


Anti-Factor is FPT Parameterized by Treewidth and List Size (but Counting is Hard)

Dániel Marx ✉ 

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Govind S. Sankar ✉ 

Duke University, Durham, USA

Philipp Schepper ✉ 

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract

In the general ANTI-FACTOR problem, a graph G and, for every vertex v of G , a set $X_v \subseteq \mathbb{N}$ of forbidden degrees is given. The task is to find a set S of edges such that the degree of v in S is *not* in the set X_v . Standard techniques (dynamic programming plus fast convolution) can be used to show that if M is the largest forbidden degree, then the problem can be solved in time $(M + 2)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ if a tree decomposition of width tw is given. However, significantly faster algorithms are possible if the sets X_v are sparse: our main algorithmic result shows that if every vertex has at most x forbidden degrees (we call this special case ANTI-FACTOR $_x$), then the problem can be solved in time $(x + 1)^{\mathcal{O}(\text{tw})} \cdot n^{\mathcal{O}(1)}$. That is, ANTI-FACTOR $_x$ is fixed-parameter tractable parameterized by treewidth tw and the maximum number x of excluded degrees.

Our algorithm uses the technique of representative sets, which can be generalized to the optimization version, but (as expected) not to the counting version of the problem. In fact, we show that $\#\text{ANTI-FACTOR}_1$ is already $\#\text{W}[1]$ -hard parameterized by the width of the given decomposition. Moreover, we show that, unlike for the decision version, the standard dynamic programming algorithm is essentially optimal for the counting version. Formally, for a fixed nonempty set X , we denote by X -ANTI-FACTOR the special case where every vertex v has the same set $X_v = X$ of forbidden degrees. We show the following lower bound for every fixed set X : if there is an $\epsilon > 0$ such that $\#X$ -ANTI-FACTOR can be solved in time $(\max X + 2 - \epsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ given a tree decomposition of width tw , then the Counting Strong Exponential-Time Hypothesis ($\#\text{SETH}$) fails.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases Anti-Factor, General Factor, Treewidth, Representative Sets, SETH

Digital Object Identifier 10.4230/LIPIcs.IPEC.2022.22

Related Version *Full Version*: <https://arxiv.org/abs/2110.09369> [29]

Funding Research supported by the European Research Council (ERC) consolidator grant No. 725978 SYSTEMATICGRAPH.

Philipp Schepper: Part of Saarbrücken Graduate School of Computer Science, Germany.



© Dániel Marx, Govind S. Sankar, and Philipp Schepper;
licensed under Creative Commons License CC-BY 4.0

17th International Symposium on Parameterized and Exact Computation (IPEC 2022).

Editors: Holger Dell and Jesper Nederlof; Article No. 22; pp. 22:1–22:24

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Matching problems and their generalizations form a well studied class of problems in combinatorial optimization and computer science [26]. A *perfect matching* is a set S of edges such that every vertex has degree exactly 1 in S ; finding a perfect matching is known to be polynomial-time solvable [16, 21, 32]. In the f -FACTOR problem, an integer $f(v)$ is given for each vertex v and the task is to find a set of edges where every vertex v has degree exactly $f(v)$. A simple transformation reduces f -FACTOR to finding a perfect matching. Conversely, in f -ANTI FACTOR the task is to find a set S of edges where the degree of v is *not* $f(v)$ [34].

The problems above can be unified under the GENERAL FACTOR (GENFAC) problem [10, 27, 30], where one is given a graph G and an associated set of integers B_v for every vertex v of G . The objective is to find a subgraph such that every vertex v has its degree in B_v . Cornuéjols [10] showed that the complexity of GENFAC depends on the maximum gap of the sets B_v . The maximum gap of a set B (denoted by $\max\text{-gap}(B)$) is defined as the largest contiguous sequence of integers not in B but whose boundaries are in B . Cornuéjols [10] showed that if $\max\text{-gap}(B_v) \leq 1$, then GENFAC is polynomial-time solvable. In a sense, we can say that this case is the only one that is polynomial-time solvable. Formally, for a fixed, finite set B of integers, B -FACTOR is the special case of GENFAC where every vertex has the same set $B_v = B$ of allowed degrees. It follows from a result of Dalmau and Ford [13] that if B is a fixed finite set such that $\max\text{-gap}(B) > 1$, then B -FACTOR is NP-hard.

Given the hardness of B -FACTOR in general, Marx et al. [30] studied the complexity of the problem on bounded treewidth graphs. Recall the long history of study on treewidth, which is a measure for how “tree-like” a graph is, [3, 4, 6]. For a wide range of hard problems, algorithms with running time of the form $f(k) \cdot n^{\mathcal{O}(1)}$ exist if the input graph comes with a tree decomposition of width k . In many cases even the best possible form of $f(k)$ in the running time is known (under suitable complexity assumptions, such as the Strong Exponential Time Hypothesis (SETH) [23]). Marx et al. [30] use a combination of standard dynamic programming techniques with fast subset convolution (cf. [36]) to give optimal (under SETH) $(\max B + 1)^{\text{tw}} n^{\mathcal{O}(1)}$ time algorithms for the decision, optimization, and counting versions.

- **Theorem 1.1** (Theorems 1.3–1.6 in [30]). *Fix a finite, non-empty set $B \subseteq \mathbb{N}$.*
- *We can count in time $(\max B + 1)^{\text{tw}} n^{\mathcal{O}(1)}$ the solutions of a certain size for a B -FACTOR instance if we are given a tree decomposition of width tw .*
 - *For any $\epsilon > 0$, there is no $(\max B + 1 - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ algorithm for the following problems, even if we are given a path decomposition of width pw , unless SETH (resp. #SETH) fails:*
 - *B -FACTOR and MIN- B -FACTOR if $0 \notin B$ and $\max\text{-gap}(B) > 1$,*
 - *MAX- B -FACTOR if $\max\text{-gap}(B) > 1$,*
 - *# B -FACTOR if $B \neq \{0\}$.*

We study the complementary problem of X -ANTI FACTOR for finite sets X of excluded degrees.

► **Definition 1.2** (X -ANTI FACTOR). *Let $x \in \mathbb{N}$ be fixed. ANTI FACTOR $_x$ is the decision problem of finding for an undirected graph G where all vertices v are assigned a finite set $X_v \subseteq \mathbb{N}$ with $|X_v| \leq x$, a set $S \subseteq E(G)$ such that for all $v \in V$ we have $\deg_S(v) \notin X_v$.*

For a fixed $X \subseteq \mathbb{N}$ with $|X| = x$, we define X -ANTI FACTOR as the restriction of ANTI FACTOR $_x$ to those graphs where all vertices are labeled with the same set X .

► **Note.** \overline{X} -FACTOR, the special case of GENFAC where every vertex has set \overline{X} , precisely corresponds to X -ANTI FACTOR where we set $\overline{X} := \mathbb{N} \setminus X$.

The decision and minimization versions are trivially solvable if $0 \notin X$ as the empty set is a valid solution. Further, if X does not contain two consecutive numbers, then \overline{X} has no gap of size at least two. In this case, by results from Cornu ejols [10] and Dudycz and Paluch [15], the decision, maximization and minimization version of \overline{X} -FACTOR are poly-time solvable.

Our Results. One could expect that similar results can be obtained for X -ANTIFACTOR as for B -FACTOR, but this is very far from the truth and the exact complexity of X -ANTIFACTOR is much less clear. In the B -FACTOR problem, a partial solution (a set of edges that we intend to further extend to a solution) can have degree at most $\max B$ at each vertex, which is the main reason one needs $(\max B + 1)^{\text{tw}} n^{\mathcal{O}(1)}$ running time. For X -ANTIFACTOR, a vertex can also have degree larger than $\max X$ in a (partial) solution, but all degrees larger than $\max X$ are equivalent in some sense. Therefore, the natural running time we expect is $(\max X + 2)^{\text{tw}} n^{\mathcal{O}(1)}$. We show that this running time can be achieved, but requires some modification of the convolution to handle the state “degree more than $\max X$.”

► **Theorem 1.3.** *Let $X \subseteq \mathbb{N}$ be finite and fixed. Given an X -ANTIFACTOR instance and its tree decomposition of width tw . Then we can count the number of solutions of size exactly s in time $(\max X + 2)^{\text{tw}} n^{\mathcal{O}(1)}$ for all s simultaneously.*

However, there are many cases where algorithms significantly faster than $(\max X + 2)^{\text{tw}} n^{\mathcal{O}(1)}$ are possible. At first, this may seem unlikely: at each node of the tree decomposition, the partial solutions can have up to $(\max X + 2)^{\text{tw}+1}$ different equivalence classes¹ and it may seem necessary to find a partial solution for each of these classes. Nevertheless, we show that the technique of *representative sets* can be used to achieve a running time lower than the number of potential equivalence classes. Representative sets were defined by Monien [33] for use in an FPT algorithm for k -PATH, and subsequently found use in many different contexts, including faster dynamic programming algorithms on tree decompositions [1, 5, 7, 17, 18, 19, 25, 31, 35]. The main idea is that we do not need to find a partial solution for each equivalence class, but it is sufficient to find a representative set of partial solutions such that if there is a partial solution that is compatible with some extension, then there is a partial solution in our set that is also compatible with this extension. Our main algorithmic result shows that if X is sparse, then this representative set can be much smaller than $(\max X + 2)^{\text{tw}+1}$, yielding improved algorithms. In particular, ANTIFACTOR_x is FPT parameterized by tw and x .

► **Theorem 1.4.** *One can decide in time $(x + 1)^{\mathcal{O}(\text{tw})} n^{\mathcal{O}(1)}$ whether there is a solution of a certain size for ANTIFACTOR_x assuming a tree decomposition of width tw is given.*

We note that Theorem 1.4 clearly distinguishes X -ANTIFACTOR from B -FACTOR. By the known lower bounds from Marx et al. [30] (cf. Theorem 1.1), a similar result for B -FACTOR is not possible. In light of Theorem 1.4, it is also far from obvious to determine, the exact complexity of X -ANTIFACTOR for a fixed set X . The combinatorial properties of the set X influence the complexity of the problem in a subtle way and new algorithmic techniques seem to be needed to fully exploit this. Currently, we do not have a tight bound similar to Theorem 1.1 for every fixed X . Instead we propose a candidate for the combinatorial property that influences the complexity: We define a bipartite compatibility graph for every set X and conjecture that the maximum size of a so-called *half-induced matching* is the key

¹ Recall that in a graph with treewidth tw , the largest bag has size $\text{tw} + 1$.

property to obtain a faster algorithm via representative sets. See Conjecture 4.5 for a formal statement.

We use such half-induced matchings of large size to show a lower bound for ANTIFACTOR_x that, assuming SETH , complements the algorithm in Theorem 1.4 up to constant factors in the exponent (see Theorem 5.5). Moreover, if there is a half-induced matching of size h , then, assuming SETH , we show that there is no $(h - \epsilon)^{\text{tw}} n^{\mathcal{O}(1)}$ algorithm for $X\text{-ANTIFACTOR}$ for any $\epsilon > 0$ (Theorem 5.4). Although, in this case the representative set cannot be smaller than $(h - \epsilon)^{\text{tw}+1}$ for any $\epsilon > 0$ (Lemma 4.4) we do not have matching upper bounds at this point. There are two main reasons why it is difficult to obtain tight upper bounds:

- **Representative set bounds.** In Theorem 1.4, the upper bound on the size of representative sets are based on earlier algebraic techniques [18, 19, 25, 35]. It is not clear how they can be extended to the combinatorial notion of half-induced matchings.
- **Join nodes.** Even if we have tight bounds on the size of representative sets there is an additional issue that can increase the running time. At join nodes of the tree decomposition, we need to compute from two representative sets a third one. Doing this operation in a naive way results in a running time that is at least the *square* of the bound on the size of the representative set. If we want to have a running time that matches the size of the representative set, we need a more clever way of handling join nodes.

Representative sets of the form we study here could be relevant for other problems and tight bounds for such representative sets could be of fundamental importance. In particular, the notion of half-induced matchings could be a key property in other contexts as well.

Counting Problems. We also investigate the $\#\text{ANTIFACTOR}$ problem, where we need to count the total number of solutions satisfying the degree constraints. The idea of representative sets is fundamentally incompatible with exact counting: if we need to count every solution, then we cannot ignore certain partial solutions even if they can be always replaced by others. Therefore, the algorithm of Theorem 1.4 cannot be extended to the counting version.² In fact, we show that already $\#\text{ANTIFACTOR}_1$ is unlikely to be FPT by showing the following stronger statement for path decompositions.

► **Theorem 1.5.** *There is a fixed constant c such that $\#\text{ANTIFACTOR}_1$ cannot be solved in time $\mathcal{O}(n^{\text{pw}-c})$ on graphs with n vertices given a path decomposition of width pw , unless $\#\text{SETH}$ is false. Furthermore, $\#\text{ANTIFACTOR}_1$ is $\#\text{W}[1]$ -hard parameterized by pathwidth.*

Recall that $\#\text{SETH}$ (cf. [11, 14]) is actually a weaker assumption than SETH . Hence, the first result is stronger than a version based on SETH . Moreover, the algorithm from Theorem 1.3 is essentially optimal for $\#X\text{-ANTIFACTOR}$.

► **Theorem 1.6.** *Let $X \subseteq \mathbb{N}$ be a non-empty, finite and fixed set. For any constant $\epsilon > 0$, there is no algorithm that can solve $\#X\text{-ANTIFACTOR}$ in time $(\max X + 2 - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ given a graph along with a path decomposition of width pw , unless $\#\text{SETH}$ fails.*

Organization. Section 2 presents the algorithm of Theorem 1.4 and Section 3 shows how to compute representative sets. Section 4 introduces half-induced matchings and discusses some combinatorial properties related to representative sets. Sections 5 and 6 present the lower bounds for the decision and counting versions, respectively.

² Counting the solutions approximately is a problem of independent interest.

2 Algorithms

In this section, we use without loss of generality “nice” tree decompositions that have *introduce edge nodes* (see, e.g., [12] for formal definitions). When given a node t of a tree decomposition, we denote by B_t the bag of t , by V_t the vertices introduced at the subtree rooted at t , and by E_t the edges introduced in the subtree rooted at t .

We leave the proof of Theorem 1.3 to the full version of this paper, as many of the details are similar to that in [30].

2.1 Parameterizing by the Number of Excluded Degrees

In this section we prove Theorem 1.4 which shows that ANTIFACTOR_x is FPT parameterized by treewidth and the *size* x of the set. We first show a naive algorithm, i.e. the standard dynamic programming approach, solving the problem. In a second step we improve this algorithm by using representative sets. That is, we do not store all solutions but only so much information such that we can correctly solve the decision and optimization version.

2.1.1 Naive Algorithm

Let $X_v \subseteq \mathbb{N}$ be the set assigned to vertex v with $|X_v| \leq x$. Let n be the number of vertices of G and m the number of edges. Let $U = [0, n]$ be the universe of the values in the following.

The idea is to fill a table $\text{ParSol}[\cdot, \cdot]$ with partial solutions. That is, for all nodes t of the tree decomposition with bag B_t of size k and all $s \in [0, m]$, we have $\text{ParSol}[t, s] \subseteq U^{B_t}$ and $a \in \text{ParSol}[t, s]$ if and only if there is a set $S \subseteq E_t$ with $|S| = s$ such that $\deg_S(v) \notin X_v$ for all $v \in V_t \setminus B_t$ and $\deg_S(v) = a[v]$ for all $v \in B_t$.

Dynamic Program. Initialize the table ParSol with \emptyset for every entry. We fill the table iteratively for all nodes t of the tree decomposition and all $s \in [0, m]$ in the following way, depending on s and the type of t .

Leaf Node. As $B_t = \emptyset$, we set $\text{ParSol}[t, 0] := \{\emptyset\}$.

Introduce Vertex Node. Assume v is introduced at t , i.e. $B_t = B_{t'} \cup \{v\}$. We define

$$\text{ParSol}[t, s] := \{a_{v \rightarrow 0} \mid a \in \text{ParSol}[t', s]\}.$$

Introduce Edge Node. Assume the edge $e = uv$ is introduced at the node t . We combine the cases where e is not selected for the solution and where e is selected. Thus, we define:

$$\text{ParSol}[t, s] := \text{ParSol}[t', s] \cup \{a_{u \rightarrow a(u)+1, v \rightarrow a(v)+1} \mid a \in \text{ParSol}[t', s-1]\}.$$

Forget Node. Assume vertex v is forgotten at t , i.e. $B_t = B_{t'} \setminus \{v\}$. We define

$$\text{ParSol}[t, s] := \{a|_{B_t} \mid a \in \text{ParSol}[t', s] : a[v] \notin X_v\}.$$

Join Node Assume t_1 and t_2 are the two children of t with $B_t = B_{t_1} = B_{t_2}$. Then we define

$$\text{ParSol}[t, s] := \{a_1 + a_2 \mid a_1 \in \text{ParSol}[t_1, s_1], a_2 \in \text{ParSol}[t_2, s_2], s_1 + s_2 = s\}.$$

Let r be the root of the tree decomposition with $B_r = \emptyset$. For a given $s \in [0, m]$, the algorithm finally checks if $\text{ParSol}[r, s] \neq \emptyset$, i.e. $\text{ParSol}[r, s]$ contains the empty vector. Otherwise no solution exists. The correctness of this algorithm follows directly from its definition. Note that the computation might take time $\Omega(n^{\text{tw}+1})$ since the largest bag has size $\text{tw} + 1$.

2.1.2 Improving the Naive Algorithm

The final algorithm is based on the naive algorithm but makes use of so-called representative sets to keep the size of the set stored for each node of the tree decomposition small.

We first define the notion of representative set to state the final algorithm. In Section 3 we show how to actually compute the representative sets.

► **Definition 2.1** (*H-Compatibility*). Let $H = (U \dot{\cup} V, E)$ be an undirected (potentially infinite) bipartite graph. We say that $a \in U$ is H -compatible with $b \in V$, denoted by $a \sim_H b$, if $(a, b) \in E$.³

Based on this compatibility notation, we define the H -representation of a set.

► **Definition 2.2** (*H-Representation*). Let $H = (U \dot{\cup} V, E)$ be an undirected (potentially infinite) bipartite graph. For any $\mathcal{S} \subseteq U$, we say that $\mathcal{S}' \subseteq V$ H -represents \mathcal{S} , denoted by $\mathcal{S}' \subseteq_{H\text{-rep}} \mathcal{S}$ if for every $b \in V$: $\exists a \in \mathcal{S} : a \sim_H b \iff \exists a' \in \mathcal{S}' : a' \sim_H b$.

For the algorithm we make use of this H -compatibility and H -representation where we use the following graphs.

► **Definition 2.3** (*Compatibility Graph*). For a set $B = \{v_1, \dots, v_k\}$ of k vertices with sets X_1, \dots, X_k of excluded degrees, we define the compatibility graph \mathcal{C}_B as follows:

- $V(\mathcal{C}_B) = U^k \dot{\cup} V^k$ where the elements in U, V are copies of numbers, i.e. $U, V = \mathbb{N}$.
- $E(\mathcal{C}_B) = \{((i_1, \dots, i_k), (j_1, \dots, j_k)) \mid \forall \ell \in [k], i_\ell + j_\ell \notin X_\ell\}$.

For a node t with bag B_t of the tree decomposition we denote by \mathcal{C}_t the graph \mathcal{C}_{B_t} .

The intuition is that the vertices in U^k represent the degrees of the constructed partial solution. The vertices in V^k correspond to the degrees of some (disjoint) partial solution one might see in the future. The edges then “check” whether both solutions can be combined, i.e. the degree of each vertex is valid with respect to the union of the solutions.

Final Algorithm. The improved algorithm applies the same operations as the naive algorithm to fill a table c . Then the algorithm computes a \mathcal{C}_t -representative set for the table entries and just stores these values in c . Only these values are used in the next steps to compute the other table entries. The correctness follows by induction on the tree decomposition.

▷ **Claim 2.4.** For all t, s : $c[t, s] \subseteq_{\mathcal{C}_t\text{-rep}} \text{ParSol}[t, s]$.

► **Lemma 2.5.** Assume there is an algorithm that can, for given $B = \{v_1, \dots, v_k\}$ with $|X_v| \leq x$ for all $v \in B$, compute for a set $\mathcal{S} \subseteq [0, n]^k$ a new set $\mathcal{S}' \subseteq_{\mathcal{C}_B\text{-rep}} \mathcal{S}$ of size $\text{Size}(k)$ in time $\text{Time}(k, |\mathcal{S}|)$, where Time and Size are allowed to depend on \mathcal{C}_B and x .

Then we can decide for a given ANTIFACTOR_x instance, whether there is a solution of size exactly s in time $\text{Time}(\text{tw}+1, (m+1) \cdot \text{Size}(\text{tw}+1)^2) n^{\mathcal{O}(1)}$ and $\text{Time}(\text{pw}+1, 2 \text{Size}(\text{pw}+1)) n^{\mathcal{O}(1)}$ given a tree and a path decomposition of width tw and pw , respectively.

Proof. We can assume that Time and Size are non-decreasing functions and inductively that the size of the given table entries is bounded by $\text{Size}(\text{tw}+1)$. The running time follows immediately by bounding the size of $c[t, s]$ and then computing its representative set. The correctness follows directly from Claim 2.4. ◀

³ Though the graph is undirected, we use tuples to denote the edges. By this the first value denotes the vertex from U and the second value the vertex from V .

3 Computing Representative Sets

As mentioned in the previous section, one can think of \mathcal{C}_t -compatibility as checking whether the given partial solution of degree a fits together with some partial solution of degree c arriving in the future. This is done via the bipartition of the compatibility graph and the (non-)existence of the edges, i.e. checking if $a + c$ is not in X . To compute the representative set we avoid this two step procedure by defining the more standard k - q -compatibility.

► **Definition 3.1** (k - q -Compatibility). *Let k, q be positive integers. For an $a \in \mathbb{N}^k$ and a $b \in \binom{\mathbb{N}}{q}^k$, we say a is k - q -compatible with b , denoted by $a \sim_q^k b$, if and only if for all $i \in [k]$ it holds that $a[i] \notin b[i]$.*

For our purposes we can relate the two compatibility definitions as follows: In \mathcal{C}_t -compatibility one computes $a + c$ and checks if $a + c \notin X$. Instead k - q -compatibility checks if $a \notin X - c$. While both checks are equivalent at this point, the new compatibility version considers *all* possible sets of size at most $q = |X|$ and not just $X - c$ for all c . Hence, k - q -compatibility is independent from the sets X_v which are assigned to the vertices v of the graph.

We extend the notion of compatibility in the standard way to k - q -representation.

► **Definition 3.2** (k - q -Representation). *Let k, q be positive integers. Given a set $\mathcal{S} \subseteq \mathbb{N}^k$, and a set $\mathcal{S}' \subseteq \mathcal{S}$. We say \mathcal{S}' k - q -represents \mathcal{S} , denoted by $\mathcal{S}' \subseteq_{q\text{-rep}}^k \mathcal{S}$, if and only if for all $b \in \binom{\mathbb{N}}{q}^k$: $\exists a \in \mathcal{S} : a \sim_q^k b \iff \exists a' \in \mathcal{S}' : a' \sim_q^k b$.*

For both notations we omit the value k from the notation if $k = 1$. It remains to check that k - q -compatibility generalizes \mathcal{C}_t -compatibility. This follows by folding and unfolding the definitions of the two types of compatibility.

► **Lemma 3.3.** *Let B be a set of k vertices where each $v \in B$ is assigned a set X_v such that $|X_v| \leq x$. Then the following holds for all $\mathcal{S}, \mathcal{S}' \subseteq \mathbb{N}^k$: If $\mathcal{S}' \subseteq_{x\text{-rep}}^k \mathcal{S}$, then $\mathcal{S}' \subseteq_{\mathcal{C}_B\text{-rep}} \mathcal{S}$.*

Matroids. For the computation of the representative sets we make use of uniform matroids. They allow us to formally state the operations we are using.

► **Definition 3.4** (Uniform Matroid). *Let U be some universe with n elements and $r \in \mathbb{N}$. Then $\mathcal{U}_{r,n} = (U, \binom{U}{\leq r})$ is the uniform matroid of rank r , that is the matroid over the ground set U and the independent sets are all subsets of U of size at most r .*

Later the rank of these uniform matroids corresponds to the number of excluded degrees (plus one). Since the matroid contains all subset of size at most the rank, we automatically consider all possibilities for upcoming solutions.

There are results proving the existence of small representative sets for matroids [18, 19, 25]. Since these results are usually for general matroids, they also apply to uniform matroids which we use here. However, as we are not considering a single matroid but the product of several matroids, the previous results can only be applied partially to our setting. Moreover, one can suspect that these results can be improved by exploiting properties of the uniform matroids. In the following we show one approach to compute the representative sets. A second method, not using matrix multiplication, is given in the full version of the paper [29]. That algorithm yields a slightly faster algorithm when parameterizing by pathwidth.

The Algorithm. We base our method on a previous result for computing representative sets. Despite the fact that the following lemma is a special case of Lemma 3.4 in [25], our proof uses a completely different technique as we exploit that the given matroids are uniform.

Let ω be the matrix multiplication coefficient in the following, i.e. $\omega < 2.37286$ [2].

► **Lemma 3.5.** *Let $\mathcal{M}_1, \dots, \mathcal{M}_k$ be k uniform matroids, each of rank r , with integer universes U_1, \dots, U_k . Given a set $\mathcal{S} \subseteq U_1 \times \dots \times U_k$ we can find a set $\mathcal{S}' \subseteq_{r-1\text{-rep}}^k \mathcal{S}$ of size r^k in time $\mathcal{O}(|\mathcal{S}| \cdot r^{k(\omega-1)} k)$.*

Proof Idea. We follow the ideas behind the proof of Theorem 12.15 in [12] where a variant of this lemma is shown for $k = 1$ with a general matroid.

Let M_1, \dots, M_k be $r \times |U|$ matrices representing the matroids $\mathcal{M}_1, \dots, \mathcal{M}_k$, which are known to exist. Enumerate all $I \in [r]^k$ in an arbitrary order I_1, \dots, I_{r^k} and compute for all $A \in \mathcal{S}$ the vector v_A , where for all $j \in [r^k]$ we set $v_A[j] = \prod_{i=1}^k M_i[I_j[i], A[i]]$. Construct a $r^k \times |\mathcal{S}|$ matrix Q with the vectors v_A as columns and find a column basis B_Q of Q . Output the set $\mathcal{S}' = \{A \mid v_A \in B_Q\}$ as solution. Since B_Q is a basis, it contains at most r^k elements.

Computing all v_A takes time $\mathcal{O}(|\mathcal{S}| \cdot r^k \cdot k)$ in total. As the computation of the basis takes time $\mathcal{O}(|\mathcal{S}| \cdot r^{k(\omega-1)})$, the complete procedure requires time $\mathcal{O}(|\mathcal{S}| \cdot r^{k(\omega-1)} \cdot k)$.

The basic idea of the correctness proof is to characterize the compatibility by a product of determinants of certain submatrices of each M_i . Then rewrite this by the Laplacian expansion using the I_j and exploit that B_Q is a basis. The formal proof is given in the full version as it is rather technical and does not give any insight into the problem. ◀

To finish the algorithm for ANTIFACTOR_x from Theorem 1.4, it remains, by Lemma 3.3, to compute a k - x -representative set as $|X_v| \leq x$. To achieve this we define k uniform matroids with universe $\{0, \dots, n\}$ and rank $x + 1$. Then, plugging in the values from Lemma 3.5 into Lemma 2.5, directly gives the following result. Note that we can assume $x \leq n$.

► **Corollary 3.6.** *Given a tree and a path decomposition, ANTIFACTOR_x can be solved in time $(x + 1)^{(\omega+1) \cdot \text{tw}} n^{\mathcal{O}(1)}$ and $(x + 1)^{\omega \cdot \text{pw}} n^{\mathcal{O}(1)}$, respectively.*

4 Half-Induced Matchings

In this section we introduce *half-induced matchings* and show relations to compatibility graphs and representative sets. We use these properties later in the lower bounds for the decision and optimization version of X - ANTIFACTOR and ANTIFACTOR_x . The proofs of all results in this section are given in Appendix A.

► **Definition 4.1** (Half-induced Matching). *Let $G = (U \cup V, E)$ be a bipartite graph. G has a half-induced matching of size ℓ if there are pairwise different $a_1, \dots, a_\ell \in U$ and pairwise different $b_1, \dots, b_\ell \in V$ such that (1) $(a_i, b_i) \in E$ for all i but (2) $(a_i, b_j) \notin E$ for all $j > i$.*

By an abuse of notation, \mathcal{C}_X denotes the compatibility graph for a vertex with set X of forbidden degrees. We show that arithmetic progressions in the set of excluded degrees are sufficient to obtain large half-induced matchings in the corresponding compatibility graph.

► **Lemma 4.2.** *If X contains an arithmetic progression of length ℓ , but not one of length $\ell + 1$, then \mathcal{C}_X has a half-induced matching of size $\ell + 1$.*

Conversely to the previous lemma, we also prove that arithmetic progressions are necessary to obtain large half-induced matchings.

► **Lemma 4.3.** *Let $X \subseteq \mathbb{N}$ with $|X| = \ell \geq 2$. Suppose \mathcal{C}_X contains a half-induced matching of size $\ell + 1$. Then X is an arithmetic progression.*

For a graph \mathcal{C} and an integer $k > 1$, we extend $\sim_{\mathcal{C}}$ and $\subseteq_{\mathcal{C}\text{-rep}}$ to k dimensions, denoted by $\sim_{\mathcal{C}}^k$ and $\subseteq_{\mathcal{C}\text{-rep}}^k$, such that the \mathcal{C} -compatibility must hold for each dimension.

► **Lemma 4.4.** *Let $X \subseteq \mathbb{N}$, and $\epsilon > 0$ and $\ell \geq 2$ be constants. Then there exists a constant k depending only on ϵ and ℓ such that the following holds. Suppose the compatibility graph \mathcal{C}_X contains a half-induced matching of size ℓ . Then there is a set $\mathcal{S} \subseteq \mathbb{N}^k$ such that every representative set $\mathcal{S}' \subseteq_{\mathcal{C}_X\text{-rep}}^k \mathcal{S}$ has size $|\mathcal{S}'| \geq (\ell - \epsilon)^k$.*

The proof is given in Appendix A but we briefly discuss its implications. The running time of the algorithm for X -ANTIFACTOR from Theorem 1.4 depends on the size of the representative sets computed. This lemma implies that any such algorithm using representative sets in a similar way takes time at least $(\ell - \epsilon)^{\text{pw}}$. This can be seen as an *unconditional* version of the lower bound for the decision version shown in Theorem 5.4.

We conjecture that the converse of the above lemma is also true. For example, for $X = \{10, 100, 1000, \dots\}$ the largest half-induced matching in \mathcal{C}_X is of size three, a constant, (even though X itself is infinite). Intuitively, the size of the representative set itself must be small because knowing any *two* forbidden degrees of a vertex in the future solution is enough for us to deduce the degree of the vertex in the partial solution.

► **Conjecture 4.5.** *Let $X \subseteq \mathbb{N}$ and $\ell \geq 2$ be a constant. Then there exists a constant k depending only on ℓ such that the following holds. Suppose the largest half-induced matching in \mathcal{C}_X has size ℓ . Then every $\mathcal{S} \subseteq \mathbb{N}^k$ has a representative set $\mathcal{S}' \subseteq_{\mathcal{C}_X\text{-rep}}^k \mathcal{S}$ with $|\mathcal{S}'| \leq \ell^{k+o(k)}$.*

Recall, the runtime of the algorithm in Theorem 1.3 depends on $\max X$ but the lower bound in Theorem 5.4 on the size ℓ of the half-induced matching. With the conjecture it seems reasonable to get algorithms for the decision and optimization version based on representative sets with a running time depending on ℓ . This would complement the lower bound. Note, for the counting version the algorithm is essentially optimal (Theorem 6.3).

5 Lower Bounds for the Decision Version

In this section we prove the lower bounds for the decision version of X -ANTIFACTOR and ANTIFACTOR_x . Instead of showing the lower bound directly, we first define the following intermediate problem and show the hardness of this problem.

► **Definition 5.1** (X -ANTIFACTOR $^{\mathcal{R}}$). *Let $X \subseteq \mathbb{N}$ be fixed and finite. Let $G = (V_S \cup V_C, E)$ be a vertex labeled graph such that*

- *all vertices in V_S , called simple vertices, are labeled with set X ,*
 - *all vertices $v \in V_C$, called complex vertices, are labeled with a relation R_v that is given as a truth table such that $R_v \subseteq 2^{I(v)}$ where $I(v)$ is the set of edges incident to v in G .*
- A set $\widehat{E} \subseteq E$ is a solution for G if (1) for $v \in V_S$: $\deg_{\widehat{E}}(v) \notin X$ and (2) for $v \in V_C$: $I(v) \cap \widehat{E} \in R_v$.*

X -ANTIFACTOR WITH RELATIONS (X -ANTIFACTOR $^{\mathcal{R}}$) *is the problem of deciding if such an instance G has a solution.*

We show our lower bounds based on this problem definition.

► **Lemma 5.2** (Lower Bound for X -ANTIFACTOR $^{\mathcal{R}}$). *Let $X \subseteq \mathbb{N}$ be a fixed set which contains a half-induced matching of size $h \geq 2$.*

22:10 Anti-Factor is FPT Parameterized by Treewidth and List Size (but Counting is Hard)

Let $f_X : \mathbb{N} \rightarrow \mathbb{R}^+$ be an arbitrary function that may depend on the set X .

For every constant $\epsilon > 0$, there is no algorithm that can solve X -ANTI FACTOR $^{\mathcal{R}}$ in time $(h - \epsilon)^{\text{pw} + f_X(\Delta^*)} n^{\mathcal{O}(1)}$, where $\Delta^* = \max_{\text{bag } B} \sum_{v \in B \cap V_C} \deg(v)$, even if we are given a path decomposition of width pw , unless SETH fails.

In a second step we remove the relations and replace them by appropriate gadgets. To be able to reuse the reduction later we introduce a slightly more general version of the problem. For two finite sets $X, Y \subseteq \mathbb{N}$, we define (X, Y) -ANTI FACTOR as the generalization of X -ANTI FACTOR where we allow the sets X and Y to be assigned to the vertices. We show hardness when $0 \in X$ and when $\max\text{-gap}(\overline{X}) > 1$. The former is to ensure there are no trivial solutions and the latter ensures that the problem is not polynomial-time solvable [10]. Recall that $\max\text{-gap}(\overline{X})$ is the size of the largest contiguous sequence of integers not in \overline{X} but whose boundaries are in \overline{X} .

► **Lemma 5.3.** *Fix a finite set $X \subseteq \mathbb{N}$ such that $0 \in X$ and $\max\text{-gap}(\overline{X}) > 1$. Let $Y \subseteq \mathbb{N}$ be arbitrary. There is a many-one reduction from Y -ANTI FACTOR $^{\mathcal{R}}$ to (X, Y) -ANTI FACTOR such that pathwidth increases by at most $f(\Delta^*)$ and size by a factor of $f(\Delta^*)$, where $\Delta^* = \max_{\text{bag } B} \sum_{v \in B \cap V_C} \deg(v)$.*

The proof essentially follows a similar approach as the one for the hardness of B -FACTOR given in [30]. However, we deal with the cofinite set \overline{X} , and thus the constructions do not carry over directly. We do a careful check of their constructions and give necessary modifications in the full version.

By combining the lower bound for the intermediate problem with this reduction, we can show the lower bounds for X -ANTI FACTOR and ANTI FACTOR $_x$.

► **Theorem 5.4 (Lower Bound for Decision Version I).** *Fix a finite set $X \subseteq \mathbb{N}$ such that*

- $0 \in X$ and $\max\text{-gap}(\overline{X}) > 1$,
- and X contains a half-induced matching of size h .

For every constant $\epsilon > 0$, there is no algorithm that can solve X -ANTI FACTOR in time $(h - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ even if we are given a path decomposition of width pw , unless SETH fails.

Proof (Sketch). For a given X -ANTI FACTOR $^{\mathcal{R}}$ instance we apply Lemma 5.3 with $X = Y$ to obtain the (X, Y) -ANTI FACTOR. When applying the fast algorithm to it, one can easily check that this would contradict SETH by Lemma 5.2. ◀

The following theorem extends the previous result to the more general ANTI FACTOR $_x$ and shows a more informative lower bound.

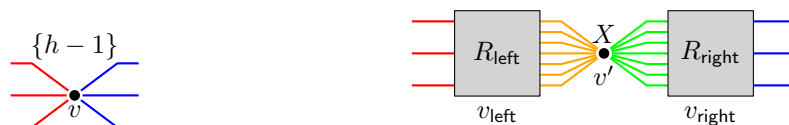
► **Theorem 5.5 (Lower Bound for Decision Version II).** *For all $x \geq 3, \epsilon > 0$, ANTI FACTOR $_x$ cannot be solved in time $(x + 1 - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ on graphs given with a path decomposition of width pw , unless SETH fails.*

Proof. We set $Y := \{2, 4, \dots, 2x\}$ and $X := \{0, 2, 3\}$. By Lemma 4.2, Y contains a half-induced matching of size $x + 1$. Moreover, \overline{X} contains a gap of size two.

We use Lemma 5.3 to transform a Y -ANTI FACTOR $^{\mathcal{R}}$ instance into an (X, Y) -ANTI FACTOR instance. From $|X|, |Y| \leq x$ and by the properties of X and Y , the claim follows directly. ◀

5.1 Replacing Finite Sets by Cofinite Sets

In this section we prove Lemma 5.2, i.e. the lower bound for X -ANTI FACTOR $^{\mathcal{R}}$, based on a lower bound from [30] for the following intermediate problem.



(a) The simple vertex v before the modifications. (b) The gadget replacing vertex v .

■ **Figure 1** The transformation in the proof of Lemma 5.2. The red, orange, green, and blue edges represent the left-external, left-internal, right-internal, and right-external edges, respectively.

► **Definition 5.6** (B -FACTOR ^{\mathcal{R}} (Simplified Definition 4.1 in [30])). Let $B \subseteq \mathbb{N}$ be fixed of finite size. B -FACTOR WITH RELATIONS (B -FACTOR ^{\mathcal{R}}) is the variation of X -ANTIFACTOR ^{\mathcal{R}} , cf. Definition 5.1, where the simple vertices are not labeled with set X but with set B .

A set $\hat{E} \subseteq E(G)$ is a solution for G if $\deg_{\hat{E}}(v) \in B$ for all simple vertices v and the relations of the complex vertices are satisfied.

We use the following lower bound and the restrictions to the graph as a starting point for our construction.

► **Lemma 5.7** (Corollaries 4.7 and 4.8 in the full version of [30]). Let $B \subseteq \mathbb{N}$ be a fixed and finite set. Given a B -FACTOR ^{\mathcal{R}} instance

- and its path decomposition of width pw with $\Delta^* = \max_{\text{bag } B} \sum_{v \in B \cap V_C} \deg(v)$,
- moreover all simple vertices are only connected to 2 complex nodes by exactly $\max B$ (parallel) edges each,
- and we are given the promise that with respect to any solution the degree of the simple vertices is exactly $\max B$.

Assume B -FACTOR ^{\mathcal{R}} can be solved in such a case in $(\max B + 1 - \epsilon)^{\text{pw} + f_B(\Delta^*)} n^{\mathcal{O}(1)}$ time for some $\epsilon > 0$ and some function $f_B : \mathbb{N} \rightarrow \mathbb{R}^+$ that may depend on the set B . Then SETH fails. Moreover the result also holds for $\#B$ -FACTOR ^{\mathcal{R}} and $\#SETH$.

To show a lower bound for X -ANTIFACTOR ^{\mathcal{R}} , it suffices to replace the simple vertices with set B by an appropriate gadgets consisting of simple vertices with set X and complex vertices.

Modification of the Graph. Let a_0, \dots, a_{h-1} and b_0, \dots, b_{h-1} be the labels of the half-induced matching of size h of X and U be the maximum over these labels. Let H be a B -FACTOR ^{\mathcal{R}} instance as stated in Lemma 5.7 with $\max B = h - 1$.⁴ We replace each simple vertex by the following gadget and keep the other vertices unchanged (see Figure 1).

By assumption, each simple vertex v is incident to $2(h - 1)$ edges which we can partition into two sets of size $h - 1$ depending on their endpoints. We call these groups of edges the left-external and right-external edges. We remove v and connect the left-external edges to a new complex vertex v_{left} with relation R_{left} . The right-external edges are connected similarly to another new complex vertex v_{right} with relation R_{right} . As a last step we create a new simple vertex v' with set X . We connect v' by U (parallel) edges to v_{left} , call these edges the left-internal edges, and by U parallel edges to v_{right} , call these edges the right-internal edges.

The relation R_{left} accepts if and only if, for some $i \in [0, h - 1]$, exactly i left-external and exactly a_{h-1-i} left-internal edges are selected. Similarly, R_{right} accepts if and only if, for some $j \in [0, h - 1]$, exactly b_j right-internal and exactly j right-external edges are selected.

We claim that the above replacement does not change the existence of solutions. For this we show that the number of selected left-external edges plus the number of selected

⁴ It actually suffices to set $B = \{h - 1\}$.

right-external edges is at most $h - 1$ for each such modification. Then, by the properties in Lemma 5.7, they sum to exactly $h - 1$ selected edges.

If i left-external edges are selected, then v' is incident to a_{h-1-i} selected left-internal edges, by definition of R_{left} . As R_{right} rejects when v_{right} is incident to exactly k selected right-internal edges where $k \neq b_j$ for all j , vertex v' must be incident to b_j right-internal edges for some j . By the definition of the half-induced matching we get $a_{h-1-i} + b_j \in X$ if $j > h - 1 - i$. Thus, some $b_{h-1-i-i'}$ with $h - 1 - i \geq i' \geq 0$ must be chosen. The relation R_{right} maps the $b_{h-1-i-i'}$ selected right-internal edges to $h - 1 - i - i'$ selected right-external edges. Thus, the gadget is incident to $i + (h - 1 - i - i') = h - 1 - i' \leq h - 1$ edges in total.

As v was only adjacent to complex vertices, we can merge the complex vertices v_{left} and v_{right} with the existing complex vertices and thus also the corresponding relations.

We analyze how the size and the pathwidth change. Replacing the simple vertices by the gadget does not change the pathwidth of the graph but only increases the degree of the complex vertices (due to the merging of the relations). Hence, Δ^* increases to $\Delta^* \cdot U$. As U only depends on the set X , it can be bounded by $\hat{f}(\max X)$ for some function \hat{f} .

Proof of Lemma 5.2 (Sketch). For a given B -FACTOR $^{\mathcal{R}}$ instance with $B = \{h - 1\}$, apply the above construction to obtain an X -ANTIFACTOR $^{\mathcal{R}}$ instance G . The total degree increases only by a factor depending on X , which is a constant as X is fixed. When running the claimed algorithm on G we contradict SETH by Lemma 5.7. ◀

6 Lower Bounds for the Counting Version

In this section we prove the two lower bounds for the counting version. While the lower bound for the decision and maximization version of X -ANTIFACTOR rely on half-induced matching, we avoid this dependence for $\#X$ -ANTIFACTOR by using interpolation techniques. This allows us to show a tight lower bound compared to the running time of the algorithm from Theorem 1.3. For the case when $X = \{0\}$, that is $\#$ EDGECOVER, we show a completely independent but also tight lower bound in the full version of this paper.

We also parameterize by the size of the set, i.e. by x . We design a new construction to prove the $\#W[1]$ -hardness of $\#$ ANTIFACTOR $_x$, even if $x = 1$, when parameterizing by treewidth. Hence, the problem is most likely not fixed-parameter tractable.

Both bounds use the same two-step approach as for the decision and optimization version; we first show the hardness of an intermediate problem which uses arbitrary relations and then remove these relations by a chain of reductions to obtain the actual lower bounds.

Parameterizing by the Maximum of the Set. We first show a lower bound for the intermediate problem $\#X$ -ANTIFACTOR $^{\mathcal{R}}$, which is the counting version of X -ANTIFACTOR $^{\mathcal{R}}$.

► **Lemma 6.1** (Lower Bound for $\#X$ -ANTIFACTOR $^{\mathcal{R}}$). *Let $X \subseteq \mathbb{N}$ be a fixed, non-empty and finite set. Let $f_X : \mathbb{N} \rightarrow \mathbb{R}^+$ be an arbitrary function that may depend on the set X .*

For every constant $\epsilon > 0$, there is no algorithm that can solve $\#X$ -ANTIFACTOR $^{\mathcal{R}}$ in time $(\max X + 2 - \epsilon)^{\text{pw} + f_X(\Delta^)} n^{\mathcal{O}(1)}$, where $\Delta^* = \max_{\text{bag } B} \sum_{v \in B \cap V_C} \deg(v)$, even if we are given a path decomposition of width pw , unless $\#$ SETH fails.*

We make use of the following lemma to remove the relations. We extend the definition of (X, Y) -ANTIFACTOR in the natural way to the counting version $\#(X, Y)$ -ANTIFACTOR.

► **Lemma 6.2.** *Let $X \subseteq \mathbb{N}$ be a finite set such that $X \not\subseteq \{0\}$. Let $Y \subseteq \mathbb{N}$ be arbitrary (possibly be given as input).*

There is a Turing reduction from $\#Y\text{-ANTIFACTOR}^{\mathcal{R}}$ to $\#(X, Y)\text{-ANTIFACTOR}$ increasing the size from n to $n \cdot f(\max X)$, decreasing Δ^* to zero, and increasing pw to $\text{pw} + \Delta^* \cdot f(\max X)$.

As for the decision version, the proof of this lemma is based on the reductions in [30] for the counting version of $B\text{-FACTOR}$. The reduction makes use of the Holant framework [8, 9, 20, 22, 24, 28] to formally relate the different steps of the reduction. See Appendix B for the main ideas behind the proof.

Combining these two lemmas, we can prove the first lower bound for the counting version.

► **Theorem 6.3** (Lower Bound for Counting Version I). *Let $X \subseteq \mathbb{N}$ be a finite and fixed set such that $X \not\subseteq \{0\}$. For every constant $\epsilon > 0$, there is no algorithm that can solve $\#X\text{-ANTIFACTOR}$ in time $(\max X + 2 - \epsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ even if we are given a path decomposition of width pw , unless $\#SETH$ fails.*

Proof (Sketch). After applying Lemma 6.2 where we set $X = Y$, the algorithm directly contradicts $\#SETH$ by Lemma 6.1. ◀

Parameterizing by the Size of the Set. If we do not fix the set X but only the *size* of the set, the decision and optimization version of ANTIFACTOR_x are still FPT parameterized by treewidth. For $\#\text{ANTIFACTOR}_x$ the following result conditionally rules out such algorithms.

► **Lemma 6.4.** *There exists a constant c such that there is no $\mathcal{O}(n^{p-c})$ algorithm for $\#\text{ANTIFACTOR}_1^{\mathcal{R}}$ on n -vertex graphs with $\Delta^* \in \mathcal{O}(1)$, even if only one set is used and we are given a path decomposition of width p , unless $\#SETH$ is false.*

Combined with Lemma 6.2 to remove the relations, we get the following hardness result.

► **Theorem 6.5** (Lower Bound for Counting Version II). *There exists a constant c such that there is no $\mathcal{O}(n^{p-c})$ algorithm for $\#\text{ANTIFACTOR}_1$ on n -vertex graphs, even if we are given a path decomposition of width p , unless $\#SETH$ is false.*

Proof. Let G be a given $\#\text{ANTIFACTOR}_1^{\mathcal{R}}$ instance which uses just one set $Y \subseteq \mathbb{N}$ with $|Y| = 1$. Use Lemma 6.2 with $X = \{2\}$ to transform G into a $\#\text{ANTIFACTOR}_1$ instance H . The remaining part of the proof follows in a standard manner by using Lemma 6.4. ◀

We prove Lemma 6.4 by a reduction from the $\#W[1]$ -hard problem $\text{COUNTING COLORFUL HITTING } k\text{-SETS}$. Hence, the following result holds by applying Lemma 6.2 as before.

► **Theorem 6.6.** *$\#\text{ANTIFACTOR}_1$ is $\#W[1]$ -hard.*

6.1 High-level Construction for SETH Lower Bound

We show the hardness of $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ by a reduction from $\#B\text{-FACTOR}^{\mathcal{R}}$, that is, we prove Lemma 6.1. As for the decision and optimization version, we mainly have to modify the simple vertices to take care of the new set. We design this reduction in three steps.

- The first step is a lower bound for the *relation-weighted* version of $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ (cf. Lemma 6.7). For this problem we assign each accepted input of a relation a weight. Then an accepted input contributes by this weight to the solution. The weight of the solution is the product of the weights of the relations. The unweighted problem can be seen as assigning weight 1 to all accepted inputs and weight 0 to all rejected inputs.
- Then in a second step we reduce this problem to the *edge-weighted* version where the edges are assigned weights by which they contribute to the solution if they are selected (cf. Lemma 6.8). Again, the weight of a solution is the product of the weights of the selected edges.

- The last step then removes these edge-weights by an appropriate Turing-reduction using the technique of interpolation (cf. Lemma 6.9).

We use the Holant framework (cf. [8, 9, 20, 22, 24, 28]) to contextualize several versions of #ANTI-FACTOR. In the Holant framework, we are given a signature graph $\Omega = (V, E)$, where every edge $e \in E$ has a weight w_e . Every vertex $v \in V$ is labeled with a signature $f_v : \{0, 1\}^{I(v)} \rightarrow \mathbb{Q}$, where $I(v)$ is the incidence vector of edges incident to v . A *solution* is a subset of edges such that the signature for each vertex is non-zero. The weight of a solution is the product of the weights of all edges in the subset and the signatures of all the vertices. Then the Holant of Ω is defined as the sum of the weights of all solutions.

$$\text{Holant}(\Omega) = \sum_{x \in \{0,1\}^{E(\Omega)}} \prod_{e \in x} w_e \prod_{v \in V(\Omega)} f_v(x|_{I(v)}).$$

The Holant problem is easily seen to be a weighted generalization of the counting versions of GENFAC and ANTI-FACTOR. For example, the problem #X-ANTI-FACTOR is a Holant problem on unweighted graphs where every vertex has the following symmetric relation.

$$f(z) = \begin{cases} 1 & \text{if } \text{hw}(z) \notin X \\ 0 & \text{if } \text{hw}(z) \in X \end{cases}$$

where $\text{hw}(\cdot)$ is the Hamming weight operator. We call vertices with such functions to be $\text{HW}_{\in X}$ nodes.

For relations R_1, \dots, R_k , we define $\text{Holant}(R_1, \dots, R_k)$ to be the set of Holant problems where every edge is unweighted and every vertex has signature R_j for some $j \in [k]$. By an abuse of notation also let R_j be a *family* of relations. For example, we may use $\text{Holant}(\text{HW}_{=1})$ when every vertex has relation $\text{HW}_{=1}^{(k)}$ for some k .

The relation-weighted version of #X-ANTI-FACTOR^R corresponds to a variant of the Holant problem where all edges have weight 1. Likewise the edge-weighted version of #X-ANTI-FACTOR^R corresponds to a variant of the Holant problem where we require that the value of the signatures is either 0 or 1, i.e. they accept or reject.

The Holant framework was extensively used in proving the #SETH lower bounds for COUNTING PERFECT MATCHINGS [11] and COUNTING GENERAL FACTORS [30]. We make use of some of their constructions to prove the following lower bound.

► **Lemma 6.7.** *Let $X \subseteq \mathbb{N}$ be a fixed, non-empty and finite set. Let $f_X : \mathbb{N} \rightarrow \mathbb{R}^+$ be an arbitrary function that may depend on the set X .*

For every constant $\epsilon > 0$, there is no algorithm that can, even if we are given a path decomposition of width pw , solve relation-weighted #X-ANTI-FACTOR^R with $(\max X + 2)^2$ weights in time $(\max X + 2 - \epsilon)^{\text{pw} + f_X(\Delta^)} n^{\mathcal{O}(1)}$, where $\Delta^* = \max_{\text{bag } B} \sum_{v \in B \cap V_G} \deg(v)$, unless #SETH fails.*

We prove the lower bound by a reduction from #B-FACTOR^R where $B = \{\max X + 1\}$. When treating the given #B-FACTOR^R instance G as a #X-ANTI-FACTOR^R instance H , all solution of G are also a solution for H because of our choice of B . The converse is not true: As X is finite (and thus the set of allowed degree is cofinite), the degree of the solutions for H can be different from $\max X + 1$ (possibly larger or even smaller). We construct a gadget such that the number of selected incident edges is equal to $\max X + 1$ and the solution is also valid for the B-FACTOR^R instance.

For this we add a weighted relation directly after the simple vertices. We choose the weights such that the entire gadget behaves as the original vertex with set B . For this we exploit the fact that the set X does not allow certain combinations of selected incident edges.

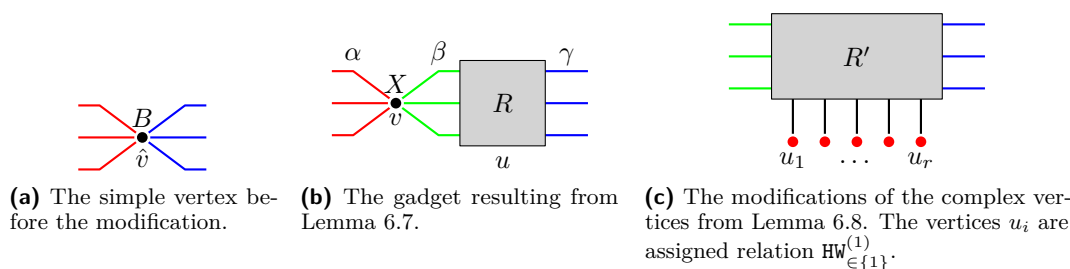


Figure 2 The modifications of the vertices in the different steps of the reductions.

Proof of Lemma 6.7. We start with a $\#B\text{-FACTOR}^{\mathcal{R}}$ instance H as stated in Lemma 5.7 where $B = \{\max X + 1\}$ and apply the following transformation, illustrated in Figures 2a and 2b, for each simple vertex \hat{v} .

Transformation. By assumption, the incident edges of \hat{v} can be split into *left* and *right* edges (depending on their endpoints). We remove \hat{v} and create a simple vertex v and a complex vertex u . Connect the left edges to v and the right edges to u . We connect v and u by $\max X + 1$ parallel edges which we call *middle* edges.⁵ We assign the set X to v and the relation R to u , which is defined as follows.

R requires that the selection of middle and right edges is monotone. That is the first k edges are selected and the last $\max X + 1 - k$ edges are not selected. Then for all $\beta, \gamma \in [0, \max X + 1]$, R accepts β middle and γ right edges with weight $w_{\beta, \gamma}$.

To define the weights $w_{\beta, \gamma}$, let $g(\alpha, \gamma)$ denote the signature of the whole gadget (including v and u) when α left and γ right edges are selected.⁶ Based on our construction we get:

$$g(\alpha, \gamma) = \sum_{\substack{\beta=0 \\ \beta+\alpha \notin X}}^{\max X+1} w_{\beta, \gamma}.$$

To simulate the original simple vertex \hat{v} with set $B = \{\max X + 1\}$, we need $g(i, \max X + 1 - i) = 1$ for all $i \in [0, \max X + 1]$ and 0 otherwise. These constraints and the definition of g describe a system of linear equations with $(\max X + 2)^2$ variables and equally many constraints. Assuming that there always exists a solution, we can use Gaussian elimination to compute the solution in time $\mathcal{O}(n^3)$. Since the weights are chosen such that the remaining graph does not see a difference between the vertex \hat{v} and this new gadget, we can replace all simple vertices by this procedure to get a relation-weighted $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ instance.

It remains to prove that such a solution always exists. For a fixed γ the values of $g(0, \gamma), \dots, g(\max X + 1, \gamma)$ depend only on the weights $w_{0, \gamma}, \dots, w_{\max X + 1, \gamma}$. Moreover, these weights do not appear in the sum of any $g(i, \gamma')$ where $\gamma' \neq \gamma$. Hence, we can treat each possible value of γ separately.

For a fixed γ , the sums for $g(\alpha, \gamma)$ and $g(\alpha + 1, \gamma)$ differ by (at least) one summand, i.e. $w_{\max X - \alpha, \gamma}$. When starting with the sums for $\max X + 1$ and $\max X$, we can eliminate $w_{0, \gamma}$ from the system of linear equations. Then we can repeat this process to eliminate $w_{1, \gamma}$ up to $w_{\max X + 1, \gamma}$. Hence, there is a solution for a fixed γ .

⁵ Though these parallel edges disappear later, one could place EQ_2 nodes on them to obtain a simple graph.

⁶ Note that a signature is normally defined on subsets of edges. As we require the selection of the edges to be monotonous, we also use “signature” to refer to g .

22:16 Anti-Factor is FPT Parameterized by Treewidth and List Size (but Counting is Hard)

Lower Bound. Let G be the final $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ instance. We have $n_G \leq 2n_H$, $\Delta_G^* \leq 2\Delta_H^*$, and $\text{pw}_G \leq \text{pw}_H + \Delta_H^*$. Assume we run a fast algorithm for relation-weighted $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ on the new instance. Then, one can easily check that this directly contradicts $\#\text{SETH}$ by Lemma 5.7. \blacktriangleleft

The next step of our reduction removes the weights from the relations by using weighted edges.

► **Lemma 6.8.** *Let $X \subseteq \mathbb{N}$ be an arbitrary set (possibly given as input).*

We can many-one reduce relation-weighted $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ with r different weights to edge-weighted $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ such that

- *the r weights do not change,*
- *the size increases by a multiplicative factor of $\mathcal{O}(r)$,*
- *Δ^* increases to $\Delta^* + r + 1$,*
- *and pw increases to $\text{pw} + 1$.*

Proof. We apply the following procedure to each complex vertex u . See Figure 2c for the modification. For this fix some u and let R be its relation. Let $w_1, \dots, w_{r'}$ be the $r' < r$ different weights used by R . Assume w.l.o.g. that $r' = r$.

For all $i \in [r]$, we add a vertex u_i with relation $\text{HW}_{\in\{0,1\}}^{(1)}$ and make it adjacent to v by an edge of weight w_i .

Based on R we design a new relation R' as follows: Whenever R accepts the input x with weight w_i for some $i \in [r]$, then R' accepts x but additionally requires that the edge to u_i is selected while the edges to the other $u_{i'}$ remain unselected.

One can easily check that this modification does not change the solution. Moreover, the pathwidth increases by at most 1 and the degree of the complex nodes by at most r . \blacktriangleleft

In the next step of our reduction we remove the edge weights from the graph. First observe that we do not have to change edges of weight 1. Furthermore, we can simply remove all edges with weight 0.

► **Lemma 6.9.** *Let $X \subseteq \mathbb{N}$ be a finite and non-empty set (possibly given as input). There is a Turing reduction from edge-weighted $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ with r different weights to unweighted $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ running in time $n^{\mathcal{O}(r)}$. The reduction is such that*

- *the size increases by a multiplicative factor of $\mathcal{O}(\log^2(n))$,*
- *the degree of the simple vertices stays the same,*
- *Δ^* increases to $\Delta^* + \mathcal{O}(1)$,*
- *and pw increases to $\text{pw} + \mathcal{O}(1)$.*

The proof of the lemma uses the same interpolation techniques already used in [11] and later in [30] to remove the edge weights. Now we can combine the previous steps and prove the lower bound for $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$.

Proof of Lemma 6.1 (Sketch). For a given relation-weighted $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ instance H we apply Lemmas 6.8 and 6.9 to obtain polynomially many $\#X\text{-ANTIFACTOR}$ instances. It is easy to check that a faster algorithm for $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ would contradict $\#\text{SETH}$ by Lemma 6.7. \blacktriangleleft

References

- 1 Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, and Saket Saurabh. Parameterized complexity of conflict-free matchings and paths. *Algorithmica*, 82(7):1939–1965, 2020. doi:10.1007/s00453-020-00681-y.
- 2 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021. doi:10.1137/1.9781611976465.32.
- 3 Hans L Bodlaender. Dynamic programming on graphs with bounded treewidth. In *International Colloquium on Automata, Languages, and Programming*, pages 105–118. Springer, 1988.
- 4 Hans L Bodlaender. Treewidth: Algorithmic techniques and results. In *International Symposium on Mathematical Foundations of Computer Science*, pages 19–36. Springer, 1997.
- 5 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.
- 6 Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008. doi:10.1093/comjnl/bxm037.
- 7 Édouard Bonnet, Nick Brettell, O-joung Kwon, and Dániel Marx. Generalized feedback vertex set problems on bounded-treewidth graphs: Chordality is the key to single-exponential parameterized algorithms. *Algorithmica*, 81(10):3890–3935, 2019. doi:10.1007/s00453-019-00579-4.
- 8 Jin-yi Cai, Sangxia Huang, and Pinyan Lu. From Holant to #CSP and back: Dichotomy for Holant^c problems. In Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park, editors, *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*, volume 6506 of *Lecture Notes in Computer Science*, pages 253–265. Springer, 2010. doi:10.1007/978-3-642-17517-6_24.
- 9 Jin-yi Cai, Pinyan Lu, and Mingji Xia. A computational proof of complexity of some restricted counting problems. *Theor. Comput. Sci.*, 412(23):2468–2485, 2011. doi:10.1016/j.tcs.2010.10.039.
- 10 Gérard Cornuéjols. General factors of graphs. *J. Comb. Theory, Ser. B*, 45(2):185–198, 1988. doi:10.1016/0095-8956(88)90068-8.
- 11 Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669. SIAM, 2016. doi:10.1137/1.9781611974331.ch113.
- 12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 13 Víctor Dalmau and Daniel K. Ford. Generalized satisfiability with limited occurrences per variable: A study through delta-matroid parity. In Branislav Rován and Peter Vojtás, editors, *Mathematical Foundations of Computer Science 2003, 28th International Symposium, MFCS 2003, Bratislava, Slovakia, August 25-29, 2003, Proceedings*, volume 2747 of *Lecture Notes in Computer Science*, pages 358–367. Springer, 2003. doi:10.1007/978-3-540-45138-9_30.
- 14 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.
- 15 Szymon Dudycz and Katarzyna Paluch. Optimal general matchings. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science - 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159 of *Lecture Notes in Computer Science*, pages 176–189. Springer, 2018. Full version: arXiv:1706.07418. doi:10.1007/978-3-030-00256-5_15.

- 16 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.
- 17 Eduard Eiben and Iyad Kanj. A colored path problem and its applications. *ACM Trans. Algorithms*, 16(4):47:1–47:48, 2020. doi:10.1145/3396573.
- 18 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.
- 19 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative families of product families. *ACM Trans. Algorithms*, 13(3):36:1–36:29, 2017. doi:10.1145/3039243.
- 20 Heng Guo and Pinyan Lu. On the complexity of holant problems. In Andrei A. Krokhin and Stanislav Zivný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 159–177. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/DFU.Vol7.15301.6.
- 21 John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.
- 22 Sangxia Huang and Pinyan Lu. A dichotomy for real weighted holant problems. *Comput. Complex.*, 25(1):255–304, 2016. doi:10.1007/s00037-015-0118-3.
- 23 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.
- 24 Michael Kowalczyk and Jin-Yi Cai. Holant problems for 3-regular graphs with complex edge functions. *Theory Comput. Syst.*, 59(1):133–158, 2016. doi:10.1007/s00224-016-9671-7.
- 25 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. doi:10.1145/3390887.
- 26 L. Lovász and M. D. Plummer. *Matching Theory*. North-Holland Publishing Co., Amsterdam, 1986. Annals of Discrete Mathematics, 29.
- 27 László Lovász. The factorization of graphs. II. *Acta Mathematica Hungarica*, 23(1-2):223–246, 1972.
- 28 Pinyan Lu. Complexity dichotomies of counting problems. *Electron. Colloquium Comput. Complex.*, 18:93, 2011. URL: <http://eccc.hpi-web.de/report/2011/093>.
- 29 Dániel Marx, Govind S. Sankar, and Philipp Schepper. Anti-factor is FPT parameterized by treewidth and list size (but counting is hard). *CoRR*, abs/2110.09369, 2021. URL: <https://arxiv.org/abs/2110.09369>, arXiv:2110.09369.
- 30 Dániel Marx, Govind S. Sankar, and Philipp Schepper. Degrees and gaps: Tight complexity results of general factor problems parameterized by treewidth and cutwidth. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 95:1–95:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. Full version: arXiv:2105.08980. doi:10.4230/LIPICs.ICALP.2021.95.
- 31 Dániel Marx and Paul Wollan. An exact characterization of tractable demand patterns for maximum disjoint path problems. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 642–661. SIAM, 2015. doi:10.1137/1.9781611973730.44.
- 32 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{(|V|) |E|})$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 17–27. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.12.
- 33 Burkhard Monien. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*, volume 109 of *North-Holland Math. Stud.*, pages 239–254. North-Holland, Amsterdam, 1985.
- 34 András Sebő. General antifactors of graphs. *J. Comb. Theory, Ser. B*, 58(2):174–184, 1993.

- 35 Hadas Shachnai and Meirav Zehavi. Representative families: A unified tradeoff-based approach. *J. Comput. Syst. Sci.*, 82(3):488–502, 2016. doi:10.1016/j.jcss.2015.11.008.
- 36 Johan M. M. van Rooij. Fast algorithms for join operations on tree decompositions. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 262–297. Springer, 2020. doi:10.1007/978-3-030-42071-0_18.

A

Omitted Proofs from Section 4

We first proof that arithmetic progressions in the set of excluded degrees are sufficient to obtain large half-induced matchings in the corresponding compatibility graph.

Proof of Lemma 4.2. Let $a, a + d, a + 2d, \dots, a + (\ell - 1)d \in X$ be an arithmetic progression with $d \geq 1$ such that $a + \ell d \notin X$. We construct the following half-induced matching in \mathcal{C}_X where, for all $i \in [\ell + 1]$, we set $a_i := d(i - 1)$ and $b_i := a + (\ell + 1 - i)d$.

Then for all $i \in [\ell + 1]$ we have $a_i + b_i = d(i - 1) + a + (\ell + 1 - i)d = a + \ell d \notin X$ and hence $(a_i, b_i) \in E(\mathcal{C}_X)$. Similarly, for all $i \in [\ell]$ and all $i < j \in [\ell + 1]$, we have $(a_i, b_j) \notin E(\mathcal{C}_X)$ because $a_i + b_j = d(i - 1) + a + (\ell + 1 - j)d = a + (\ell + i - j)d \in X$. \blacktriangleleft

Now we show the converse of the above result. That is, arithmetic progressions are necessary to obtain large half-induced matchings.

Proof of Lemma 4.3. Let $a_1, \dots, a_{\ell+1}$ and $b_1, \dots, b_{\ell+1}$ be the vertices of the half-induced matching of size $\ell + 1$ in \mathcal{C}_X . Then we have the following constraints:

$$\begin{array}{cccc} a_1 + b_2 \in X, & a_1 + b_3 \in X, & \dots, & a_1 + b_{\ell+1} \in X \\ & a_2 + b_3 \in X, & \dots, & a_2 + b_{\ell+1} \in X \end{array}$$

Let $X = \{x_1, x_2, \dots, x_\ell\}$ where $x_i \leq x_{i+1}$. From $a_1 + b_j \neq a_1 + b_{j'}$ for any $j \neq j'$, we get

$$\{a_1 + b_2, a_1 + b_3, \dots, a_1 + b_{\ell+1}\} = X.$$

Now consider the second set of constraints. Assuming $a_2 - a_1 = d > 0$, we have

$$\{a_2 + b_3, a_2 + b_4, \dots, a_2 + b_{\ell+1}\} = X \setminus \{x_i\}$$

for some i . Since $d > 0$, we get $x_\ell + d \notin X$ and thus

$$\{x_1 + d, x_2 + d, x_3 + d, \dots, x_{\ell-1} + d\} = X \setminus \{x_i\}.$$

Now further observe that x_1 cannot belong to the left-hand side because $d > 0$ and $x_1 = \min(X)$. Thus, we have that $i = 1$. Similarly, when $a_2 - a_1 = d < 0$ we can argue that $i = \ell$. Without loss of generality consider the former case. Then we have that $x_i + d = x_{i+1}$ for all $i \in [\ell]$. Hence, X is an arithmetic progression of length ℓ . \blacktriangleleft

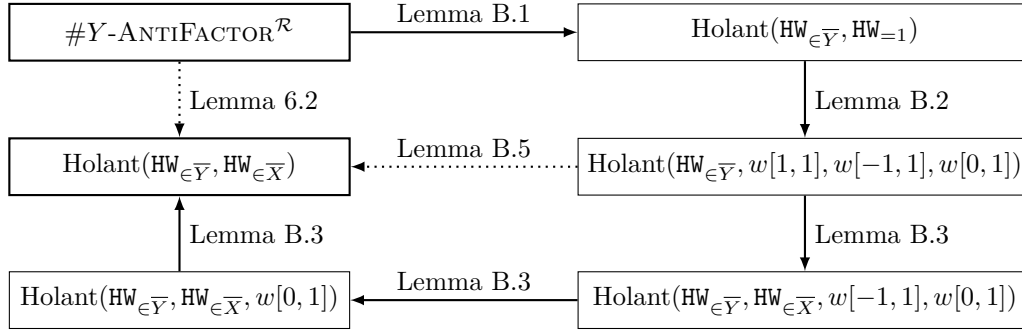
As the last part of this section, we prove Lemma 4.4 which states that a half-induced matching implies a lower bound for the size of the representative set.

Proof of Lemma 4.4. We set the value of k later. Let the half-induced matching be between $A, B \subseteq \mathbb{N}$ with $A = \{a_1, \dots, a_\ell\}$, $B = \{b_1, \dots, b_\ell\}$. Define indexing functions ind_A and ind_B such that $\text{ind}_A(a_i) = \text{ind}_B(b_i) = i$. For $s \in A^k$, define $\text{ind}_A(s) = \sum_{i \in [k]} \text{ind}_A(s[i])$. We partition A^k into sets \mathcal{S}_q with $q \in [\ell \cdot k]$ such that

$$\mathcal{S}_q = \{s \in A^k \mid \text{ind}_A(s) = q\}.$$

Hence, there exists some $q' \in [\ell \cdot k]$ such that

$$|\mathcal{S}_{q'}| \geq \frac{\ell^k}{\ell \cdot k}.$$



■ **Figure 3** Steps in the chain of reductions from $\#Y\text{-ANTIFACTOR}^{\mathcal{R}}$ to $\#(X,Y)\text{-ANTIFACTOR}$, i.e. $\text{Holant}(\text{HW}_{\in \bar{Y}}, \text{HW}_{\in \bar{X}})$. Dotted lines indicate results obtained by combined reductions.

Let $\mathcal{S} = \mathcal{S}_{q'}$ be the set for which we want to lower bound the size of its representative sets. To simplify notation, let $q = q'$. Consider some $s \in \mathcal{S}$. We claim that s is the unique compatible element for $t \in B^k$, where

$$t[i] = b_{\text{ind}_A(s[i])}.$$

It is clear that $s \sim_{\mathcal{C}_X}^k t$. Suppose there is some other $s' \in \mathcal{S}$ such that $s' \sim_{\mathcal{C}_X}^k t$. Then since $\text{ind}_A(s) = \text{ind}_A(s') = q$ and since $s \neq s'$, there is some index j such that $\text{ind}_A(s[j]) > \text{ind}_A(s'[j])$. The j th index of $s' + t$ is $a_{\text{ind}_A(s'[j])} + b_{\text{ind}_A(s[j])}$ because $s'[j] = a_{\text{ind}_A(s'[j])}$. However, observe that this sum must be in X from the fact that there is a half-induced matching between A, B in \mathcal{C}_X and $\text{ind}_A(s[j]) > \text{ind}_A(s'[j])$. This is a contradiction, implying that s is the only compatible partner of t . Thus, s is forced to belong to any representative set $\mathcal{S}' \subseteq_{\mathcal{C}_X\text{-rep}}^k \mathcal{S}$.

Since the above argument holds for all $s \in \mathcal{S}$, we conclude that the only representative set for \mathcal{S} is itself. Now we set k to be large enough such that $k \log(\ell - \epsilon) \leq k \log(\ell) - \log(\ell \cdot k)$. Then we have

$$|\mathcal{S}| \geq \frac{\ell^k}{\ell \cdot k} \geq (\ell - \epsilon)^k. \quad \blacktriangleleft$$

B Proof of Lemma 6.2: Removing Relations

In this section, we prove the reduction from $\#Y\text{-ANTIFACTOR}^{\mathcal{R}}$ to $\#(X,Y)\text{-ANTIFACTOR}$, i.e. Lemma 6.2, by a chain of reductions (cf. Figure 3). We make use of the Holant framework, which was also used in [30], to formally state the results. The first step uses Lemmas 7.5 and 7.6 from [30]. Observe for this that the lemmas work for $\#B\text{-FACTOR}$ even when B is co-finite, that is $\#\bar{B}\text{-ANTIFACTOR}$ because the simple vertices of the instance are not changed in any way.

► **Lemma B.1** (Lemma 7.5 and 7.6 in [30]). *There is a polynomial-time Turing reduction from $\#X\text{-ANTIFACTOR}^{\mathcal{R}}$ to $\text{Holant}(\text{HW}_{\in \bar{X}}, \text{HW}_{=1})$ such that the maximum degree increases to at least 6 and the pathwidth increases by at most a constant depending only on Δ^* , i.e. the maximum total degree of the complex nodes in any bag of the path decomposition.*

► **Note.** Observe that Lemma 7.5 in [30] requires that the relation is even, i.e. the Hamming weight of every accepted input is even. We can easily make every relation even by adding an additional input that is selected whenever the parity of the original input is odd. This

22:22 Anti-Factor is FPT Parameterized by Treewidth and List Size (but Counting is Hard)

additional input is then connected to a EQ_1 node, which can easily be realized by forcing $\max X + 1$ edges to a fresh vertex using $\text{HW}_{=1}^{(1)}$ nodes.

Before proceeding with the next steps, we define, for all $x, y \in \mathbb{Z}$, $w[x, y]$ as a new type of node which has one dangling edge e and the following signature:

$$f(e) = \begin{cases} x & \text{if } e \text{ is not selected} \\ y & \text{if } e \text{ is selected} \end{cases}.$$

Observe that $\text{HW}_{=1}^{(1)}$ is precisely $w[0, 1]$ and $\text{HW}_{\in\{0,1\}}^{(1)}$ corresponds to $w[1, 1]$. In the following constructions we additionally use a $w[-1, 1]$ node. We use the $w[x, y]$ notation in the following wherever possible.

► **Lemma B.2.** *Let $X \subseteq \mathbb{N}$ be a finite set such that $X \not\subseteq \{0\}$. Let R_1, \dots, R_d be d arbitrary relations for some $d \geq 0$. There is a polynomial-time Turing reduction from*

$$\text{Holant}(R_1, \dots, R_d, \text{HW}_{\in\bar{X}}, \text{HW}_{=1}) \text{ to } \text{Holant}(R_1, \dots, R_d, \text{HW}_{\in\bar{X}}, w[1, 1], w[-1, 1], w[0, 1])$$

such that Δ^ increases to $\Delta^* \cdot f(\max X)$ and pw increases to $\text{pw} + \Delta^* \cdot f(\max X)$.*

The proof of the lemma is given in the full version [29] and uses three different gadgets depending on X to realize $\text{HW}_{=1}$. Next we show that we can realize $w[x, y]$ nodes. In particular, we can get the $w[1, 1]$, $w[-1, 1]$, and $w[0, 1]$ nodes introduced by Lemma B.2.

► **Lemma B.3.** *Let $X \subseteq \mathbb{N}$ be a fixed, finite set with $X \not\subseteq \{0\}$. Let R_1, \dots, R_d be d arbitrary relations for some $d \geq 0$. The following holds for arbitrary values x, y . There is a polynomial-time Turing reduction from $\text{Holant}(R_1, \dots, R_d, \text{HW}_{\in\bar{X}}, w[x, y])$ to $\text{Holant}(R_1, \dots, R_d, \text{HW}_{\in\bar{X}})$ such that Δ^* decreases and pw increases to $\text{pw} + \Delta^* \cdot f(\max X)$.*

Proof. We use Lemma 7.11 from [30] as our prototype. However, some arguments from their proof do not follow in our case.

Let U be the set of $w[x, y]$ nodes in the given graph G . Let A_i denotes the number of possible solutions in G where for exactly i of the $w[x, y]$ nodes the dangling edge is not selected and for the other $|U| - i$ nodes the dangling edge is selected. Then we have

$$\text{Holant}(G) = \sum_{i=0}^{|U|} A_i x^i y^{|U|-i}. \quad (1)$$

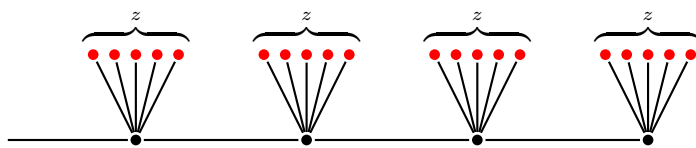
We construct graphs G_d for a new parameter d from G where we replace each $w[x, y]$ node by a gadget H_d with exactly one dangling edge. The construction of H_d is given later as it depends on X . Let $h_0(d)$ denote the number of solutions for H_d when the dangling edge is not selected and $h_1(d)$ when the dangling edge is selected. Then we get

$$\text{Holant}(G_d) = \sum_{i=0}^{|U|} A_i h_0(d)^i h_1(d)^{|U|-i} = h_1(d)^{|U|} \sum_{i=0}^{|U|} A_i \left(\frac{h_0(d)}{h_1(d)} \right)^i.$$

Assume we can find at least $|U| + 1$ values for d such that for all values the ratios $h_0(d)/h_1(d)$ are pairwise different. After computing $\text{Holant}(G_d)$ for these values of d we can recover the value of each A_i . By Equation (1) we can finally output the value of $\text{Holant}(G)$.

It remains to construct the gadgets H_d and to find the values for d . We later construct the gadgets in a way such that there are constants F_1, F_2 , and F_3 only depending on X with

$$\begin{aligned} h_0(d) &:= F_0 \cdot h_0(d-1) + F_1 \cdot h_1(d-1) & h_0(1) &:= F_0 \\ h_1(d) &:= F_1 \cdot h_0(d-1) + F_2 \cdot h_1(d-1) & h_1(1) &:= F_1. \end{aligned}$$



■ **Figure 4** Gadget to realize $w[x, y]$ nodes in Case 1. Red nodes are $\text{HW}_{\in\{0,1\}}^{(1)}$ nodes.

Given these properties of H_d , we can use the following proposition to find sufficiently many values for d . The proof is given in the full version [29].

► **Proposition B.4** (Special Case of Proposition 7.7 in [30]). *Given three constants $F_0, F_1,$ and F_2 with $F_0F_2 \neq (F_1)^2$ and $F_0, F_1 \neq 0$. Let $\{A_n\}_{n \in \mathbb{N}}, \{B_n\}_{n \in \mathbb{N}}$ be two sequences with*

$$\begin{bmatrix} A_n \\ B_n \end{bmatrix} = M \cdot \begin{bmatrix} A_{n-1} \\ B_{n-1} \end{bmatrix} = M^n \cdot U \quad \text{where} \quad M = \begin{bmatrix} F_0 & F_1 \\ F_1 & F_2 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} A_0 \\ B_0 \end{bmatrix} = \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}.$$

Then $\{A_n/B_n\}_{n \in \mathbb{N}}$ is a sequence which does not contain any repetitions.

As a last step we construct the H_d gadgets.

Case 1: $0 \in X$ or $1 \notin X$. We first show how to get a $\text{HW}_{\in\{0,1\}}^{(1)}$ node.

- If $0, 1 \notin X$, then any vertex with a dangling edge acts as a $\text{HW}_{\in\{0,1\}}^{(1)}$ node.
- If $0 \in X, 1 \notin X$, then attach $\max X + 1$ pendant vertices to any vertex v . Then v acts as a $\text{HW}_{\in\{0,1\}}^{(1)}$ node.
- If $0 \in X, 1 \in X$, then take a clique of size $\min(\overline{X}) + 1$. Split the edge between two vertices into two dangling edges. This now acts as a $\text{HW}_{=2}^{(2)}$ node. Attaching $\lceil (\max X + 1)/2 \rceil$ many $\text{HW}_{=2}^{(2)}$ nodes to a new vertex with one dangling edge gives us a $\text{HW}_{\in\{0,1\}}^{(1)}$ node.

H_d consists of a path of d vertices with a dangling edge on the first vertex. For an integer $z \geq \max X + 1$ that we will choose later, attach z pendant $\text{HW}_{\in\{0,1\}}^{(1)}$ nodes to each vertex in the path. See Figure 4 for an illustration. By this definition we get:

$$F_0 = \sum_{i \geq 0: i \in \overline{X}} \binom{z}{i}, \quad F_1 = \sum_{i \geq 0: i+1 \in \overline{X}} \binom{z}{i}, \quad F_2 = \sum_{i \geq 0: i+2 \in \overline{X}} \binom{z}{i}.$$

We claim that there is a z such that assumptions from Proposition B.4 hold. If we can choose z larger than $\max X + 1$, then $F_0, F_1,$ and F_2 are never equal to 0. Now suppose that $F_0F_2 = (F_1)^2$. We will show a contradiction. We first expand the equations above. Then for every z ,

$$\begin{aligned} & \left(\sum_{i \in \overline{X}} \binom{z}{i} \right) \left(\sum_{i+2 \in \overline{X}} \binom{z}{i} \right) = \left(\sum_{i+1 \in \overline{X}} \binom{z}{i} \right)^2 \\ & \left(2^z - \sum_{i \in X} \binom{z}{i} \right) \left(2^z - \sum_{i+2 \in X} \binom{z}{i} \right) = \left(2^z - \sum_{i+1 \in X} \binom{z}{i} \right)^2 \end{aligned}$$

which implies $2^z Q_1(z) = Q_2(z)$, where

$$Q_1(z) = \left(2 \sum_{i+1 \in X} \binom{z}{i} - \sum_{i \in X} \binom{z}{i} - \sum_{i+2 \in X} \binom{z}{i} \right)$$

$$Q_2(z) = \left(\sum_{i+1 \in X} \binom{z}{i} \right)^2 - \left(\sum_{i \in X} \binom{z}{i} \right) \left(\sum_{i+2 \in X} \binom{z}{i} \right).$$

For large enough z , we argue that $Q_1(z)$ is not identically zero. Observe that the second term in $Q_1(z)$ gives a non-zero $z^{\max X}$ monomial whereas the other two terms cannot give a monomial of this degree. Now, since X is a fixed, finite set, Q_1, Q_2 are polynomials with constant degree. Thus, $Q_1(z)$ is zero only for finitely many z . Hence, there are infinitely many (positive) z such that $Q_1(z)$ is non-zero. For each such z we have

$$|2^z Q_1(z)| = |Q_2(z)|.$$

This is immediately a contradiction since $2^z = \omega(z^c)$ for any constant c if z is large enough. Thus, there is some positive integral value of z such that $F_0 F_2 \neq (F_1)^2$. We use this value of z in the construction of the gadget. Note that z only depends on X and can thus be precomputed.

Case 2: $0 \notin X, 1 \in X$. In this case we do not use $\text{HW}_{\in\{0,1\}}^{(1)}$ nodes but EQ_2 nodes, i.e. $\text{HW}_{\in\{0,2\}}^{(2)}$ nodes, instead. We still attach z of these nodes by $2z$ edges to the vertices on the path. Then the proof follows similarly to the previous case. The formal proof is given in the full version [29]. ◀

► **Lemma B.5.** *Let $X \subseteq \mathbb{N}$ be a fixed, finite set with $X \not\subseteq \{0\}$. Let R_1, \dots, R_d be d arbitrary relations for some $d \geq 0$. There is a polynomial-time Turing reduction from*

$$\text{Holant}(R_1, \dots, R_d, \text{HW}_{\in\bar{X}}, w[1, 1], w[-1, 1], w[0, 1]) \text{ to } \text{Holant}(R_1, \dots, R_d, \text{HW}_{\in\bar{X}})$$

such that Δ^* decreases and pw increases to $\text{pw} + \Delta^* \cdot f(\max X)$.

Proof. We first use Lemma B.3 to remove the $w[1, 1]$ nodes. Observe that this can alternatively be done by a simple construction using a fresh vertex with $\max X + 1$ forced edges. Then we apply the lemma two more times to remove the $w[-1, 1]$ nodes and finally the $w[0, 1]$ nodes. ◀

Now we can prove the reduction from $\#Y\text{-ANTIFACTOR}^{\mathcal{R}}$ to $\#(X, Y)\text{-ANTIFACTOR}$.

Proof of Lemma 6.2. By the reduction of Lemma B.1 we can reduce $\#Y\text{-ANTIFACTOR}^{\mathcal{R}}$ to $\text{Holant}(\text{HW}_{\in\bar{Y}}, \text{HW}_{=1})$. This can trivially be reduced to $\text{Holant}(\text{HW}_{\in\bar{Y}}, \text{HW}_{\in\bar{X}}, \text{HW}_{=1})$ as we do not have any vertices with relation $\text{HW}_{\in\bar{X}}$. Then we invoke Lemmas B.2 and B.5 such that the vertices with relation $\text{HW}_{\in\bar{Y}}$ are not changed (or used for any construction). By this we end the reduction with $\text{Holant}(\text{HW}_{\in\bar{Y}}, \text{HW}_{\in\bar{X}})$ which precisely corresponds to $\#(X, Y)\text{-ANTIFACTOR}$. ◀