; revised March 2020; accepted March 2020