The Complexity of Approximately Counting Retractions[*]

Jacob Focke, Leslie Ann Goldberg and Stanislav Živný[†]

Abstract

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 $\mathrm{P} \neq \mathrm{NP}$). Similarly, the complexity of (exactly) counting retractions from G to H is classified (assuming $\mathrm{FP} \neq \#\mathrm{P}$). However, almost nothing is known about approximately counting retractions. Our first contribution is to give a complete trichotomy for approximately counting retractions to trees. The result is as follows: (1) Approximately counting retractions to a tree H is in FP if H is a star, a single looped vertex, or an edge with two loops. (2) Otherwise, if H 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 #P-complete under approximation-preserving reductions. Our second contribution is to locate the retraction counting problem 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 these problems.

The full version containing detailed proofs is available at https://arxiv.org/abs/1807.00590v1 (version from 2 July 2018). The theorem numbering here matches the full version.

[*]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.

[†]University of Oxford, UK. `jacob.focke@cs.ox.ac.uk`, `leslie.goldberg@cs.ox.ac.uk`, `standa.zivny@cs.ox.ac.uk`

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 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 [24, 25, 29, 37]. In particular, the computational decision problem of determining whether there is a retraction from G to H is well-studied [28, 43, 45, 48, 13]. Retractions have also been studied under the name of *one-or-all list homomorphisms*, *pre-colouring extensions* or simply *extensions*, see, e.g., [10, 11, 1, 33, 41, 35, 12]. See Hell and Nešetřil's review article [26] for a more extensive list of such work.

Homomorphism *counting* problems have been researched extensively as well [5, 19, 6, 16, 15, 31, 20, 9, 27, 3, 14]. 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.

1.1 First Contribution: A Trichotomy for Approximately Counting Tree Retractions

In this work we give a complete complexity classification for the problem of approximately counting retractions to trees (Theorem 1.1). We now informally introduce some definitions 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. We view the fixed graph H as a parameter of the problem $\#\textsc{Ret}(H)$.

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 (self-)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 other than loops. 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* 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 3.

THEOREM 1.1. *Let H be a tree.*

i) If H is an irreflexive star, a single looped vertex, or an edge with two loops, then $\#\textsc{Ret}(H)$ is in FP.

ii) Otherwise, if H is an irreflexive caterpillar or a partially bristled reflexive path, then $\#\textsc{Ret}(H)$ is #BIS*-equivalent under AP-reductions.*

iii) Otherwise, $\#\textsc{Ret}(H)$ is #SAT*-equivalent under AP-reductions.*

Theorem 1.1 immediately applies to the special cases of irreflexive trees and reflexive trees and gives the following classifications. First, suppose that H is an irreflexive tree. Then $\#\textsc{Ret}(H)$ is in FP if H is a star. If H is a caterpillar but not a star, then $\#\textsc{Ret}(H) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$. For all other irreflexive trees H, the problem $\#\textsc{Ret}(H)$ is #SAT-equivalent under AP-reductions. Now suppose that H is a reflexive tree. Then $\#\textsc{Ret}(H)$ 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 $\#\textsc{Ret}(H) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$. Otherwise, $\#\textsc{Ret}(H)$ is #SAT-equivalent with respect to AP-reductions.

We note that our hardness result for irreflexive trees extends beyond trees and actually covers all square-free graphs (Lemma 15 of the full version).

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 $N\big(G \to H\big)$ to denote the size of $\mathcal{H}(G, H)$. The following is the well-known homomorphism counting problem.

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

Note that, in the problem $\#\textsc{Hom}(H)$, 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 $\#\textsc{Ret}(H)$, 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 $\mathbf{S} = \{S_v \subseteq V(H) \mid v \in V(G)\}$ 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 $(G, \mathbf{S})$ 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 $\mathcal{H}((G, \mathbf{S}), H)$ to denote the set of homomorphisms from $(G, \mathbf{S})$ to H and we use $N\big((G, \mathbf{S}) \to H\big)$ to denote the size of $\mathcal{H}((G, \mathbf{S}), H)$. We will be interested in the following generalisation of $\#\textsc{Hom}(H)$.

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

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

Name: $\#\textsc{Ret}(H)$.

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 $\#\textsc{Ret}(H)$ 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. This reduction also shows that the corresponding decision problems are polynomial-time interreducible.

We consider two more related counting problems, namely $\#\textsc{SHom}(H)$, the problem of counting vertex-surjective homomorphisms, and $\#\textsc{Comp}(H)$, the problem of counting edge-surjective homomorphisms, which are called *compactions*. We give their formal definitions in Section 2. Both problems are well-studied in the decision setting [2, 22, 23, 21, 34, 42, 44, 46, 47]. All three of the problems $\#\textsc{SHom}(H)$, $\#\textsc{Comp}(H)$ and $\#\textsc{Ret}(H)$ can be interpreted as problems requiring one to count homomorphisms with some kind of surjectivity constraint. $\#\textsc{LSHom}(H)$ and $\#\textsc{LComp}(H)$ are the corresponding list homomorphism problems and these are formally defined in the full version.

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, $\#\textsc{Hom}(H)$, $\#\textsc{Ret}(H)$ and $\#\textsc{LHom}(H)$ are interreducible. Also, all of the exact counting problems that we have mentioned reduce to $\#\textsc{Comp}(H)$ and $\#\textsc{LComp}(H)$ [14], as depicted in Figure 2. Moreover, $\#\textsc{Comp}(H)$ and $\#\textsc{LComp}(H)$ 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 1.1. *Let H be a graph. Then $\#\textsc{Hom}(H) \leq_{\mathrm{AP}} \#\textsc{Ret}(H) \leq_{\mathrm{AP}} \#\textsc{LHom}(H)$.*

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, Observation 1.1 indicates that $\#\textsc{Ret}(H)$ is an important intermediate problem, between the solved $\#\textsc{LHom}(H)$ and the wide-open $\#\textsc{Hom}(H)$.

The first interesting consequence of Theorem 1.1 is a separation between $\#\textsc{Ret}(H)$ and $\#\textsc{LHom}(H)$.

Corollary 1.1. *$\#\textsc{Ret}(H)$ and $\#\textsc{LHom}(H)$ 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 $\#\textsc{Ret}(H) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$, whereas $\#\textsc{LHom}(H) \equiv_{\mathrm{AP}} \#\mathrm{SAT}$.*

The fact that $\#\textsc{Ret}(H) \equiv_{\mathrm{AP}} \#\mathrm{BIS}$ for partially bristled reflexive paths follows from Theorem 1.1. The fact that $\#\textsc{LHom}(H) \equiv_{\mathrm{AP}} \#\mathrm{SAT}$ is from [16], see Theorem 1.4 in the Related Work section.

As a second consequence, Theorem 1.1 separates $\#\textsc{Ret}(H)$ from $\#\textsc{Hom}(H)$, but in a different sense. For $q \geq 3$ let J_q be the irreflexive tree obtained from the q-leaf star by subdividing each edge. From Goldberg and Jerrum [19] it is known that the problem $\#\textsc{Hom}(J_q)$ is AP-interreducible with the task of computing the partition function of the q-state ferromagnetic Potts model [38] — a well-studied model from statistical physics. Despite extensive work on this problem [19, 18, 17] 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 1.2. *Let q be an integer with $q \geq 3$. $\#\textsc{Hom}(H)$ and $\#\textsc{Ret}(H)$ 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 $\#\mathrm{SAT} \leq_{\mathrm{AP}} \#\textsc{Ret}(J_q)$.*

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

$$\begin{tikzcd}
 & \#\text{SHom}(H) \arrow[dr, "\text{Thm. 1.2}"] & & \#\text{LSHom}(H) \arrow[d, leftrightarrow, "\text{Thm. 1.3}"] \\
\#\text{Hom}(H) \arrow[ur, "\text{[14, Lem.37]}"] \arrow[rr, "\text{Obs. 1.1}", "\text{Cor. 1.2}"', "/" marking] \arrow[dr, "\text{[14, Lem.38]}"'] & & \#\text{Ret}(H) \arrow[r, "\text{Obs. 1.1}", "\text{Cor. 1.1}"', "/" marking] & \#\text{LHom}(H) \arrow[d, leftrightarrow, "\text{Thm. 1.3}"] \\
 & \#\text{Comp}(H) \arrow[ur, "\text{Thm. 1.2}"'] & & \#\text{LComp}(H)
\end{tikzcd}$$

Figure 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.

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

Figure 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.

Theorem 1.3. *Let H be a graph. Then* $\#\text{LSHom}(H) \equiv_{\text{AP}} \#\text{LHom}(H)$ *and* $\#\text{LComp}(H) \equiv_{\text{AP}} \#\text{LHom}(H)$.

Using Theorem 1.2 and Corollary 1.1 we can deduce that $\#\text{SHom}(H)$ and $\#\text{LHom}(H)$ are also separated subject to the assumption that #BIS and #SAT are not AP-interreducible. The same holds for $\#\text{Comp}(H)$ and $\#\text{LHom}(H)$. Moreover, from Theorem 1.3 it follows that we can replace the problem $\#\text{LHom}(H)$ with $\#\text{LSHom}(H)$ or $\#\text{LComp}(H)$ in these separations.

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

Corollary 1.3. *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 $\#\text{SHom}(H)$, $\#\text{Comp}(H)$ *and* $\#\text{Ret}(H)$ *are* #BIS-*equivalent.*

The #BIS-easiness results in Corollary 1.3 come from our Theorem 1.2 together with Observation 1.1 and the #BIS-easiness results for $\#\text{LHom}(H)$ given in Theorem 1.4 on page 5. The corresponding #BIS-hardness comes from [14, Theorem 35]).

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 17 of the full version) we build upon a technique that was introduced by Dyer et al. [5] and extended by Kelk [31] 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. Our framework is convenient to generate #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.

For the #SAT-hardness part of Theorem 1.1, we identify a special class of trees that we call $\mathcal{R}$. The hardness results for trees outside of $\mathcal{R}$ are shown mainly by combining existing results. For instance, we use modifications of, and a more careful analysis of, a gadget from [19] to prove the #SAT-hardness for irreflexive trees (Lemma 15 of the full version) — which actually extends to square-free graphs. The bulk of the work is then to prove hardness for the trees in $\mathcal{R}$ (Lemma 45 of the full version). This requires a more sophisticated analysis of homomorphism structures arising from intricate gadgets that are based on simpler versions used in [5] and [31].

Our proof of the reductions $\#\text{SHom}(H) \leq_{\text{AP}} \#\text{Ret}(H)$ and $\#\text{Comp}(H) \leq_{\text{AP}} \#\text{Ret}(H)$ is based on a Monte Carlo argument (a sketch of the argument is given in Section 2 — the proof is in Lemma 48 of the full version). It uses the fact that #RET(H) is a self-reducible problem, and therefore a #RET(H) oracle gives an algorithm to efficiently sample from the set of retractions to H, as was shown in general by Jerrum, Valiant and Vazirani [30]. A naive rejection sampling approach, however, does not lead to an efficient algorithm as the number of surjective homomorphisms and the number of compactions might be very small compared to the total number of retractions. Our method to shrink the sample space is based on the following fact. For every surjective homomorphism (for every compaction) 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 (is already a compaction). 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 $I_1, \ldots, I_k$ of the problem #RET(H). For $i \in \{1, \ldots, k\}$ let R_i be the set of retractions of the instance I_i. Then the total number of surjective homomorphisms (the total number of compactions) from G to H is the size of 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 [36]. 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 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 1.4. ([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.*

Naturally, Theorem 1.4 also gives a complete classification for the subclass of problems where the graph H is a tree. From our Theorem 1.1 and Theorem 1.4 we immediately obtain the separation between #RET(H) and #LHOM(H) given in Corollary 1.1. Note that the classifications of #RET(H) for irreflexive trees and reflexive trees are identical to the corresponding classifications of the problem #LHOM(H). A separation between these problems only occurs for trees 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 1.5. ([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 1.4 and 1.5.

THEOREM 1.6. ([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 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.

Name: $\textsc{Ret}(H)$.
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?

$\textsc{Ret}(H)$ is completely classified as a result of the recent proof of the CSP dichotomy conjecture [4, 50]. However, these proofs do not give a graph-theoretical characterisation. Feder, Hell, Jonsson, Krokhin and Nordh [13] give the following graph-theoretical characterisation for pseudotrees, where a pseudotree is a graph with at most one non-loop 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 1.7. ([13]) *Let H be a pseudotree. Then $\textsc{Ret}(H)$ is in* P *only if all of the following hold:*

- *H is loop-connected,*
- *H does not contain an induced cycle of size at least* 5*,*
- *H does not contain an induced reflexive cycle of size* 4 *and*
- *H does not contain an induced irreflexive cycle of size* 3.

Otherwise $\textsc{Ret}(H)$ *is* NP*-complete.*

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].

2 Retractions and the complexity landscape

This section relates to the "second contribution" discussed in the introduction — locating $\#\textsc{Ret}(H)$ in the approximate counting landscape. In particular, we discuss the ideas needed to prove Theorems 1.2 and 1.3. The proofs of these theorems use a Monte Carlo approach to reduce $\#\textsc{SHom}(H)$ and $\#\textsc{Comp}(H)$ to $\#\textsc{Ret}(H)$ and also to reduce $\#\textsc{LSHom}(H)$ and $\#\textsc{LComp}(H)$ to $\#\textsc{LHom}(H)$.

We start with brief definitions. Let G be an irreflexive graph. A homomorphism $h\colon G \to H$ is *surjective* if, for every vertex $v \in V(H)$, there is a vertex $u \in V(G)$ such that $h(u) = v$. Also, 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 $h(u_2) = v_2$. $\#\textsc{SHom}(H)$ is the problem of counting surjective homomorphisms from G to H and $\#\textsc{Comp}(H)$ is the problem of counting compactions from G to H. $\#\textsc{LSHom}(H)$ and $\#\textsc{LComp}(H)$ are the list-homomorphism versions of these problems (see the full version for details).

To smooth the presentation, and avoid redundancy, the full version defines a common generalisation of the problems $\#\textsc{Comp}(H)$ and $\#\textsc{LComp}(H)$ so that a *single* algorithm provides both the reduction from $\#\textsc{Comp}(H)$ to $\#\textsc{Ret}(H)$ and the reduction from $\#\textsc{LComp}(H)$ to $\#\textsc{LHom}(H)$.[2] This is accomplished by defining an additional parameter $\mathcal{L}$, which is a set of subsets of $V(H)$. Given an irreflexive graph G and a collection of lists $\mathbf{S} = \{S_v \in \mathcal{L} \mid v \in V(G)\}$, we show how to approximately count, e.g., compactions from $(G,\mathbf{S})$ to H using an oracle for list homomorphisms to H where all lists are in $\mathcal{L} \cup \{\{v\} \mid v \in V(H)\}$.

The "base case" where $\mathcal{L} = \{V(H)\}$ constitutes an AP-reduction from $\#\textsc{Comp}(H)$ to $\#\textsc{Ret}(H)$. We stick to this "base case" here — avoiding the extra notation (and referring the reader to the full version for the other results). Thus, our goal here is to give a randomised algorithm for $\#\textsc{Comp}(H)$, using an oracle for $\#\textsc{Ret}(H)$. We will use the (possibly randomised) $\#\textsc{Ret}(H)$ oracle and self-reducibility to obtain the following subroutines.

1. $\textsc{CountHom}_H$ is an algorithm that takes an input $(G,\mathbf{S})$ to $\#\textsc{Ret}(H)$, along with accuracy parameters ε and δ. Its output is, with probability at least $1-\delta$, within $1 \pm \varepsilon$ of the desired output $N\big((G,\mathbf{S}) \to H\big)$.
2. $\textsc{SampleHom}_H$ is an algorithm that takes as input an input $(G,\mathbf{S})$ to $\#\textsc{Ret}(H)$, along with an accuracy parameter ε. It samples from a distribution which is very close to the uniform distribution on $\mathcal{H}((G,\mathbf{S}),H)$ — the total variation distance between the two distributions is at most ε.

The running time of $\textsc{CountHom}_H$ is polynomial in $n = |V(G)|$, ε^{-1} , and $\log \delta^{-1}$. Similarly, the running time of $\textsc{SampleHom}_H$ is polynomial in n and $\log \varepsilon^{-1}$. We will need the fact that this is logarithmic as the precision that we use is exponential in n.

Given a subset U of $V(G)$, $G[U]$ denotes the subgraph of G that is induced by U. The following observation is the key to the design of our algorithm: If there is a compaction from G to H then there is a set $U \subseteq V(G)$ with $|U| \le |V(H)| + 2|E(H)|$ and a com-

[2]In this short section, we stick with compactions — details concerning $\#\textsc{SHom}(H)$ and $\#\textsc{LSHom}(H)$ are in the full version.

paction τ from $G[U]$ to H. Thus, we define

$$T_G = \{(U,\tau) \mid U \subseteq V(G), |U| \le |V(H)| + 2|E(H)|, \\ \tau \text{ is a compaction from } G[U] \text{ to } H\}$$

and $t_G = |T_G|$. To shorten the notation, for a positive integer k, we let $[k] = \{1,\dots,k\}$. Let $(U_i,\tau_i)_{i\in[t_G]}$ be an arbitrary indexing of the elements of T_G. For $i \in [t_G]$ we define $\Omega_{G,i} = \{\sigma \in \mathcal{H}(G,H) \mid \sigma|_{U_i} = \tau_i\}$, $\Omega_G^+ = \{(i,\sigma) \mid i \in [t_G] \text{ and } \sigma \in \Omega_{G,i}\}$ and $\Omega_G = \Big\{(i,\sigma) \in \Omega_G^+ \mid \sigma \notin \bigcup_{k=1}^{i-1} \Omega_{G,k}\Big\}$. Note that $|\Omega_G^+| = \sum_{i\in[t_G]}|\Omega_{G,i}|$. As every element of a set $\Omega_{G,i}$ is a compaction from G to H and every such compaction is contained in a set $\Omega_{G,i}$, we have

$$\begin{aligned}|\Omega_G| &= \Big|\bigcup_{i\in[t_G]} \Omega_{G,i}\Big| \\ &= \Big|\{\sigma \in \mathcal{H}(G,H) \mid \exists i \in [t_G] \text{ with } \sigma \in \Omega_{G,i}\}\Big| \\ &= N^{\text{comp}}\big(G \to H\big).\end{aligned}$$

It is clear from the definitions that $|\Omega_G| \ge |\Omega_G^+|/t_G$. Thus, $N^{\text{comp}}\big(G \to H\big) = |\Omega_G| \ge \Omega_G^+/t_G$.

This lower bound shows that $|\Omega_G|$ is reasonably dense in $|\Omega_G^+|$. Using the routines $\textsc{CountHom}_H$ and $\textsc{SampleHom}_H$ we can obtain samples from $|\Omega_G^+|$ and design a Monte Carlo algorithm to approximately compute $|\Omega_G| = N^{\text{comp}}\big((G,\mathbf{S}) \to H\big)$ with high probability. This algorithm is presented in Algorithm 1. Note that single-vertex lists are used in our algorithm to fix the images of the sets U_i. The size of each U_i is bounded by $|V(H)| + 2|E(H)|$. Consequently, the algorithm can enumerate all elements of T_G in polynomial time. The more general version of the algorithm and its analysis are given in Section 3 of the full version.

Algorithm 1 Approximate Computation of $|\Omega_G|$. Note that (U_i,τ_i) is the i'th element of T_G.

```
Input: Irreflexive graph $G$ and $\varepsilon$, $\delta \in (0,1)$.
  if $t_G = 0$
      $Y = 0$.
  else
      $\varepsilon' = \frac{\varepsilon}{12}$, $\delta' = \frac{\delta}{2}$, $\delta'' = \frac{\delta'}{t_G}$.
      for $i = 1,\dots,t_G$
          For all $v \in V(G)$, if $v \in U_i$, set $S_v^i = \{\tau_i(v)\}$,
              otherwise set $S_v^i = V(H)$.
          $\mathbf{S}^i = \{S_v^i \subseteq V(H) \mid v \in V(G)\}$
          $\omega_i = \textsc{CountHom}_H(G, \mathbf{S}^i, \varepsilon', \delta'')$.
      $\omega = \sum_{i=1}^{t_G} \omega_i$.
      $m = \left\lceil 2t_G \cdot 3\frac{\ln(2/\delta')}{\varepsilon'^2}\right\rceil$.
      for $j = 1,\dots,m$
          Choose $i \in [t_G]$ with probability $\frac{\omega_i}{\omega}$.
          $\sigma_j = \textsc{SampleHom}_H(G, \mathbf{S}^i, \varepsilon'/(2|V(H)|^n))$.
          Let $X_j$ be 1 in the event $(i,\sigma_j) \in \Omega_G$ and 0
              otherwise.
      $Y = \frac{\omega}{m}\sum_{j=1}^m X_j$.
Output: $Y$
```

3 Approximately Counting Retractions to Trees

In this section we give an overview of the proof of our first contribution, Theorem 1.1. The most difficult part of the proof is the #SAT-hardness part. To shorten the terminology, we will refer to a tree H as "#SAT-hard" whenever $\#\textsc{Ret}(H)$ is #SAT-hard. In the proof, we must identify a manageable number of different classes of trees such that, for each class we can give a separate #SAT-hardness proof, and the union of these classes covers the entire (infinite) class of #SAT-hard trees. Finding the right classes (and the right formulation of these classes) is the first challenge.

First, consider the graph H from Figure 3. H is an important example of what turns out to be the key class of #SAT-hard trees. To prove $\#\text{SAT} \le_{\text{AP}} \#\textsc{Ret}(H)$, we use a gadget $(J,\mathbf{S})$ as displayed in Figure 4. It turns out that almost all homomorphisms from $(J,\mathbf{S})$ to H are in one of two "states". In particular, for almost every such homomorphism h, one of the following holds:

- $h(A) = \{b,y\}$, $h(B) = h(C') = \{r_1,r_2,b,g\}$ and $h(C) = h(B') = h(A') = \{b\}$ (this is state 1).
- $h(A') = \{b,y\}$, $h(B') = h(C) = \{r_1,r_2,b,g\}$ and $h(C') = h(B) = h(A) = \{b\}$ (this is state 2).

This bistable behaviour of homomorphisms from $(J,\mathbf{S})$ to H is suitable for a reduction from the problem of counting large cuts, which is known to be #SAT-hard [5]. The underlying idea is as follows. We are to (approximately) count (large) cuts of a graph G using a $\#\textsc{Ret}(H)$ oracle. To this end, we replace

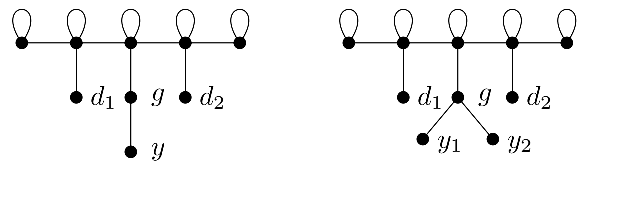

Figure 3: The graphs H (and the left) and H' (on the right).

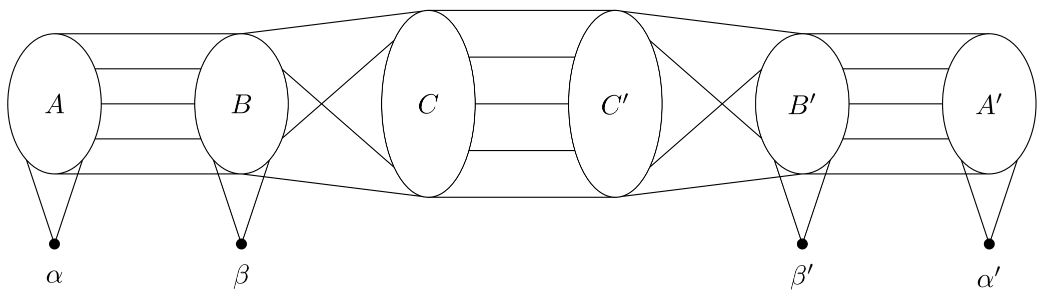

Figure 4: In the graph J, the sets A, B, C, C', B' and A' are independent sets. The size of C and C' is chosen to be appropriately larger than the others. A is matched to B, which is completely connected to C, which is matched to C'. A' is matched to B', which is completely connected to C'. There are four additional vertices: α, β, β' and α', which are completely connected to A, B, B' and A', respectively. All lists in $\mathbf{S}$ are $V(H)$ except $S_\alpha = S_{\alpha'} = \{g\}$ and$S_\beta = S_{\beta'} = \{b\}$.

each vertex of G with a vertex gadget $(J, \mathbf{S})$. The two states of the vertex gadget then model the two different parts of a cut of the graph G. This way, (almost all) retractions correspond to cuts of G. We then use different edge gadgets to ensure that the retractions actually correspond to *large* cuts. The details on this construction are in Section 2.2.2 of the full version. The essence is that we have to carefully tailor a gadget to the graph H with the goal of obtaining bistable behaviour of the gadget.

The next challenge is to use this gadget to obtain hardness results for a broader class of trees, which we call $\mathcal{R}$. $\mathcal{R}$ turns out to be the most important subclass of trees for which we show #SAT-hardness. The definition of this class will exclude partially bristled reflexive paths, since we show that these are #BIS-easy. To ease the notation, we first give the formal definition of a partially bristled reflexive path. Then we define the class $\mathcal{R}$.

Definition 3.1. *A* partially bristled reflexive path *(see Figure 5) 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 $V(H) = \{c_0, \dots, c_{Q+1}\} \cup \bigcup_{i \in S}\{g_i\}$ 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 the graph H from Figure 3 is almost a partially bristled reflexive path. The only difference is the vertex y. However, we will see that counting retractions to partially bristled reflexive paths is actually #BIS-easy. Therefore the vertex y is what causes the difference in complexity. Note that this vertex appears in the two dominating states of the vertex gadget $(J, \mathbf{S})$. In general, the class $\mathcal{R}$ is more complicated, and there can be entire trees in place of the vertex y (and there can be multiple such trees as well). Here is the definition of $\mathcal{R}$.

Definition 3.2. *A tree H is in the set $\mathcal{R}$ if it is not a partially bristled reflexive path, but it has the following form (see Figure 5). Let Q be a positive integer and let S be a non-empty subset of $[Q]$. For each $i \in S$, there is an irreflexive tree $\mathcal{T}_i$ with a designated vertex $g_i \in V(\mathcal{T}_i)$. Then $V(H) = \{c_0, \dots, c_{Q+1}\} \cup \bigcup_{i \in S} V(\mathcal{T}_i)$ and $E(H) = \bigcup_{i=0}^{Q}\{c_i, c_{i+1}\} \cup \bigcup_{i=0}^{Q+1}\{c_i, c_i\} \cup \bigcup_{i \in S}(\{c_i, g_i\} \cup E(\mathcal{T}_i))$.*

The good thing about $\mathcal{R}$ is that we can always apply the gadget idea described for the graph H to obtain bistable behaviour. The behaviour might differ depending on the graph in $\mathcal{R}$, but that is fine. For instance, consider the graph H' as given in Figure 3. Here the dominating states are

- $h(A) = h(C) = \{b, y_1, y_2\}$, $h(B) = \{g\}, h(C') = \{r_1, r_2, b, g\}$ and $h(B') = h(A') = \{b\}$, and
- $h(A') = h(C') = \{b, y_1, y_2\}$, $h(B') = \{g\}, h(C) = \{r_1, r_2, b, g\}$ and $h(B) = h(A) = \{b\}$.

This gives the intuition on how to handle graphs in $\mathcal{R}$. What about trees outside of $\mathcal{R}$ for which #Ret(H) is #SAT-hard? There are actually a number of different classes for which we show #SAT-hardness by different means:

1. For irreflexive trees that contain an induced J_3 we show hardness via a reduction from counting multiterminal cuts.
2. For graphs that are not loop-connected, hardness carries over from the fact that the corresponding decision problem is NP-complete.
3. For a number of remaining cases we combine existing hardness results for the problem of counting

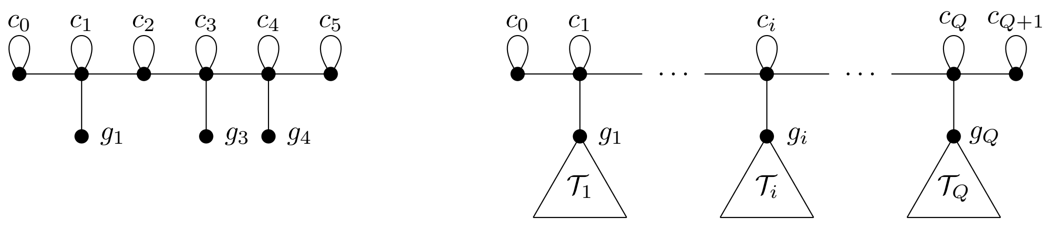

Figure 5: On the left, a partially bristled reflexive path. On the right, a graph in the class $\mathcal{R}$.

homomorphisms to certain small graphs to obtain hardness for #RET(H).

This completes our sketch of the #SAT-hardness proof, the details of which are given in Sections 2.1 and 2.2.2 of the full version. The important consequence of the different hardness proofs is that they actually cover all trees that are not covered by our easiness results and therefore give a complete classification.

Finally, we briefly consider the #BIS-easiness part of Theorem 1.1. The interesting piece is the result for partially bristled reflexive paths, as for these graphs approximately counting list homomorphisms is known to be #SAT-hard (and therefore easiness does not carry over from #LHOM(H)).

As shown by Dyer et al. [5] #BIS-easiness proofs are often based on a reduction to the problem of counting downsets of a partial order. It turns out that we can generalise this technique to apply to retractions. To this end, we cast the problem of counting downsets as a counting constraint satisfaction problem (CSP). We introduce a convenient general framework which can be used to generate #BIS-easiness results both for the problem of approximately counting homomorphisms and for that of approximately counting retractions. Moreover, the framework is sufficiently general to handle not only undirected but also directed graphs — even though the latter is not used in this work.

The idea is to reduce to the problem #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$), which is the counting CSP problem where the constraint language consists of the two unary relations $\delta_0 = \{(0)\}$ and $\delta_1 = \{(1)\}$ and the "Implies" relation $\mathrm{Imp} = \{(0,0),(0,1),(1,1)\}$. #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$) is known to be #BIS-equivalent [7].

We sketch the construction for the undirected case. Given *any* instances I_v and I_e of #CSP($\{\mathrm{Imp}\}$) on a variable set X we define an 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 $\sigma'(x) \Rightarrow \sigma(y)$.

We then show that the problem #RET($H_{I_\mathrm{v},I_\mathrm{e}}$) reduces to the #BIS-easy problem #CSP($\{\mathrm{Imp}, \delta_0, \delta_1\}$). By choosing different instances I_v and I_e one can generate different #BIS-easiness results. It remains to show how to set up these instances to obtain the easiness result for partially bristled reflexive paths. This builds on the work of Kelk [31]. The details of this, along with the generalisation of the framework to directed graphs, are given in Section 2.2.1 of the full version.

References

[1] M. Bíró, M. Hujter, and Zs. Tuza. Precoloring extension. I. Interval graphs. *Discrete Math.*, 100(1-3):267–279, 1992. 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. The complexity of surjective homomorphism problems—a survey. *Discrete Appl. Math.*, 160(12):1680–1690, 2012.

[3] Christian Borgs, Jennifer Chayes, László Lovász, Vera T. Sós, and Katalin Vesztergombi. Counting graph homomorphisms. In *Topics in discrete mathematics*, volume 26 of *Algorithms Combin.*, pages 315–371. Springer, Berlin, 2006.

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

[5] Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The Relative Complexity of Approximate Counting Problems. *Algorithmica*, 38(3):471–500, 2004. Approximation algorithms.

[6] Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Counting and sampling H-colourings. *Inform. and Comput.*, 189(1):1–16, 2004.

[7] Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. An approximation trichotomy for Boolean ♯CSP. *J. Comput. System Sci.*, 76(3-4):267–277, 2010.

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

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

[10] Tomas Feder and Pavol Hell. List homomorphisms to reflexive graphs. *J. Combin. Theory Ser. B*, 72(2):236–250, 1998.

[11] Tomas Feder, Pavol Hell, and Jing Huang. List Homomorphisms and Circular Arc Graphs. *Combinatorica*, 19(4):487–505, 1999.

[12] Tomas Feder, Pavol Hell, and Jing Huang. Extension problems with degree bounds. *Discrete Appl. Math.*, 157(7):1592–1599, 2009.

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

[14] Jacob Focke, Leslie Ann Goldberg, and Stanislav Živný. The Complexity of Counting Surjective Homomorphisms and Compactions. *CoRR*, abs/1706.08786, 2017. 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. Approximately counting H-colorings is #BIS-hard. *SIAM J. Comput.*, 45(3):680–711, 2016.

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

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

[18] Leslie Ann Goldberg and Mark Jerrum. Approximating the Partition Function of the Ferromagnetic Potts Model. *J. ACM*, 59(5):Art. 25, 31, 2012.

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

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

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

[22] Petr A. Golovach, Bernard Lidický, Barnaby Martin, and Daniël Paulusma. Finding vertex-surjective graph homomorphisms. *Acta Inform.*, 49(6):381–394, 2012.

[23] Petr A. Golovach, Daniël Paulusma, and Jian Song. Computing vertex-surjective homomorphisms to partially reflexive trees. *Theoret. Comput. Sci.*, 457:86–100, 2012.

[24] Pavol Hell. *Retractions des graphes*. ProQuest LLC, Ann Arbor, MI, 1973. Thesis (Ph.D.)–Universite de Montreal (Canada).

[25] Pavol Hell. Absolute retracts in graphs. pages 291–301. Lecture Notes in Math., Vol. 406, 1974.

[26] Pavol Hell and Jaroslav Nesetril. Colouring, constraint satisfaction, and complexity. *Computer Science Review*, 2(3):143–163, 2008.

[27] Pavol Hell and Jaroslav Nešetřil. Counting list homomorphisms for graphs with bounded degrees. In *Graphs, morphisms and statistical physics*, volume 63 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 105–112. Amer. Math. Soc., Providence, RI, 2004.

[28] Pavol Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*, volume 28 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2004.

[29] Pavol Hell and Ivan Rival. Absolute retracts and varieties of reflexive graphs. *Canad. J. Math.*, 39(3):544–567, 1987.

[30] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoret. Comput. Sci.*, 43(2-3):169–188, 1986.

[31] Steven Kelk. *On the relative complexity of approximately counting H-colourings.* PhD thesis, Warwick University, 2003.

[32] Ekkehard G. Köhler. *Graphs without asteroidal triples.* PhD thesis, Technische Universität Berlin, 1999.

[33] Jan Kratochvíl and András Sebő. Coloring precolored perfect graphs. *J. Graph Theory*, 25(3):207–215, 1997.

[34] Barnaby Martin and Daniël Paulusma. The computational complexity of disconnected cut and $2K_2$-partition. *J. Combin. Theory Ser. B*, 111:17–37, 2015.

[35] Dániel Marx. Parameterized coloring problems on chordal graphs. *Theoret. Comput. Sci.*, 351(3):407–424, 2006.

[36] Michael Mitzenmacher and Eli Upfal. *Probability and Computing.* Cambridge University Press, Cambridge, second edition, 2017. Randomization and Probabilistic techniques in Algorithms and Data Analysis.

[37] Erwin Pesch. *Retracts of graphs*, volume 110 of *Mathematical Systems in Economics.* Athenäum Verlag GmbH, Frankfurt am Main, 1988.

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

[39] Wolfgang M. Schmidt. *Diophantine approximations and Diophantine equations*, volume 1467 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1991.

[40] Claus-Peter Schnorr. Optimal Algorithms for Self-Reducible Problems. In *ICALP*, pages 322–337, 1976.

[41] Zsolt Tuza. Graph colorings with local constraints—a survey. *Discuss. Math. Graph Theory*, 17(2):161–228, 1997.

[42] Narayan Vikas. Computational Complexity of Compaction to Reflexive Cycles. *SIAM J. Comput.*, 32(1):253–280, 2002/03.

[43] Narayan Vikas. Compaction, Retraction, and Con-

straint Satisfaction. *SIAM J. Comput.*, 33(4):761–782, 2004.

[44] Narayan Vikas. Computational complexity of compaction to irreflexive cycles. *J. Comput. System Sci.*, 68(3):473–496, 2004.

[45] Narayan Vikas. 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):406–439, 2005.

[46] Narayan Vikas. Algorithms for Partition of Some Class of Graphs under Compaction and Vertex-Compaction. *Algorithmica*, 67(2):180–206, 2013.

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

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

[49] Benjamin Widom and John S. Rowlinson. New Model for the Study of LiquidVapor Phase Transitions. *The Journal of Chemical Physics*, 52(4):1670–1684, 1970.

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