Counting Homomorphisms to K_4-minor-free Graphs, modulo 2*

Jacob Focke† Leslie Ann Goldberg† Marc Roth‡ Stanislav Živný†

Abstract

We study the problem of computing the parity of the number of homomorphisms from an input graph G to a fixed graph H. Faben and Jerrum [ToC'15] introduced an explicit criterion on the graph H and conjectured that, if satisfied, the problem is solvable in polynomial time and, otherwise, the problem is complete for the complexity class $\oplus$P of parity problems.

We verify their conjecture for all graphs H that exclude the complete graph on 4 vertices as a minor. Further, we rule out the existence of a subexponential-time algorithm for the $\oplus$P-complete cases, assuming the randomised Exponential Time Hypothesis.

Our proofs introduce a novel method of deriving hardness from globally defined substructures of the fixed graph H. Using this, we subsume all prior progress towards resolving the conjecture (Faben and Jerrum [ToC'15]; Göbel, Goldberg and Richerby [ToCT'14,'16]). As special cases, our machinery also yields a proof of the conjecture for graphs with maximum degree at most 3, as well as a full classification for the problem of counting list homomorphisms, modulo 2.

A full version of our paper, containing all proofs, is available at `https://arxiv.org/abs/2006.16632v2`. Here we number key lemmas to match the numbering in the full version.

1 Introduction

A homomorphism from a graph G to a graph H is a map h from $V(G)$ to $V(H)$ that preserves edges in the sense that, for every edge $\{u, v\}$ of G, the image $\{h(u), h(v)\}$ is an edge of H. Many combinatorial structures can be modelled using graph homomorphisms. For this reason, graph homomorphisms are ubiquitous in both classical and modern-day complexity theory with applications in areas such as constraint satisfaction problems [24], evaluations of spin systems in statistical physics [1, 2], database theory [25, 31], and parameterised algorithms [4, 34]. The computational problems of finding and counting homomorphisms are therefore amongst the most well-studied computational problems; the analysis of their complexity dates back to the intractability result for computing the chromatic number, one of Karp's original 21 NP-complete problems [28]. More recent work builds on Hell and Nešetřil's celebrated dichotomy theorem [26], which shows that determining whether an input graph G has a homomorphism to a fixed graph H is polynomial-time solvable if H is bipartite, or if H has a self-loop. For any other graph H, they show that the problem is NP-complete.

This paper focusses on the problem of counting homomorphisms. Applications of this problem are discussed in [1]. The complexity of the problem has been the focus of much research (see, for example, [3, 9, 16, 17, 30]).[1]

The complexity of counting homomorphisms was initiated by Dyer and Greenhill [9], who gave a complete dichotomy theorem. The complexity of counting the homomorphisms from an input graph G to a fixed graph H is polynomial-time solvable if every component of H is either a complete bipartite graph with no self-loops or a complete graph in which every vertex has a self-loop. For any other graph H, they show that the problem is #P-complete.

Given that (exactly) counting the homomorphisms to H is #P-complete for almost every graph H, research has focussed on restrictions of the problem. Instead of determining the exact number of homomorphisms from G to H, compute an approximation to this number [16, 17, 30], or determine whether it is odd or even [11, 12, 18, 19], or determine its value modulo any prime p [20, 29]. Alternatively, consider the parameterised complexity [14]. For example, the problem can be studied when the input G is assumed to have bounded treewidth [6] or when H has a bounded treewidth, for example when H is a tree [12, 20, 21, 29].

Restricting the input G to have bounded treewidth makes counting homomorphisms tractable — given this

*The research leading to these results has received funding from the European Research Council (ERC) 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.

†Department of Computer Science, University of Oxford

‡Merton College, University of Oxford

[1] There is also a huge literature on generalisations of this problem such as counting *weighted* homomorphisms (computing partition functions of spin systems or holant problems), counting homomorphisms to *directed* graphs, counting partition functions of constraint satisfaction problems, and counting homomorphisms with restrictions such as surjectivity. These generalisations and restrictions are beyond the scope of this paper.

restriction, the problem is solvable in polynomial time for any fixed H [6, Corollary 5.1]. Restricting the fixed target graph H to have bounded treewidth leads to a more nuanced complexity classification, even for treewidth 1 (when H is a tree). For example, the complexity of approximately counting homomorphisms to a tree H has still not been fully resolved, and it is known that different trees lead to vastly different complexities. For example, approximately counting homomorphisms to the very simple tree that is a path of length 3 is equivalent to #BIS, which is the canonical open problem in approximate counting [10]. Moreover, [21] shows that for every integer $q \geq 3$ there is a tree J_q such that approximately counting homomorphisms to J_q is equivalent to classic problem of approximating the partition function of the q-state Potts model from statistical physics. Also, it shows that there are trees H such that approximately counting homomorphisms to H is NP-hard.

1.1 Counting modulo 2 and Past Work Faben and Jerrum [12] combined the restriction that H is a tree with the restriction that counting is modulo 2. Their result will be important for our work, so we next give the definitions that we need to state their result.

The complexity class $\oplus$P [22, 33] contains all problems of the form "compute $f(x)$ mod 2" such that computing $f(x)$ is a problem in #P. Toda [35] showed that there is a randomised polynomial-time reduction from every problem in the polynomial hierarchy to some problem in $\oplus$P. Thus, $\oplus$P-hardness is viewed as a stronger kind of intractability than NP-hardness. We use $\oplus\textsc{Hom}(H)$ to denote the computational problem of computing the number of homomorphisms from G to H, modulo 2, given an input graph G. It is immediate from the definition that $\oplus\textsc{Hom}(H)$ is in $\oplus$P.

The *involution-free reduction* of a graph H, from [12], is defined as follows. An *involution* σ of H is an automorphism of H whose order is at most 2 (that is, $\sigma \circ \sigma$ is the identity permutation). An involution is *non-trivial* if it is not the identity permutation. A graph H is *involution-free* if it has no non-trivial involutions. H^σ denotes the subgraph of H induced by the fixed points of σ (the vertices v with $\sigma(v) = v$). We write $H \to K$ if there is a non-trivial involution σ of H such that $K = H^\sigma$. The relation $\to^*$ is the reflexive-transitive closure of the relation $\to$. Thus, $H \to^* K$ means that either $K = H$, or there is a positive integer j and a sequence $H_1, \ldots, H_j$ of graphs such that $H = H_1$, $K = H_j$ and, for all $i \in [j]$, $H_i \to H_{i+1}$. Faben and Jerrum [12, Theorem 3.7] showed that every graph H has, up to isomorphism, exactly one involution-free graph H^* such that $H \to^* H^*$. This graph H^* (labelled in a canonical way) is the *involution-free reduction* of H. The relevance of the involution-free reduction is given by the following theorem.

Theorem 1.1 ([12, Theorem 3.4])**.** *For all graphs G and H, the number of homomorphisms from G to H has the same parity as the number of homomorphisms from G to H^*.*

Thus, the computational problem $\oplus\textsc{Hom}(H)$ reduces to $\oplus\textsc{Hom}(H^*)$. Faben and Jerrum made the following conjecture [12].

Conjecture 1.1 (Faben-Jerrum Conjecture)**.** *Let H be a graph. If its involution-free reduction H^* has at most one vertex, then $\oplus\textsc{Hom}(H)$ can be solved in polynomial time. Otherwise, $\oplus\textsc{Hom}(H)$ is $\oplus$P-complete.*

The following progress has been made on the Faben-Jerrum conjecture.

- Faben and Jerrum [12, Theorem 3.8, Theorem 6.1] proved the conjecture for the case where every connected component of H is a tree.
- Göbel, Goldberg and Richerby [18, Theorem 3.8] proved the conjecture for the case where every connected component of H is a cactus graph, which is a connected, simple graph in which every edge belongs to at most one cycle.
- Göbel, Goldberg and Richerby [19, Theorem 1.2] proved the conjecture for the case where H is a simple graph whose involution-free reduction H^* is square-free (meaning that it has no 4-cycle).

The cactus-graph result generalises the tree result, and is incomparable with the square-free result.

1.2 Contributions and Techniques Our first (and main) contribution is to prove the Faben-Jerrum conjecture for every simple graph H that does not have a K_4-minor.

Here, K_4 denotes the complete graph with four vertices. The concept of graph minors is well known (see, for example, [7]). In short, a graph H is K_4-minor-free if K_4 cannot be obtained from H by a sequence of vertex deletions, edge deletions, and edge contractions (removing any self-loops and multiple-edges that are formed by the contraction). Graph classes based on excluded minors form the basis of the graph structure theory of Robertson and Seymour (see [32]).

The class of K_4-minor-free graphs is a rich and well-studied class. It is equivalent to the class of graphs with treewidth at most 2 and it includes all outerplanar and series-parallel graphs [8].

Both trees and cactus graphs are K_4-minor free, so our result subsumes the tree result of Faben and Jerrum

[12] and also the cactus-graph result of Göbel et al. [18]. K_4-minor-free graphs can contain a 4-cycle and, going the other way, square-free graphs can have a K_4-minor. Thus, our result is incomparable with the result of [19]. (As a more minor contribution, our techniques also give a shorter proof of the result of [19] — see Remark 19 of the full version.)

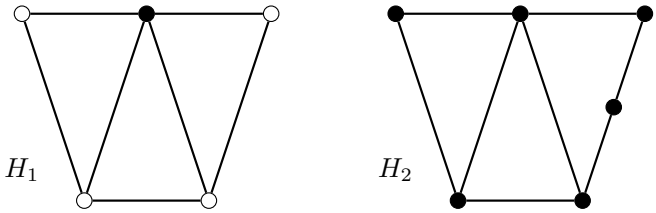
Our second contribution is to extend $\oplus$P-hardness, using the randomised version of the Exponential Time Hypothesis of Impagliazzo and Paturi (rETH) to rule out subexponential algorithms. In order to state our result, we first state the hypothesis.

Conjecture 1.2 (rETH, [27])**.** *There is a positive constant c such that no algorithm, deterministic or randomised, can decide the satisfiability of an n-variable* 3-SAT *instance in time* $\exp(c \cdot n)$.

Using the rETH, we can now state our main result. Here (and in the rest of the paper) we denote the size of the input graph G as $|G| = |V(G)| + |E(G)|$.

Theorem 1.2. *Let H be a simple graph whose involution-free reduction H^* is K_4-minor free. If H^* contains at most one vertex, then $\oplus\textsc{Hom}(H)$ can be solved in polynomial time. Otherwise, $\oplus\textsc{Hom}(H)$ is $\oplus$P-complete and, assuming the randomised Exponential Time Hypothesis, it cannot be solved in time* $\exp(o(|G|))$.

As an example of an application of Theorem 1.2, consider the following K_4-minor-free graphs H_1 and H_2.



The graph H_1 has a non-trivial involution whose only fixed-point is the solid vertex, so H_1^* has one vertex. By Theorem 1.2, $\oplus\textsc{Hom}(H_1)$ can be solved in polynomial time. The graph H_2 does not have any non-trivial involutions, so $H_2^* = H_2$. By Theorem 1.2, $\oplus\textsc{Hom}(H_2)$ is $\oplus$P-complete and it cannot be solved in time $\exp(o(|G|))$, unless the rETH fails.

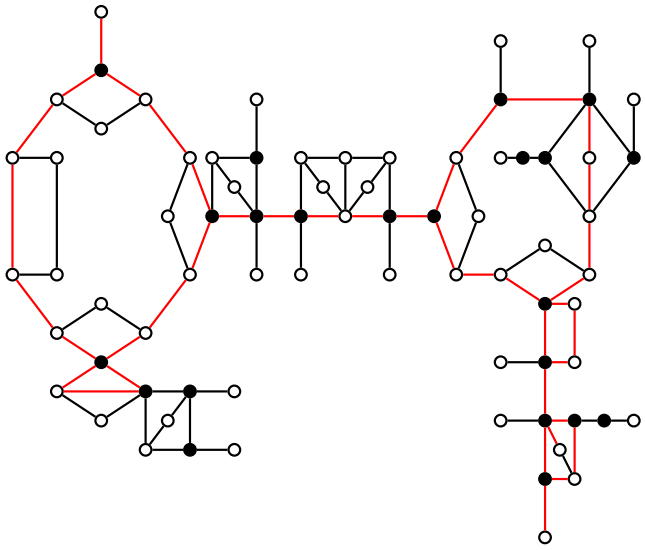
Before describing our techniques, we mention that they lead easily to a couple of other results — a proof of the Faben-Jerrum conjecture for graphs whose involution-free reduction have degree at most 3 (Theorem 87 of the full version) and a complete complexity classification for counting list homomorphisms modulo 2 (Theorem 90 of the full version).

Brief Technical Overview Given Theorem 1.1, we focus on the case where H is involution-free. In general, our proof proceeds in two steps. Given an involution-free K_4-minor-free graph H, in step 1 we try to find a biconnected component of H, let us call it B, that allows us to derive $\oplus$P-hardness of $\oplus\textsc{Hom}(H)$ by exploiting the *local* structure of B to construct a reduction from counting independent sets, modulo 2. The latter problem, denoted by $\oplus$IS, is known to be $\oplus$P-complete [36] and cannot be solved in subexponential time, unless the rETH fails [5].

A careful analysis of biconnected and K_4-minor-free graphs, which crucially relies on the absence of non-trivial involutions, shows that the first step is always possible, unless all biconnected components of H have a very restricted form; examples are depicted in Figure 1.

The second step of the proof exploits the global structure of H and deals with the case where step 1 fails. Note that all of the depicted biconnected components have non-trivial involutions; consider for example the involution given by swapping the vertices x and y in Figure 1. Since the overall graph H is promised to be free of such involutions, we infer that at least one of x and y has a neighbour in a further biconnected component of H, which will allow us to successively construct a global "walk-like" structure in H that eventually yields a reduction from $\oplus$IS.

We consider the construction of those global substructures as our main technical contribution. While the formal specifications are beyond the scope of the introduction, we give an illustrated example which we hope gives some flavour of the graph theory that we will encounter in this work:



The above illustration depicts a K_4-minor-free graph H' without non-trivial involutions, together with a subgraph, highlighted in red, that allows for a reduction

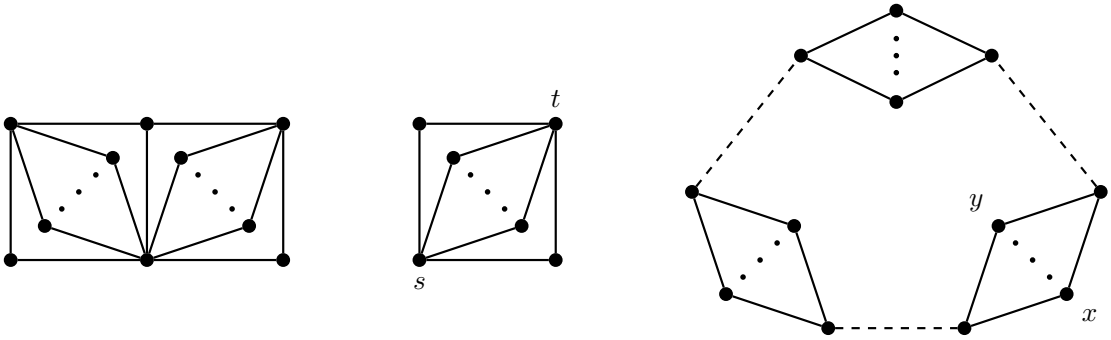


Figure 1: Examples of three types of biconnected and K_4-minor-free graphs that, if viewed as biconnected components, do not yield a reduction from $\oplus$IS. We will re-encounter those graphs as "impasses" *(left)*, "diamonds" *(centre)*, and "obstructions" *(right)*.

from $\oplus$IS. Solid vertices depict articulation points, i.e., vertices that lie in the intersection of at least 2 biconnected components. Note that each biconnected component of H' that is not an edge is of one of the three types given in Figure 1. Also, each biconnected component of H' has an involution. These non-trivial involutions prevent us from exploiting the local structure of the biconnected components to derive $\oplus$P-hardness. Instead, we will see that the highlighted subgraph is what makes $\oplus\textsc{Hom}(H')$ hard.

In the next section we provide an overview of the general framework that allows us to reduce $\oplus$IS to $\oplus\textsc{Hom}(H)$. The structures used in such reductions are captured by the so-called hardness gadgets introduced by Göbel, Goldberg and Richerby [18, 19]. Prior applications of hardness gadgets could only be used to construct a reduction from $\oplus$IS to $\oplus\textsc{Hom}(H)$ if H has certain local substructures, based around a path or a cycle. In contrast, our analysis will establish global walks such as the one highlighted in H'. As far as we can tell, none of the prior machinery [12, 18, 19, 29] is capable of proving the $\oplus$P-hardness of $\oplus\textsc{Hom}(H')$, however, this will follow as a result of our abstract consideration of global substructures of K_4-minor-free graphs.

2 Warm-up: useful Ideas from Previous Papers — Retractions and Hardness Gadgets

Instead of directly reducing $\oplus$IS to $\oplus\textsc{Hom}(H)$, it is useful to consider the intermediate problem $\oplus\textsc{Ret}(H)$, the problem of counting *retractions* to H, modulo 2. Given a graph H, a *partially H-labelled graph* $J = (G, \tau)$ consists of an *underlying graph* G and a corresponding *pinning function* τ, which is a partial function from $V(G)$ to $V(H)$. A homomorphism from J to H is a homomorphism h from G to H such that, for all vertices v in the domain of τ, $h(v) = \tau(v)$.

A homomorphism from a partially H-labelled graph J to H is also called a *retraction*[2] to H because we can think of the pinning function τ as a way of identifying an induced subgraph H of G which must "retract" to H under the action of the homomorphism — see [13] for details. We use $\oplus\textsc{Ret}(H)$ to denote the computational problem of computing the number of homomorphisms from J to H, modulo 2, given as input a partially H-labelled graph J.

It is known [19] that $\oplus\textsc{Ret}(H)$ reduces to $\oplus\textsc{Hom}(H)$ whenever H is involution-free. Since τ allows us to pin vertices of G to vertices of H arbitrarily, it is much easier to construct a reduction from $\oplus$IS to $\oplus\textsc{Ret}(H)$ than to construct a direct reduction from $\oplus$IS to $\oplus\textsc{Hom}(H)$.

Consider the following example. Suppose that H is the 4-vertex path (o, s, i, x) and that our goal is to reduce $\oplus$IS to $\oplus\textsc{Ret}(H)$. Let G be an input to $\oplus$IS. That is, G is a graph whose independent sets we wish to count, modulo 2. For ease of presentation, suppose that G is bipartite,[3] that is, the vertices of G can be partitioned into two independent sets U and V. Let $\widehat{G}$ be the graph obtained from G by adding two additional vertices u and v, and by connecting u to all vertices in U, and v to all vertices in V, respectively. Let τ be the pinning function defined by $\tau(u) = s$ and $\tau(v) = i$. We provide an illustration of the construction in Figure 2.

Observe that any homomorphism φ from $(\widehat{G}, \tau)$ to H must map every vertex in U to either o or i, and every vertex in V to either s or x. Since H has no edge from o to x, the definition of homomorphism ensures that $\varphi^{-1}(o) \cup \varphi^{-1}(x)$ is an independent set of G. It is easy to verify that the function $\varphi \mapsto \varphi^{-1}(o) \cup \varphi^{-1}(x)$ is a bijection between the homomorphisms from $(\widehat{G}, \tau)$ to H

[2] In some definition versions a retraction is surjective. However, for algorithmic problems this surjectivity requirement is not important [13, 15].

[3] The case of general graphs will be discussed later in the paper.

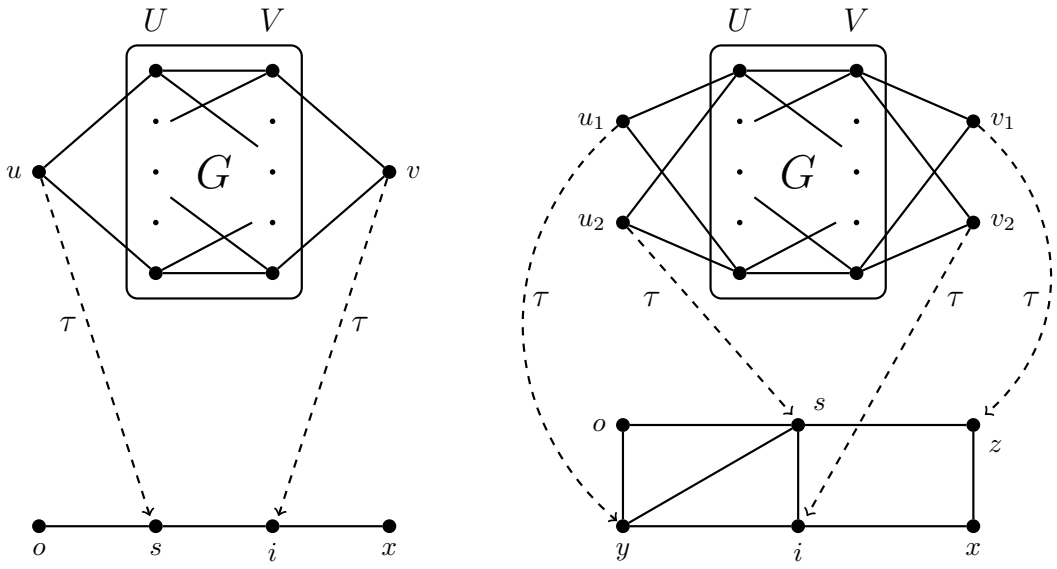


Figure 2: Illustration of the reduction from (bipartite) ⊕IS to ⊕Ret(H) where H is the 4-vertex path *(left)*, and H is the graph H_2 from page 3 *(right)*.

and the independent sets of G, which gives a reduction from (bipartite) ⊕IS to ⊕Ret(H).

The observant reader might notice that the 4-vertex path has a non-trivial involution, and thus, we cannot further reduce ⊕Ret(H) to ⊕Hom(H) in this case.[4] However, the construction works for *any* graph H with an induced path (o, s, i, x) such that s and i each only have two neighbours.

The notion of a *hardness gadget*, which we formally introduce in Section 3, is essentially a generalisation of the previous construction. For example, we could substitute each of o, s, i and x with an odd number of copies, since we are only interested in the parity of the number of independent sets. Furthermore, we could identify o and x, since we only need the edge $\{o, x\}$ to be absent in H. A more sophisticated generalisation is obtained by observing that we can, to some extent, substitute the edges $\{o, s\}$, $\{s, i\}$ and $\{i, x\}$ with more complicated graphs, e.g. with length-2 paths, if we substitute the edges in $\widehat{G}$ accordingly. Finally, observe that the construction $(\widehat{G}, \tau)$ uses the partial function τ in a very simple manner: By adding a common neighbour u for all vertices in U and setting $\tau(u) = s$, the construction enforces the constraint that any homomorphism from $(\widehat{G}, \tau)$ to H must map every vertex in U to a neighbour of s. More sophisticated constructions will allow us to enforce much stronger constraints on homomorphisms. We will need this flexibility to construct reductions from ⊕IS to ⊕Ret(H) for more general graphs H.

We conclude by making a generalisation explicit for one further example — the graph H_2 from page 3. We provide a more convenient drawing of H_2, including a labelling of its vertices and an illustration of the reduction in Figure 2. Again, we will assume for ease of presentation that the input G to ⊕IS is bipartite. To construct $\widehat{G}$, we add two additional vertices u_1 and u_2 and make them adjacent to every vertex in U. Similarly, we add two additional vertices v_1 and v_2 and make them adjacent to every vertex in V. Let τ be the pinning function defined by $\tau(u_1) = y$, $\tau(u_2) = s$, $\tau(v_1) = z$, and $\tau(v_2) = i$.

Consider any homomorphism φ from $(\widehat{G}, \tau)$ to H_2. Since φ is edge-preserving, it must map every vertex in U to a common neighbour of s and y in H_2. Consequently, $\varphi(U) \subseteq \{o, i\}$. Similarly, we obtain $\varphi(V) \subseteq \{s, x\}$. Again, it is easy to see that the mapping $\varphi \mapsto \varphi^{-1}(o) \cup \varphi^{-1}(x)$ is a bijection between the homomorphisms from $(\widehat{G}, \tau)$ to H and the independent sets of G, which gives a reduction from (bipartite) ⊕IS to ⊕Ret(H).

Note that the second example, while being less straightforward than the first, is still a very simple reduction. The proof of Theorem 1.2 requires us to consider much more intricate "hardness gadgets".

3 Detailed Proof Outline

Following the discussion of the previous section, we start by providing the formal definition of a hardness gadget; slightly generalising the original definition of

[4] In fact, the problem ⊕Hom(H) is trivial when H is the 4-vertex path since the number of homomorphisms will always be even.

Göbel, Goldberg and Richerby [19]:

Definition. *[19, Definition 4.1] A* hardness gadget $(I, S, (J_1, y), (J_2, z), (J_3, y, z))$ *for a graph H consists of odd-cardinality sets $I, S \subseteq V(H)$ together with three connected partially H-labelled graphs with distinguished vertices (J_1, y), (J_2, z), and (J_3, y, z). The following properties are satisfied. Let $\Omega_y = \{a \in V(H) \mid |\hom((J_1, y) \to (H, a))|$ is odd$\}$, $\Omega_z = \{b \in V(H) \mid |\hom((J_2, z) \to (H, b))|$ is odd$\}$, and $\Sigma_{a,b} = \hom((J_3, y, z) \to (H, a, b))$. Then $|\Omega_y|$ is even, $I \subset \Omega_y$, $|\Omega_z|$ is even, $S \subset \Omega_z$, and for each $i \in I$, $o \in \Omega_y \setminus I$, $s \in S$ and $x \in \Omega_z \setminus S$, $|\Sigma_{o,x}|$ is even whereas $|\Sigma_{i,s}|$, $|\Sigma_{o,s}|$, and $|\Sigma_{i,x}|$ are odd.*

Combining [Theorem 4.2][19] with the rETH-based lower bound for counting independent sets modulo 2 due to Dell et al. [5], we find (Theorem 11 of the full version) that for every involution-free graph H with a hardness gadget, $\oplus\text{Ret}(H)$ is $\oplus$P-hard and cannot be solved in time $\exp(o(|J|))$, unless rETH fails. Given the previously-mentioned reduction from $\oplus\text{Ret}(H)$ to $\oplus\text{Hom}(H)$ for involution-free graphs H, the key step in proving Theorem 1.2 is showing that every involution-free K_4-minor-free connected graph with at least 2 vertices has a hardness gadget. Section 3.1 discusses the case where we can find such a hardness gadget "locally" (within a biconnected component of H). Section 3.2 discusses the more challenging case where instead we rely on the global structure of H.

3.1 Local Hardness: Biconnected K_4-minor-free graphs A supergraph H of a graph B is called a $(1, 2)$-*supergraph* of B if every edge of H between vertices of B is also an edge of B and every length-2 path of H between vertices of B is also a path of B. A graph B is called a *strong hardness gadget* if every K_4-minor-free supergraph of B has a hardness gadget. B is called a *pre-hardness gadget* if every K_4-minor-free (1,2)-supergraph of B has a hardness gadget. The important fact for us is that, if a K_4-minor-free graph H has a biconnected component B that is a strong or pre-hardness gadget, then H has a hardness gadget.

Unsurprisingly, not every biconnected K_4-minor-free graph B is a strong or a pre-hardness gadget; the most immediate counterexamples are the cases where B is an edge or a square. One might think, for a moment, that those counterexamples are unproblematic since we eventually only consider involution-free graphs. However, involution-free graphs can obviously have biconnected components with involutions, so to complete the proof we have to understand the structure of all biconnected K_4-minor-free graphs that are not pre-hardness gadgets. This is achieved in Lemma 3.1.

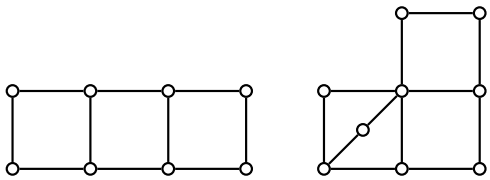
Lemma 3.1 (K_4-minor-free Component Lemma)**.** *Let B be a biconnected K_4-minor-free graph that is not an edge, a diamond, an obstruction or an impasse. Then for every K_4-minor-free graph H containing B as a biconnected component, H has a hardness gadget.*

In order to explain the lemma, we need to identify the relevant biconnected components. A *diamond* is a biconnected graph consisting of two vertices s and t and an independent set of size at least two, all of whose vertices are connected to s and t — see the middle picture in Figure 1. An *impasse* is any biconnected K_4-minor free (1,2)-supergraph of the graph $S_{k,\ell}$ from Figure 3 such that k and ℓ are odd, and all of the vertices that have degree 2 in $S_{k,\ell}$ have degree 2 in B. Finally, an *obstruction* is defined as follows —- this is depicted in Figure 1.

Definition (obstruction)**.** *Let B be a K_4-minor-free biconnected graph and let $C = (c_0, \ldots, c_{q-1}, c_0)$ be an induced cycle of B with $q \neq 4$. We say that B is an* obstruction with cycle C *if the following is true for each $i \in [q]$, taking indices modulo q, whenever the set $N_{C,H}(c_i)$ of common neighbours of c_{i-1} and c_{i+1} in B has even cardinality, each of these common neighbours has degree 2 in B. We use $Cy(B)$ to denote $\{C \mid B$ is an obstruction with cycle $C\}$.*

An important part of the proof of Lemma 3.1 is understanding the induced cycles in biconnected components. Lemma 60 of the full version uses novel "cycle gadgets" to show that every biconnected K_4-minor free graph H with an induced cycle of length at least 5 is either an obstruction or a pre-hardness gadget. A similar result for graphs with induced cycles of length 3 appears as Lemma 59.

So to prove Lemma 3.1, it remains to consider biconnected K_4-minor-free graphs in which every induced cycle is a square — these graphs are called *chordal bipartite graphs*. Towards the proof of Lemma 3.1, Lemmas 31 and 32 of the full version show that the following two graphs are strong hardness gadgets.



The next step is to investigate their common subgraph F from Figure 3. It turns out that F is not a strong hardness gadget. In fact, counting retractions to the graph $S_{k,\ell}$ (from the same figure) can be done in polynomial time whenever k and ℓ are odd. Nevertheless, it turns out that the only problematic super-

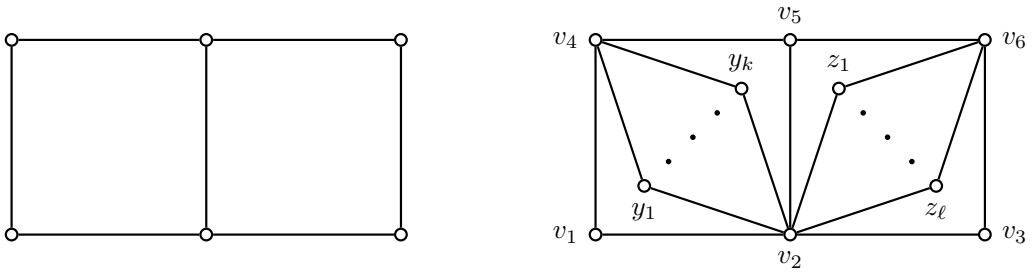


Figure 3: The graphs F (left) and $S_{k,\ell}$ (right).

graphs of F are the impasses that we have already defined. Lemma 36 of the full version shows that every K_4-minor-free graph H containing a chordal bipartite biconnected component B that is not an edge, diamond, or impasse has a hardness gadget, enabling us to prove Lemma 3.1.

3.2 Global Hardness: Block-Cut-Trees of involution-free K_4-minor-free graphs Let H be a K_4-minor-free graph. By Lemma 3.1, if H has no hardness gadget, then every biconnected component of H is an edge, a diamond, an impasse, or an obstruction (see Figure 1). Counting retractions to an edge can be done in polynomial time, and similarly, there are diamonds, impasses and obstructions to which counting retractions, modulo 2, is polynomial-time solvable. This shows that we cannot always establish hardness locally by using a single biconnected component. As noted earlier, all of these biconnected components have nontrivial involutions. Thus, if H is involution-free, there are articulation points in these biconnected components. We will use these articulation points as "connectors" to identify global structure in H.

Definition (suitable connector)**.** *Let H be a graph, let B be a biconnected component of H, and let $A \subseteq V(B)$ be a set of articulation points of H. We say that (B, A) is a* suitable connector *in H if one of the following cases holds:*

- *B is an edge $\{a, b\}$ and $A = \{a, b\}$, or*
- *B is a diamond that contains an edge $\{a, b\}$ such that $A = \{a, b\}$, or*
- *B is an impasse and $A = \{v_1, v_3\}$ (as depicted in Figure 3), or*
- *B is an obstruction. In this case (B, A) is a suitable connector in H if there is a cycle $C \in Cy(B)$ such that $A = \{c \in C \mid$ the cardinality of $N_{C,H}(c)$ is even$\}$. If (B, A) is a suitable connector in H then we fix a particular cycle $C(B, A) \in Cy(B)$ such that $A = \{c \in C(B, A) \mid$ the cardinality of $N_{C(B,A),H}(c)$ is even$\}$.*

The global structure of H is represented by its block-cut tree $\mathsf{BC}(H)$. The block-cut-tree $\mathsf{BC}(H)$ has a vertex for each biconnected component of H (such vertices are called *blocks*) and a vertex for each articulation point of H (such vertices are called *cut vertices*). There is an edge between each block B and each cut vertex a in B. We use the definition of "suitable connector" to identify a subtree T of $\mathsf{BC}(H)$, which we call a *closed suitable subtree*. Two important properties of such a subtree T are (1) that any pair consisting of a block of T together with its adjacent cut vertices is a suitable connector, and (2) every cut vertex of T has degree at most 2 in T. Here we will skip the full technical details — they are given in Section 9.2 of the full version. In Section 9.2 we also give an algorithm (Algorithm 1) that finds such a closed suitable subtree in $\mathsf{BC}(H)$, given that H is connected, involution-free and every biconnected component is an edge, diamond, impasse, or obstruction[5]. From this point on we operate on a closed suitable subtree T of $\mathsf{BC}(H)$, and this is the structure that we will use to build hardness gadgets. We distinguish two main cases, depending on whether or not T contains an obstruction.

If T does not contain an obstruction, then all blocks in T have degree at most 2 (since every block in T is part of a suitable connector). So, T is a path as every cut vertex in T also has degree at most 2. The corresponding subgraph of H is a sequence of edges, diamonds and impasses that are connected as specified by the definition of suitable connectors. To deal with these sequences we establish Lemma 3.2, which uses the definition of *good starts* and *good stops* that we state as follows. We write $\Gamma_H(v)$ for the neighbourhood of a vertex v in H.

Definition (good start, good stop)**.** *Let H be a graph and let B be a subgraph of H. Let y be a vertex in B*

[5]Since the graph H is fixed, the running time of Algorithm 1 is not important for us. What is important is that the algorithm gives us a (constructive) proof that such a closed suitable subtree exists.

and let $L_B \subseteq \Gamma_H(y) \cap V(B)$ *and* $R_B = \Gamma_H(y) \setminus V(B)$.

- *We say that* (L_B, y) *is a* good start *in* B *if* $|L_B|$ *is odd and there is a gadget* (J, z) *such that the set* $\{v \in V(H) \mid |\hom((J,z) \to (H,v))| \text{ is odd}\}$ *equals* $L_B \cup R_B$.
- *We say that* (L_B, y) *is a* good stop *in* B *if it is a good start in* B *and* $|R_B|$ *is odd.*

Lemma 3.2 (Chordal Bipartite Sequence Lemma)**.** *For an integer* $q \geq 1$*, let* $B_1, \ldots, B_q$ *be biconnected components of a graph* H *and let* $b_0, \ldots, b_q$ *be vertices such that, for all* $i \in [q]$*,* b_{i-1} *and* b_i *are distinct vertices of* B_i*, and* B_i *satisfies one of the following:*

- B_i *is an edge from* b_{i-1} *to* b_i*,*
- B_i *is a diamond in which* $\{b_{i-1}, b_i\}$ *is an edge, or*
- B_i *is an impasse, where* (b_{i-1}, b_i) *is a pair of connectors of* B_i*. In this case, let* d_i *be the unique common neighbour of* b_{i-1} *and* b_i *in* H*.*

If $|\Gamma_H(b_0) \setminus V(B_1)|$ *is odd, then at least one of the following holds:*

- B_q *is an edge or a diamond and* $(\{b_{q-1}\}, b_q)$ *is a good start in* B_q *but not a good stop in* B_q*,*
- B_q *is an impasse and* $(\{d_q\}, b_q)$ *is a good start in* B_q *but not a good stop in* B_q*, or*
- H *has a hardness gadget.*

The high-level intuition behind Lemma 3.2 is that whenever a biconnected component B_i in the sequence $B_1, \ldots, B_q$ has a good start, then B_{i+1} also has a good start. If such a good start is a good stop, then we find a hardness gadget. These facts heavily rely on the form of each B_i and the corresponding results are given in Sections 7.1 and 7.2 of the full version. Then, starting from the fact that B_1 has a good start, the lemma states that either there is a hardness gadget or the last component B_q has a good start which is not a good stop. Both of these possible outcomes will prove helpful, and our use of Lemma 3.2 will be two-fold. First, if T does not contain an obstruction, then it will turn out that B_q has a good stop, so the lemma yields a hardness gadget. Second, even if T contains an obstruction, the lemma gives insight about the sequences of biconnected components that connect obstructions in T.

The most involved part of our proof arises when the closed suitable subtree T contains a block B that is an obstruction. By the definition of an obstruction, B contains an induced cycle of length not equal to 4. These induced cycles play an important role, and we have already seen that, outside of obstructions, they lead to local hardness gadgets. Sometimes we can use the previously-mentioned cycle gadgets to obtain hardness even when these cycles occur in obstructions. In principle, the cycle gadget obtains hardness from a path (i, s, o, x) that is part of a cycle, similarly to our example from Figure 2 — it is just more complicated to restrict the homomorphic image to the sets $\{o, i\}$ and $\{s, x\}$. However, one of the crucial requirements for this construction to work is that for any vertex c of the cycle, the two neighbours of c on the cycle have an odd number of common neighbours in H. For example, consider the obstruction O_3 depicted in blue at the top of Figure 4 with the cycle $C = (c_0, \ldots, c_9, c_0)$. In C there are two problematic sets: The two common neighbours of c_1 and c_9 are $\{a_7, a_7'\}$, and $\{a_8, a_8'\}$ are the two common neighbours of c_2 and c_4.

In order to deal with these problematic sets, we use the fact that O_3 is a block of T, and by the definition of a closed proper subtree it follows that each of the problematic sets contains an articulation point that is a cut vertex in T. From these articulation points we can further explore the structure of H (following the closed proper subtree T in $\mathsf{BC}(H)$). It turns out that in general there are two possible options, which we state informally using the example O_3 (where both possibilities occur). Formal definitions follow.

(1) Starting at the articulation point $a_7 = c_0$, we encounter a sequence of chordal bipartite components that leads to another obstruction (O_1). Here the crucial insight is that we can extend the hardness criteria that work for cycles to the more general setting of closed walks. We say that c_0 is an *exit* of O_3 and we can extend the cycle in O_3 to a closed walk that traverses cycles in both encountered obstructions (as well as the sequence of chordal bipartite components in between). This will resolve the problematic set $\{a_7, a_7'\}$ but might lead to other problematic sets, which are resolved recursively.

(2) Starting at $a_8 = c_3$, we encounter a sequence of chordal bipartite components that does not lead to an obstruction. Then we say that c_3 is an *attachment point* and the structure attached to c_3 is a sequence of edges, diamonds and impasses and will allow us to use Lemma 3.2 to resolve the problematic set $\{a_8, a_8'\}$ (so that we can obtain a hardness gadget).

We now state the formal definitions necessary to formulate an algorithm (Algorithm 3) which finds the sought-for closed walk in a closed suitable subtree T, starting from an obstruction in T. The analysis of this algorithm (and establishing that the returned closed walk has the right properties to yield a hardness gadget) is the central (and most technical) part of this work and is presented in Section 9.4 of the full version.

Definition (attachment point, exit, destination)**.** *Let* H *be a connected graph and let* T *be a closed suitable subtree of* $\mathsf{BC}(H)$*. Let* a *be a cut vertex that has an obstruction* B *as a neighbour in* T*. Then, since every*

cut vertex of T has degree at most 2, there is a unique maximal-length proper obstruction-free path P^* in T starting at a. Let b be the other endpoint of P^* (possibly $P^* = (a)$ in which case $b = a$). The vertex a is an attachment point of (T, B) *if b is a leaf in T. Otherwise, a is an* exit of (T, B). *In this case, b is adjacent to a block $B' \neq B$ which is an obstruction. We say that (b, B') is the* destination *of a in T.*

Definition ($D(C)$, $\mathrm{W}_C(a)$, $\mathrm{W}_C(a,b)$)**.** *For an integer $q \geq 3$, let $C = (c_0, \ldots, c_{q-1}, c_0)$ be a cycle in a graph H. Then $D(C)$ is the cyclic order induced by the order in which the walk C traverses the vertices $\{c_0, \ldots, c_{q-1}\}$. For $a \in C$, $\mathrm{W}_C(a)$ is the walk from a to itself following all of the vertices of C in the order given by $D(C)$. For $a, b \in C$, $\mathrm{W}_C(a,b)$ is the walk from a to b along C in the order given by $D(C)$.*

In the following algorithms, $\mathsf{dist}_T(u,v)$ is the length of a shortest path in T between u and v. $C(B, \Gamma_T(B))$ is a particular cycle in $\mathrm{Cy}(B)$ of an obstruction B — see the definition of "suitable connector". The expression $\mathrm{P}_H(a,b)$ returns a shortest path from a to b in H (which is uniquely defined whenever it is used here).

Algorithm 2 ExitWalk(T, a^*, B, ℓ, a_0)

Input: A closed suitable subtree T of $\mathsf{BC}(H)$ of a connected graph H, a cut vertex a^* in T, an obstruction B that is a block in T such that $\mathsf{dist}_T(a^*, B) = \ell$, and an exit a_0 of (T, B)

$C \leftarrow C(B, \Gamma_T(B))$

$\{a_0, \ldots, a_k\} \leftarrow$ The exits of (T, B) in the order of $D(C)$, starting from a_0

if $k = 0$

  $W \leftarrow \mathrm{W}_C(a_0)$.

else

  $\{(b_1, B_1), \ldots, (b_k, B_k)\} \leftarrow$ The destinations of $a_1, \ldots, a_k$, respectively

  $W \leftarrow \mathrm{W}_C(a_0, a_1)$

  for $i = 1, \ldots, k$

    $r_i \leftarrow \mathsf{dist}_T(B, b_i)$

    $W \leftarrow W + \mathrm{P}_H(a_i, b_i) +$ ExitWalk$(T, a^*, B_i, \ell + r_i + 1, b_i) +$ $\mathrm{P}_H(b_i, a_i) + \mathrm{W}_C(a_i, a_{i+1 \bmod k+1})$

Output: W

Algorithm 3 Walk(T, B')

Input: A closed suitable subtree T of $\mathsf{BC}(H)$ of a connected graph H, an obstruction B' that is a block in T

if there is an exit a^* of (T, B')

  $(b^*, B^*) \leftarrow$ The destination of a^*

  $r^* \leftarrow \mathsf{dist}_T(a^*, b^*)$

  $W \leftarrow$ ExitWalk$(T, a^*, B', 1, a^*) + \mathrm{P}_H(a^*, b^*) +$ ExitWalk$(T, a^*, B^*, r^*+1, b^*) + \mathrm{P}_H(b^*, a^*)$

else

  $W \leftarrow C(B', \Gamma_T(B'))$

Output: W

In Figure 4 we provide illustrations of a graph H, a closed suitable subtree $T \in \mathsf{BC}(H)$, and (in red) the walk $W = (w_0, \ldots, w_{33}, w_0)$ returned by Walk(T, O_1) (Algorithm 3). (An example explaining how the algorithm produces this output is given in Section 9.4 of the full version.)

Recall that the goal is to use a cycle gadget with this (red) closed walk W to obtain a hardness gadget in H. The only sets that are problematic for this construction are even-cardinality sets of the form $\Gamma_H(w_i) \cap \Gamma_H(w_{i+2})$ for any $i \in [34]$, taking indices modulo 34. Note that $\{a_7, a_7'\}$ is no longer problematic as a_7' is neither a common neighbour of w_7 and w_9, nor a common neighbour of w_{17} and w_{19}. The only problematic sets are $\{w_{26}, w_{26}'\}$ and (still) $\{a_8, a_8'\}$. However, both w_{26} and $w_{11} = a_8$ are attachment points and each have a sequence of edges, diamonds and impasses attached to them such that we can use Lemma 3.2 to resolve the corresponding problematic sets. This outline is very simplified and merely gives a flavour of the construction. The full details, showing how to do this for every such H are presented in Lemma 85 of the full version, which is the main step of the proof of Theorem 1.2.

Acknowledgements

We would like to thank Dave Richerby for valuable discussions. Furthermore we thank Holger Dell for pointing out the tight conditional lower bound for counting independent sets modulo 2 in [5].

References

[1] Christian Borgs, Jennifer Chayes, László Lovász, Vera T. Sós, and Katalin Vesztergombi. Counting graph homomorphisms. In Martin Klazar, Jan Kratochvíl, Martin Loebl, Jiří Matoušek, Pavel Valtr, and Robin Thomas, editors, *Topics in Discrete Mathematics*, pages 315–371, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[2] Graham R. Brightwell and Peter Winkler. Graph homomorphisms and phase transitions. *J. Comb.*

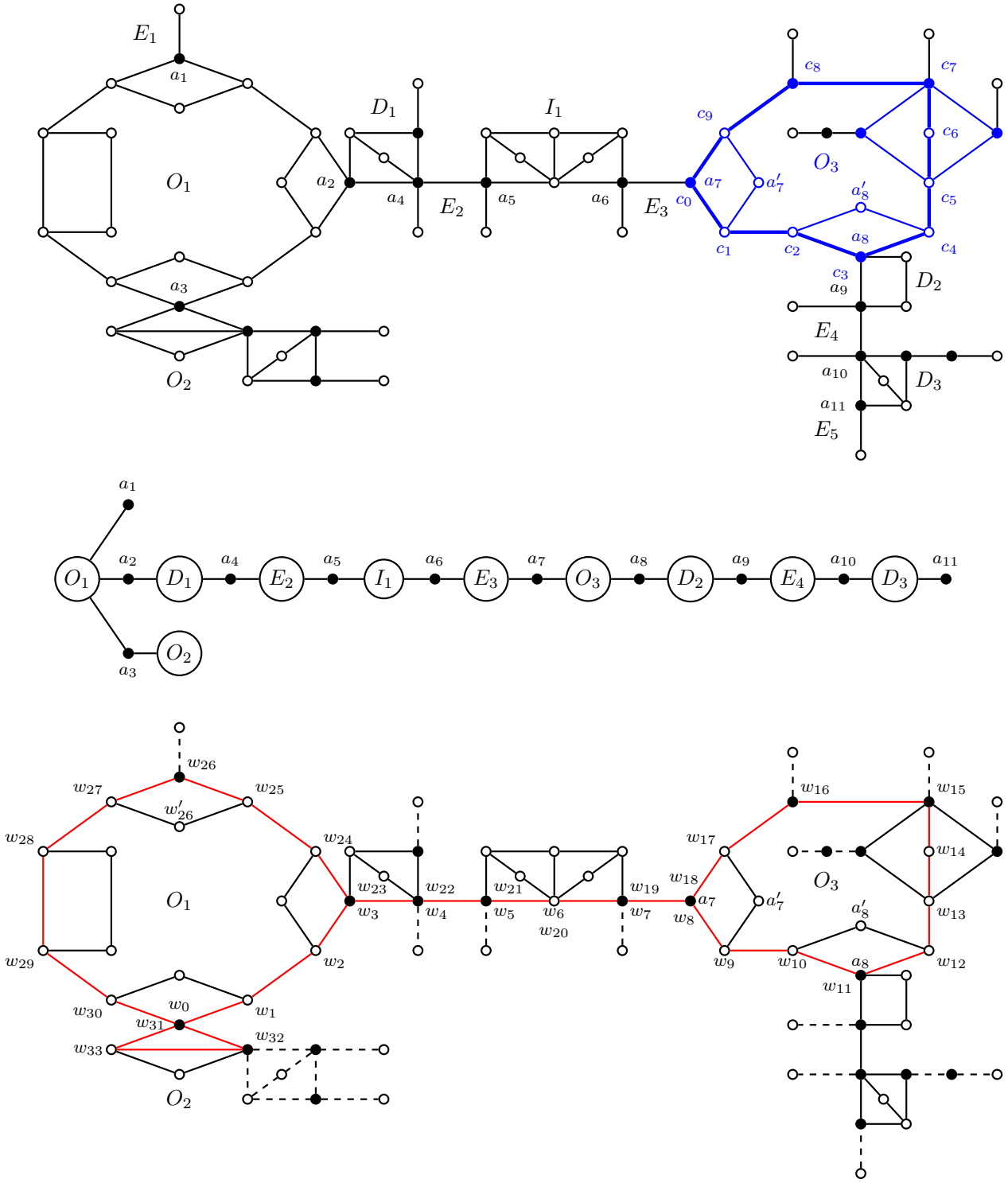


Figure 4: *A graph H, a closed suitable subtree T of $\mathsf{BC}(H)$ with a block O_1 that is an obstruction, and* $\textsc{Walk}(T, O_1)$.
(Top) An involution-free and K_4-minor-free graph H such that every biconnected component is an edge, a diamond, an impasse or an obstruction. Articulation points are depicted as filled vertices.
(Center) A *closed* and *suitable* subtree T of the block-cut tree of H, rooted at O_1.
(Bottom) Solid lines are contained in the subgraph of H induced by $V(T)$, while dashed lines are not. The red closed walk $(w_0, \ldots, w_{33}, w_0)$ is the output of $\textsc{Walk}(T, O_1)$. Observe that w_0 and w_3 are exits of (T, O_1) with destinations (w_8, O_3) and (w_0, O_2), respectively, and that w_{26} and w_{11} are attachment points of (T, O_1) and (T, O_3), respectively.

Theory, Ser. B, 77(2):221–262, 1999. doi:10.1006/jctb.1999.1899.

[3] Hubie Chen, Radu Curticapean, and Holger Dell. The exponential-time complexity of counting (quantum) graph homomorphisms. In Ignasi Sau and Dimitrios M. Thilikos, editors, *Graph-Theoretic Concepts in Computer Science - 45th International Workshop, WG 2019, Vall de Núria, Spain, June 19-21, 2019, Revised Papers*, volume 11789 of *Lecture Notes in Computer Science*, pages 364–378. Springer, 2019. doi:10.1007/978-3-030-30786-8_28.

[4] Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 210–223. ACM, 2017. doi:10.1145/3055399.3055502.

[5] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Trans. Algorithms*, 10(4):21:1–21:32, 2014. doi:10.1145/2635812.

[6] Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Counting H-colorings of partial k-trees. *Theor. Comput. Sci.*, 281(1-2):291–309, 2002. doi:10.1016/S0304-3975(02)00017-8.

[7] Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2016.

[8] Richard J Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10(2):303–318, 1965.

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

[10] Martin E. Dyer, Leslie Ann Goldberg, Catherine S. Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2004. doi:10.1007/s00453-003-1073-y.

[11] John Faben. The complexity of counting solutions to generalised satisfiability problems modulo k. *arXiv preprint arXiv:0809.1836*, 2008.

[12] John Faben and Mark Jerrum. The Complexity of Parity Graph Homomorphism: An Initial Investigation. *Theory of Computing*, 11(2):35–57, 2015. doi:10.4086/toc.2015.v011a002.

[13] Tomas Feder and Pavol Hell. List homomorphisms to reflexive graphs. *J. Combin. Theory Ser. B*, 72(2):236–250, 1998. URL: https://ezproxy-prd.bodleian.ox.ac.uk:4563/10.1006/jctb.1997.1812, doi:10.1006/jctb.1997.1812.

[14] Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004. doi:10.1137/S0097539703427203.

[15] Jacob Focke, Leslie Ann Goldberg, and Stanislav Živnỳ. The complexity of approximately counting retractions to square-free graphs. *arXiv preprint arXiv:1907.02319*, 2019.

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

[17] Andreas Galanis, Leslie Ann Goldberg, and Mark Jerrum. A complexity trichotomy for approximately counting list H-colorings. *ACM Trans. Comput. Theory*, 9(2):9:1–9:22, 2017. doi:10.1145/3037381.

[18] Andreas Göbel, Leslie Ann Goldberg, and David Richerby. The complexity of counting homomorphisms to cactus graphs modulo 2. *ACM Trans. Comput. Theory*, 6(4):17:1–17:29, 2014. doi:10.1145/2635825.

[19] Andreas Göbel, Leslie Ann Goldberg, and David Richerby. Counting homomorphisms to square-free graphs, modulo 2. *ACM Transactions on Computation Theory (TOCT)*, 8(3):12, 2016.

[20] Andreas Göbel, J. A. Gregor Lagodzinski, and Karen Seidel. Counting homomorphisms to trees modulo a prime. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 49:1–49:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.49.

[21] Leslie Ann Goldberg and Mark Jerrum. The complexity of approximately counting tree homomorphisms. *ACM Trans. Comput. Theory*, 6(2):8:1–8:31, 2014. doi:10.1145/2600917.

[22] L. M. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of Boolean functions. *Theor. Comput. Sci.*, 43:43–58, 1986.

[23] Martin Charles Golumbic and Clinton F. Goss. Perfect Elimination and Chordal Bipartite Graphs. *Journal of Graph Theory*, 2(2):155–163, 1978. doi:10.1002/jgt.3190020209.

[24] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1):1:1–1:24, 2007. doi:10.1145/1206035.1206036.

[25] Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 657–666. ACM, 2001. doi:10.1145/380752.380867.

[26] Pavol Hell and Jaroslav Nesetril. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J.

[27] Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

[28] Richard M. Karp. Reducibility among combinato-

rial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

[29] Amirhossein Kazeminia and Andrei A. Bulatov. Counting homomorphisms modulo a prime number. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 59:1–59:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.59.

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

[31] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000. doi:10.1006/jcss.2000.1713.

[32] László Lovász. Graph minor theory. *Bull. Amer. Math. Soc. (N.S.)*, 43(1):75–86, 2006. doi:10.1090/S0273-0979-05-01088-8.

[33] C. H. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proc. 6th GI-Conference on Theoretical Computer Science*, pages 269–275. Springer-Verlag, 1982.

[34] Marc Roth and Philip Wellnitz. Counting and finding homomorphisms is universal for parameterized complexity theory. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2161–2180. SIAM, 2020. doi:10.1137/1.9781611975994.133.

[35] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[36] Leslie G Valiant. Accidental algorthims. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 509–517. IEEE, 2006.