



Sampling Unlabeled Chordal Graphs in Expected Polynomial Time

Úrsula Hébert-Johnson  

University of California, Santa Barbara, CA, USA

Daniel Lokshtanov  

University of California, Santa Barbara, CA, USA

Abstract

We design an algorithm that generates an n -vertex unlabeled chordal graph uniformly at random in expected polynomial time. Along the way, we develop the following two results: (1) an FPT algorithm for counting and sampling labeled chordal graphs with a given automorphism π , parameterized by the number of moved points of π , and (2) a proof that the probability that a random n -vertex labeled chordal graph has a given automorphism $\pi \in S_n$ is at most $1/2^{c \max\{\mu^2, n\}}$, where μ is the number of moved points of π and c is a constant. Our algorithm for sampling unlabeled chordal graphs calls the aforementioned FPT algorithm as a black box with potentially large values of the parameter μ , but the probability of calling this algorithm with a large value of μ is exponentially small.

2012 ACM Subject Classification Theory of computation \rightarrow Generating random combinatorial structures; Theory of computation \rightarrow Graph algorithms analysis

Keywords and phrases Chordal graphs, graph sampling, graph counting, unlabeled graphs

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.46

Related Version *Full Version:* <https://arxiv.org/abs/2501.05024>

Funding Úrsula Hébert-Johnson: Supported by NSF grant CCF-2147094.

Acknowledgements We would like to thank Eric Vigoda for discussions that led to a concise proof of an important step in the running time analysis.

1 Introduction

A graph is *chordal* if it has no induced cycles of length at least 4. The term was coined by Gavril in 1972 [12], more than fifty years ago, but the notion of chordal graphs in fact goes as far back as 1958 [13]. In the early papers, chordal graphs were referred to by other names, such as *triangulated* graphs. By now, many structural results have been proved about chordal graphs, and there are many algorithms that take a chordal graph as input. Thus it would be useful to have an efficient algorithm for generating random chordal graphs, both for the practical purpose of software testing, as well as the more mathematical purpose of testing conjectures.

In [14], Hébert-Johnson et al. designed an algorithm that generates n -vertex labeled chordal graphs uniformly at random. This algorithm runs in polynomial time, using at most $O(n^7)$ arithmetic operations for the first sample and $O(n^4)$ arithmetic operations for each subsequent sample. However, when discussing the performance of an algorithm that is being tested, the correct output typically does not depend on the labeling of the vertices. If we use a labeled-graph sampling algorithm to generate random test cases, then asymmetric graphs will be given too much weight/probability compared to those that happen to have many automorphisms. This naturally leads to the question of efficiently generating *unlabeled* chordal graphs uniformly at random. In this paper, we present an algorithm that solves this problem and runs in expected polynomial time.



© Úrsula Hébert-Johnson and Daniel Lokshtanov;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 46; pp. 46:1–46:20



Leibniz International Proceedings in Informatics

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



► **Theorem 1.** *There is a randomized algorithm that given $n \in \mathbb{N}$, generates a graph uniformly at random from the set of all unlabeled chordal graphs on n vertices. This algorithm uses $O(n^7)$ arithmetic operations in expectation.*

It is worth mentioning the difference between the running times for labeled vs. unlabeled chordal graph sampling. The sampling algorithm of Hébert-Johnson et al. generates a random n -vertex labeled chordal graph in polynomial time, even in the worst case. However, obtaining a worst-case polynomial-time algorithm for sampling *unlabeled* chordal graphs – or an expected-polynomial-time counting algorithm for such graphs – appears to be very difficult since these questions remain open even for general graphs. This is relevant because the class of chordal graphs is known to be GI-complete. While there exist classes of graphs (which we discuss below) for which efficient unlabeled sampling algorithms are known, we are not aware of any GI-complete graph class with such an algorithm.

Our algorithm for sampling unlabeled chordal graphs builds upon an algorithm of Wormald that generates n -vertex unlabeled graphs uniformly at random in expected time $O(n^2)$ [20]. This in turn builds upon a related algorithm by Dixon and Wilf [7] that solves the same problem but assumes that the exact number of n -vertex unlabeled graphs has already been computed. In [20], the algorithm of Wormald follows a somewhat similar structure but removes that assumption. As mentioned above, the question of computing the exact number of n -vertex unlabeled graphs in expected polynomial time remains open to this day.

It often happens that we wish to sample from a particular graph class (e.g., chordal graphs). For unlabeled trees, there is a uniform sampling algorithm that runs in polynomial time [19]. One can also count the exact number of unlabeled trees on n vertices in polynomial time [17, A000055]. On the topic of counting, an algorithm for counting unlabeled k -trees is presented in [9], but the running time is not stated. An expected-polynomial-time algorithm for uniform sampling of 2-connected unlabeled planar graphs was presented by Bodirsky et al. in 2005 [4], followed by the same result for connected unlabeled cubic planar graphs in 2008 [6]. For the class of connected unlabeled bipartite permutation graphs, a uniform sampling algorithm was designed by Saitoh et al. that runs in $O(n)$ time [18].

Although extensive research has been done on the topic of labeled graph sampling [5, 8, 10, 11], to the best of our knowledge, the literature on sampling *unlabeled* graphs from a given graph class is relatively sparse. As is the case for chordal graphs, the corresponding labeled sampling problem tends to be solved first for a given graph class, and then perhaps one can address the problem of efficiently sampling unlabeled graphs from the same graph class.

1.1 Methods

The sampling algorithm of Wormald [20] is based on the fact that unlabeled graphs correspond to orbits of the following group action: the symmetric group S_n acts on the set of labeled graphs by permuting the vertex labels. This correspondence follows from the Frobenius-Burnside lemma. Therefore, to sample a random unlabeled graph, it is enough to sample a random orbit of this group action.

To make this approach work for chordal graphs, we need two ingredients: (1) an algorithm for counting and sampling labeled chordal graphs with a given automorphism π , and (2) a proof that the probability that a random n -vertex labeled chordal graph has a given automorphism $\pi \in S_n$ is at most $1/2^{c \max\{\mu^2, n\}}$, where μ is the number of moved points of π and c is a constant.

For (1), we design an FPT (fixed-parameter tractable) counting algorithm that is parameterized by μ , the number of moved points of π . This algorithm uses $O(2^{7\mu}n^9)$ arithmetic operations. Using the standard sampling-to-counting reduction of [15], we also obtain a corresponding sampling algorithm with the same running time. Our main algorithm (for sampling *unlabeled* chordal graphs) calls each of these FPT algorithms as a black box with potentially large values of the parameter μ . Nevertheless, using the bound from (2), we are able to show that the probability of using a large value of μ is exponentially small, so the expected running time is not significantly affected.

To design the counting algorithm for (1), we rewrite each of the recurrences from the algorithm of Hébert-Johnson et al., now carrying around information about the automorphism π and its moved points. The original algorithm is a dynamic-programming algorithm in which there is a constant number of types of vertices (X , L , etc.) in each graph that we wish to count. In our updated version, we now keep track of the type of each vertex that is moved by π .

To prove the bound for (2), we distinguish between the cases when μ is small and μ is large (μ is either less than or greater than $\frac{n}{d \log n}$, where d is a constant). When μ is small, we use the fact that almost every chordal graph is a split graph [1], and we strengthen this by showing that in fact, almost every chordal graph is a balanced split graph. For the case of balanced split graphs, the argument is easy and is similar to the proof of the bound for general graphs [16]. When μ is large, the argument is more complicated. We observe that the vast majority of n -vertex labeled chordal graphs have maximum clique size close to $n/2$. Along the way, we also use the fact that for every chordal graph G , there exists a PEO (perfect elimination ordering) of G such that some maximum clique appears at the tail end of that PEO.

2 Preliminaries

Let \mathbb{N} be the set of natural numbers, not including 0. For $n \in \mathbb{N}$, we use the notation $[n] := \{1, 2, \dots, n\}$. For a graph G and vertex subsets $S, T \subseteq V(G)$, we say S *sees all of* T if $T \subseteq N(S)$.

► **Definition 2.** Let $A = \{a_1, \dots, a_r\}$ and $B = \{b_1, \dots, b_r\}$ be finite subsets of \mathbb{N} such that $|A| = |B|$, where the elements a_i and b_i are listed in increasing order. We define $\phi(A, B): A \rightarrow B$ as the bijection that maps a_i to b_i for all $i \in [r]$.

2.1 Permutations and labeled graphs

For $n \in \mathbb{N}$, let S_n denote the group of all permutations of $[n]$. For a permutation $\pi \in S_n$, we define $M_\pi := \{i \in [n] : \pi(i) \neq i\}$ to be the set of points moved by π . For $n \in \mathbb{N}$, $[n]_0 := \{0, 2, 3, \dots, n\}$ denotes the set of all possible values of $|M_\pi|$ for $\pi \in S_n$.

Suppose $\pi \in S_n$, $C \subseteq [n]$. We write $\pi(C) := \{\pi(i) : i \in C\}$ to denote the image of C under π . We say C is *invariant* under π if $\pi(i) \in C$ for all $i \in C$. For a set C that is invariant under π , we write $\pi|_C$ to denote the permutation π restricted to the domain C .

A *labeled graph* is a pair $G = (V, E)$, where the vertex set V is a finite subset of \mathbb{N} and the edge set E is a set of two-element subsets of V . For a permutation $\pi \in S_n$ and a labeled graph G such that $V(G) \subseteq [n]$ is invariant under π , we say $\pi|_{V(G)}$ is an *automorphism* of G if for all $u, v \in V(G)$, u and v are adjacent if and only if $\pi(u)$ and $\pi(v)$ are adjacent.

2.2 Chordal graphs and related notions

A vertex v in a graph G is *simplicial* if its neighborhood $N(v)$ is a clique. A *perfect elimination ordering* (PEO) of a graph G is an ordering v_1, \dots, v_n of the vertices of G such that for all $i \in [n]$, v_i is simplicial in the subgraph induced by the vertices v_i, \dots, v_n . A graph is chordal if and only if it has a perfect elimination ordering [3]. The following two lemmas are well-known facts about chordal graphs. The proof of the first can be found in [3].

► **Lemma 3.** *Every chordal graph G contains a simplicial vertex. If G is not a complete graph, then G contains two non-adjacent simplicial vertices.*

► **Lemma 4.** *A chordal graph G on n vertices has at most n maximal cliques.*

Proof. Let C be a maximal clique in G , and let v_i be the leftmost vertex of C in a given perfect elimination ordering v_1, \dots, v_n of G . We claim that C is equal to the closed neighborhood to the right of v_i , i.e., $C = N[v_i] \cap \{v_i, \dots, v_n\}$. It is clear that C is contained in $N[v_i] \cap \{v_i, \dots, v_n\}$ since v_i has no other neighbors to its right, so we indeed have $C = N[v_i] \cap \{v_i, \dots, v_n\}$ by maximality of C . Therefore, there are at most n maximal cliques. ◀

► **Definition 5.** *Let G_1, G_2 be two graphs, and suppose $C := V(G_1) \cap V(G_2)$ is a clique in both G_1 and G_2 . When we say we **glue** G_1 and G_2 together at C to obtain G , this means G is the union of G_1 and G_2 : the vertex set is $V(G) = V(G_1) \cup V(G_2)$, and the edge set is $E(G) = E(G_1) \cup E(G_2)$.*

As is shown in [14], if G_1 and G_2 are both chordal, then the resulting graph G is chordal.

2.3 Evaporation sequences

Our algorithm for counting the number of labeled chordal graphs with a given automorphism will use the notion of evaporation sequences from [14].

Suppose we are given a chordal graph G and a clique $X \subseteq V(G)$. The *evaporation sequence* of G with *exception set* X is defined as follows: If $X = V(G)$, then the evaporation sequence of G is the empty sequence. If $X \subsetneq V(G)$, then let \tilde{L}_1 be the set of all simplicial vertices in G , and let $L_1 = \tilde{L}_1 \setminus X$. Suppose L_2, \dots, L_t is the evaporation sequence of $G \setminus L_1$ (with exception set X). Then L_1, L_2, \dots, L_t is the evaporation sequence of G . As is shown in [14], the fact that X is a clique implies that all vertices outside of X eventually evaporate, so this is well-defined.

If the evaporation sequence L_1, L_2, \dots, L_t of G has length t , then we say G *evaporates* at time t with *exception set* X , and t is called the *evaporation time*. We define $L_G(X) := L_t$ to be the last set in the evaporation sequence of G , and we let $L_G(X) = \emptyset$ if the sequence is empty. Similarly, we define the evaporation time of a vertex subset. Suppose G has evaporation sequence L_1, L_2, \dots, L_t with exception set X , and suppose $S \subseteq V(G) \setminus X$ is a nonempty vertex subset. Let t_S be the largest index i such that $L_i \cap S \neq \emptyset$. We say S *evaporates* at time t_S in G with exception set X .

3 Sampling unlabeled chordal graphs

In [20], Wormald presented an algorithm that generates an n -vertex unlabeled graph uniformly at random in expected time $O(n^2)$. In this paper, we achieve a similar result for chordal graphs:

► **Theorem 1.** *There is a randomized algorithm that given $n \in \mathbb{N}$, generates a graph uniformly at random from the set of all unlabeled chordal graphs on n vertices. This algorithm uses $O(n^7)$ arithmetic operations in expectation.*

The sampling algorithm of Wormald makes use of the fact that unlabeled graphs correspond to orbits of a particular group action. Since our algorithm will follow a similar outline, we begin by discussing some of the ideas behind the algorithm of Wormald.

Suppose we have been given $n \in \mathbb{N}$ as input, and let Ω be the set of all labeled graphs with vertex set $[n]$. The symmetric group S_n acts on Ω in the following way: For each $\pi \in S_n$ and $G \in \Omega$, $\pi \cdot G$ is the graph that results from permuting the vertex labels of G according to π . The orbits of Ω under the action of S_n are the isomorphism classes of labeled graphs, each of which corresponds to an unlabeled graph. Let

$$\Gamma = \{(\pi, G) \in S_n \times \Omega : \pi \text{ is an automorphism of } G\}.$$

Suppose we fix an n -vertex unlabeled graph H , and let the corresponding isomorphism class of labeled graphs be \mathcal{H} . As is shown in [20], the number of pairs $(\pi, G) \in \Gamma$ such that $G \in \mathcal{H}$ is equal to $|S_n| = n!$. This follows from the Frobenius-Burnside lemma. Therefore, if we sample a random pair $(\pi, G) \in \Gamma$ uniformly at random, and then we forget the labels on the graph G , this amounts to sampling an n -vertex unlabeled graph uniformly at random.

In the case of chordal graphs, the same statements hold true. Let Ω_{chord} be the set of all labeled chordal graphs with vertex set $[n]$. The symmetric group S_n acts on Ω_{chord} in the same way as above, by permuting the vertex labels. The set of orbits of this group action corresponds to the set of unlabeled chordal graphs. Let

$$\Gamma_{\text{chord}} = \{(\pi, G) \in S_n \times \Omega_{\text{chord}} : \pi \text{ is an automorphism of } G\}.$$

Let H_c be an unlabeled chordal graph, and let \mathcal{H}_c be the corresponding isomorphism class of labeled graphs. Applying the Frobenius-Burnside lemma to the orbit corresponding to \mathcal{H}_c shows that the number of pairs $(\pi, G) \in \Gamma_{\text{chord}}$ such that $G \in \mathcal{H}_c$ is equal to $n!$.

In [20], Wormald describes an algorithm for sampling a random pair $(\pi, G) \in \Gamma$ in order to sample a random unlabeled graph. The same outline can be used to sample a random pair $(\pi, G) \in \Gamma_{\text{chord}}$. However, there are two key points where some difficulty arises. First of all, in one of the steps of the algorithm that samples from Γ , it is necessary to count the number of n -vertex labeled graphs with a given automorphism (and sample from the set of such graphs). This is easy to do for general graphs but becomes more complicated for chordal graphs (see Section 4). Second, the algorithm of Wormald uses the fact that the number of labeled graphs with a given automorphism π is at most $2^{\binom{n}{2} - \mu n/2 + \mu(\mu+2)/4}$, where $\mu = |M_\pi|$ is the number of moved points of π . To transform this into an algorithm for sampling unlabeled chordal graphs, it is necessary to prove similar bounds on the number of labeled chordal graphs with a given automorphism. These will be the bounds B_μ in our algorithm.

3.1 Algorithm for sampling unlabeled chordal graphs

Let $\text{COUNT_CHORDAL_LAB}(n)$ stand for the counting algorithm in [14] that computes the number of n -vertex labeled chordal graphs. In the full version of the paper, we prove the following two theorems.

► **Theorem 6.** *There is a deterministic algorithm that given $n \in \mathbb{N}$ and $\pi \in S_n$, computes the number of labeled chordal graphs with vertex set $[n]$ for which π is an automorphism. This algorithm uses $O(2^{7\mu} n^9)$ arithmetic operations, where $\mu = |M_\pi|$.*

► **Theorem 7.** *There is a randomized algorithm that given $n \in \mathbb{N}$ and $\pi \in S_n$, generates a graph uniformly at random from the set of all labeled chordal graphs with vertex set $[n]$ for which π is an automorphism. This algorithm uses $O(2^{7\mu}n^9)$ arithmetic operations, where $\mu = |M_\pi|$.*

See Section 4 for the details of the counting algorithm of Theorem 6 along with some intuition behind the proof of correctness. In our algorithm for sampling unlabeled chordal graphs, `COUNT_CHORDAL_LAB`(n, π) stands for the algorithm of Theorem 6 and `SAMPLE_CHORDAL_LAB`(n, π) stands for the algorithm of Theorem 7.

Recall that $[n]_0 = \{0, 2, 3, \dots, n\}$. For $\mu \in [n]_0$, let $R_\mu := \{\pi \in S_n : |M_\pi| = \mu\}$ be the set of permutations with exactly μ moved points. For $\pi \in S_n$, let $\text{Fix}(\pi)$ denote the set of n -vertex labeled chordal graphs G such that π is an automorphism of G . To follow the same approach as the algorithm of Wormald, for each $\mu \in [n]_0$, we need an upper bound B_μ that satisfies $B_\mu \geq |R_\mu| |\text{Fix}(\pi)|$ for all $\pi \in R_\mu$. Let B_0 be the number of n -vertex labeled chordal graphs, which is exactly equal to $|R_0| |\text{Fix}(\text{id})|$. For $2 \leq \mu \leq \frac{n}{200 \log n}$, let

$$B_\mu = \left(B_0 (9/10)^n + 2^n 2^{2n^2/9} + n \cdot 2^{n^2/4+n/2} \right) n^\mu \mu!, \quad (1)$$

and for $\frac{n}{200 \log n} < \mu \leq n$, let

$$B_\mu = n^{2n+1} 2^{n^2/4-f(\mu)} n^\mu \mu!, \quad (2)$$

where $f(\mu) = \frac{\mu^2}{900} - \frac{\mu}{10}$. In Section 3.2, we will prove that we indeed have $B_\mu \geq |R_\mu| |\text{Fix}(\pi)|$ for all $\pi \in R_\mu$, when n is sufficiently large. Let $B = \sum_{\mu \in [n]_0} B_\mu$.

Our algorithm for sampling a random n -vertex unlabeled chordal graph is given in Algorithm 1. The general idea is as follows. For $\mu \in [n]_0$, let $\Gamma_\mu = \{(\pi, G) \in \Gamma_{\text{chord}} : |M_\pi| = \mu\}$. We choose μ such that the probability of each value of μ is B_μ/B , which is approximately equal to $\Gamma_\mu/\Gamma_{\text{chord}}$. (We do not know how to efficiently compute the exact value of $\Gamma_\mu/\Gamma_{\text{chord}}$ since we do not know the exact number of unlabeled chordal graphs.) Since B_μ/B is not exactly equal to $\Gamma_\mu/\Gamma_{\text{chord}}$, we adjust for this by restarting with a certain probability in Step 11. We then proceed to select a random pair $(\pi, G) \in \Gamma_\mu$, and we output the graph G without labels.

■ **Algorithm 1** Unlabeled chordal graph sampler.

```

1: procedure SAMPLE_CHORDAL_UNLABELED( $n$ )
2:   ▷ Setup
3:    $B_0 \leftarrow \text{COUNT\_CHORDAL\_LAB}(n)$ 
4:   Let  $B_\mu$  be given by Equation (1) for  $2 \leq \mu \leq \frac{n}{200 \log n}$ 
5:   Let  $B_\mu$  be given by Equation (2) for  $\frac{n}{200 \log n} < \mu \leq n$ 
6:    $B \leftarrow \sum_{\mu \in [n]_0} B_\mu$ 
7:
8:   ▷ Main algorithm
9:   Choose  $\mu \in [n]_0$  at random such that  $\mu = i$  with probability  $B_i/B$  for each  $i \in [n]_0$ 
10:  Choose  $\pi \in R_\mu$  uniformly at random
11:  Restart (go back to Step 9) with probability
       $1 - |R_\mu| \cdot \text{COUNT\_CHORDAL\_LAB}(n, \pi) / B_\mu$ 
12:   $G \leftarrow \text{SAMPLE\_CHORDAL\_LAB}(n, \pi)$ 
13:  Forget the labels on the vertices of  $G$ 
14:  return  $G$ 
15: end procedure

```

Step 10 can easily be implemented in $O(n)$ time in the following way. If $\mu = 0$, let $\pi = \text{id}$. For $\mu \geq 2$, we can repeatedly choose a random permutation of $[\mu]$ until we obtain a derangement. The expected number of trials for this is a constant (see [20]).

In Step 11, to compute $|R_\mu|$, we observe that $|R_0| = 1$ and $|R_\mu| = !\mu \binom{n}{\mu}$ for $\mu \geq 2$. Here $!\mu$ is the number of derangements of μ . We compute $!\mu$ using the formula $!m = m! \sum_{i=0}^m (-1)^i / i!$ for $m \in \mathbb{N}$, which can be derived using the inclusion-exclusion principle.

3.2 Correctness of Algorithm 1

The correctness of `COUNT_CHORDAL_LAB`(n, π) and `SAMPLE_CHORDAL_LAB`(n, π) is proved in the full version of the paper.

We need to show that the output graph is chosen uniformly at random. One “iteration” refers to one run of Steps 9 to 11 or 9 to 14. In a given iteration, we say the pair (π, G) was “chosen” if π was chosen from R_μ and G was chosen by `SAMPLE_CHORDAL_LAB`(n, π). We claim that for all $(\pi, G) \in \Gamma_{\text{chord}}$, in any given iteration, the probability that (π, G) is chosen is $1/B$. Indeed, this probability is equal to

$$\frac{B_\mu}{B} \frac{1}{|R_\mu|} \frac{|R_\mu| |\text{Fix}(\pi)|}{B_\mu} \frac{1}{|\text{Fix}(\pi)|} = \frac{1}{B},$$

where $\mu = |M_\pi|$, since the probability of choosing G in Step 12 is $1/|\text{Fix}(\pi)|$. Therefore, we output all n -vertex unlabeled chordal graphs with equal probability.

Next, we need to show that $B_\mu \geq |R_\mu| |\text{Fix}(\pi)|$ for all $\pi \in R_\mu$ to verify that the probability $|R_\mu| |\text{Fix}(\pi)| / B_\mu$ in Step 11 is at most 1. When $\mu = 0$ this is an equality, so suppose $\mu \geq 2$. Clearly $|R_\mu| \leq n^\mu \mu!$, so we just need to prove that the number of n -vertex labeled chordal graphs with automorphism π is at most $B_\mu / (n^\mu \mu!)$ for all $\pi \in S_n$ with μ moved points.

In the case when B_μ is defined according to Equation (2), this follows from Theorem 8. The proof this theorem can be found in the full version of the paper. (The bound in Theorem 8 is in fact true for all values of μ – the reason why we define B_μ differently for smaller values of μ will become clear when we discuss the running time in Section 3.3.)

► **Theorem 8.** *Let $n \in \mathbb{N}$, $\pi \in S_n$, and let $\mu = |M_\pi|$. The number of labeled chordal graphs with vertex set $[n]$ for which π is an automorphism is at most*

$$n^{2n+1} 2^{n^2/4 - f(\mu)},$$

where $f(\mu) = \frac{\mu^2}{900} - \frac{\mu}{10}$.

For the other case, suppose $2 \leq \mu \leq \frac{n}{200 \log n}$, and suppose $\pi \in S_n$ is a permutation with μ moved points. We need to show that the number of n -vertex labeled chordal graphs with automorphism π is at most $B_\mu / (n^\mu \mu!)$. We begin by reducing to the case of split graphs. A *split* graph is a graph whose vertex set can be partitioned into a clique and an independent set, with arbitrary edges between the two parts. It is easy to see that every split graph is chordal. Furthermore, the following result by Bender et al. [1] shows that a random n -vertex labeled chordal graph is a split graph with probability $1 - o(1)$.

► **Proposition 9** (Bender et al. [1]). *If $\alpha > \sqrt{3}/2$, n is sufficiently large, and G is a random n -vertex labeled chordal graph, then*

$$\Pr(G \text{ is a split graph}) > 1 - \alpha^n.$$

Applying this proposition with $\alpha = \frac{9}{10}$ tells us that the number of n -vertex labeled chordal graphs that are *not split* is at most $B_0 \left(\frac{9}{10}\right)^n$. To bound the number of n -vertex labeled split graphs with automorphism π , we consider two cases: balanced split graphs and unbalanced split graphs. We say a partition of the vertex set of a split graph G is a *split partition* if it partitions G into a clique and an independent set; i.e., a split partition is a partition that demonstrates that G is split. We denote a split partition that consists of the clique C and the independent set I by the ordered pair (C, I) . We say an n -vertex split graph is *balanced* if $|C| \geq \frac{n}{3}$ and $|I| \geq \frac{n}{3}$ for every split partition (C, I) of G . It is easy to bound the number of unbalanced split graph as follows.

► **Lemma 10.** *The number of labeled split graphs on n vertices that are not balanced is at most $2^n 2^{2n^2/9}$.*

Proof. Suppose (C, I) is a partition of $[n]$ into two parts such that $|C| < \frac{n}{3}$ or $|I| < \frac{n}{3}$. The number of labeled split graphs with this particular split partition is at most $2^{|I||C|} \leq 2^{\frac{n}{3} \cdot \frac{2n}{3}} = 2^{2n^2/9}$. Therefore, the number of unbalanced labeled split graphs on n vertices is at most $2^n 2^{2n^2/9}$ since there are at most 2^n possible partitions. ◀

The following lemma will be useful for bounding the number of balanced split graphs.

► **Lemma 11.** *Let $\pi \in S_n$, and let G be an n -vertex labeled split graph. If π is an automorphism of G , then there exists a split partition (C, I) of G such that C and I are invariant under π .*

Proof. Let \hat{C} be the set of vertices that belong to the clique in every split partition of G , let \hat{I} be the set of vertices that belong to the independent set in every split partition of G , and let $\hat{Q} = V(G) \setminus (\hat{C} \cup \hat{I})$. By Observation 7.3 in [14], every vertex in \hat{Q} is adjacent to every vertex in \hat{C} and is non-adjacent to every vertex in \hat{I} . By Lemma 7.4 in [14], \hat{Q} is either a clique or an independent set. Suppose π is an automorphism of G . If \hat{Q} is a clique, then the split partition $(\hat{C} \cup \hat{Q}, \hat{I})$ has the desired property. If \hat{Q} is an independent set, then the split partition $(\hat{C}, \hat{I} \cup \hat{Q})$ has the desired property. ◀

► **Lemma 12.** *Let $\pi \in S_n$, and suppose $|M_\pi| \geq 2$. The number of balanced labeled split graphs G on n vertices such that π is an automorphism of G is at most*

$$n \cdot 2^{n^2/4+n/2}.$$

Proof. Suppose (C, I) is a partition of $[n]$ into two parts. Let $i = |I|$ and $c = |C|$. The number of labeled split graphs with this particular split partition is 2^{ic} . Therefore, the number of labeled split graphs with automorphism π for which (C, I) has the property from Lemma 11 is at most 2^{ic} . Let $Z_{(C, I)}$ be the number of such graphs. Whenever two vertices $u, v \in I$ (resp. C) belong to the same cycle in the cycle decomposition of π , u and v must have the same relationship as each other to each of the vertices in C (resp. I). Therefore, we in fact have an upper bound of $\max\{2^{(i-1)c}, 2^{i(c-1)}\}$ on $Z_{(C, I)}$, since π has at least two moved points. If $i \geq \frac{n}{3}$ and $c \geq \frac{n}{3}$, then we have $\max\{2^{(i-1)c}, 2^{i(c-1)}\} \leq 2^{n^2/4-n/2}$.

By Lemma 11, every balanced labeled split graph on n vertices with automorphism π has a split partition (C, I) such that C and I are invariant under π . Furthermore, this split partition is balanced (i.e., both parts have size at least $\frac{n}{3}$). Therefore, the number of balanced labeled split graphs on n vertices with automorphism π is at most

$$\sum_{\lceil \frac{n}{3} \rceil \leq i \leq \lfloor \frac{2n}{3} \rfloor} 2^n \cdot 2^{n^2/4-n/2} \leq n \cdot 2^{n^2/4+n/2}$$

since the number of partitions (C, I) with $|I| = i$ is certainly at most 2^n . ◀

Putting together Proposition 9 and Lemmas 10 and 12, we can see that

$$\left(B_0(9/10)^n + 2^n 2^{2n^2/9} + n \cdot 2^{n^2/4+n/2} \right) n^\mu \mu!$$

is an upper bound on $|R_\mu| |\text{Fix}(\pi)|$ when n is sufficiently large.

Let N_0 be the cutoff such that this works for $n \geq N_0$. To be precise, when implementing this algorithm, we would solve the problem by brute force if $n < N_0$ (by generating all possible n -vertex chordal graphs and then selecting one at random), and we would run Algorithm 1 as written if $n \geq N_0$.

3.3 Running time of Algorithm 1

The running time of Steps 1-6 is $O(n^7)$ arithmetic operations since that is the running time of `COUNT_CHORDAL_LAB`(n). In this section, we will show that the rest of the algorithm uses only $O(n^7)$ arithmetic operations in expectation.

If we choose $\mu = 0$ in Step 9 of Algorithm 1, then the algorithm is guaranteed to terminate in that iteration. Thus the expected number of iterations is at most B/B_0 . Let T be the expected running time of Algorithm 1 after completing Step 6 (this is where we choose μ at random and the loop begins). In Lemma 13, we show that $\mathbf{E}[T]$ is at most the product of B/B_0 and the expected running time of one iteration.

Let N be the number of iterations of Algorithm 1, and let T_j be the time spent in iteration j for each $j \in [N]$. We have $T = \sum_{j=1}^N T_j$.

► **Lemma 13.** *We have $\mathbf{E}[T] \leq \frac{B}{B_0} \mathbf{E}[T_1]$.*

Proof. Clearly $\mathbf{E}[T]$ is finite since the expected number of iterations is finite and the procedures `COUNT_CHORDAL_LAB`(n, π) and `SAMPLE_CHORDAL_LAB`(n, π) have worst-case running time bounds. Therefore, we can solve for $\mathbf{E}[T]$ in the following way.

The steps that we run in one iteration (Steps 9-14) do not depend on j – they are always the same, regardless of how many iterations have happened so far. Thus we have $\mathbf{E}\left[\sum_{j=2}^N T_j \mid N > 1\right] = \mathbf{E}[T]$, which implies $\mathbf{E}[T \mid N > 1] = \mathbf{E}[T_1 \mid N > 1] + \mathbf{E}[T]$. Therefore, we have

$$\begin{aligned} \mathbf{E}[T] &= \mathbf{E}[T \mid N = 1] \Pr(N = 1) + \mathbf{E}[T \mid N > 1] \Pr(N > 1) \\ &= \mathbf{E}[T_1 \mid N = 1] \Pr(N = 1) + (\mathbf{E}[T_1 \mid N > 1] + \mathbf{E}[T]) \cdot \Pr(N > 1) \end{aligned}$$

$$\begin{aligned} \implies \mathbf{E}[T](1 - \Pr(N > 1)) &= \mathbf{E}[T_1 \mid N = 1] \Pr(N = 1) + \mathbf{E}[T_1 \mid N > 1] \Pr(N > 1) \\ &= \mathbf{E}[T_1] \end{aligned}$$

$$\implies \mathbf{E}[T] = \frac{\mathbf{E}[T_1]}{\Pr(N = 1)} \leq \frac{B}{B_0} \mathbf{E}[T_1]. \quad \blacktriangleleft$$

For $\mu \in [n]_0$, let $T(n, \mu)$ be an upper bound on the time it takes to run one iteration, assuming we have chosen this particular value of μ in Step 9. By the running time of `COUNT_CHORDAL_LAB`(n, π) and `SAMPLE_CHORDAL_LAB`(n, π), we can assume $T(n, \mu) = O(2^{7\mu} n^9)$. We have

$$\mathbf{E}[T_1] = \sum_{\mu \in [n]_0} \frac{B_\mu}{B} T(n, \mu).$$

46:10 Sampling Unlabeled Chordal Graphs in Expected Polynomial Time

To prove a bound on $\mathbf{E}[T_1]$, we will start by proving a bound on B_μ/B for all $\mu \in [n]_0$. Since $B_0 \leq B$, it is sufficient to prove a bound on B_μ/B_0 for all $\mu \in [n]_0$. The following lemma gives us a lower bound on B_0 .

► **Lemma 14.** *For $n \geq 2$, the number of n -vertex labeled chordal graphs is at least*

$$\frac{2^n 2^{n^2/4}}{n^2}.$$

Proof. It is enough to just consider n -vertex labeled split graphs with a split partition (C, I) such that $|C| = \lfloor \frac{n}{2} \rfloor$. Since $\binom{n}{\lfloor \frac{n}{2} \rfloor} \geq 2^n/n$ for $n \geq 2$, the number of such graphs is at least

$$\binom{n}{\lfloor \frac{n}{2} \rfloor} \frac{2^{n^2/4}}{n} \geq \frac{2^n 2^{n^2/4}}{n^2}.$$

We divide by n in the first expression since each split graph can have up to n distinct split partitions in which C is of a given size [2]. ◀

► **Lemma 15.** *Suppose $n \geq 13$. If $2 \leq \mu \leq \frac{n}{200 \log n}$, then*

$$\frac{B_\mu}{B_0} \leq \frac{3}{n^{16\mu}}.$$

Proof. Let $B_\mu^{(1)}$, $B_\mu^{(2)}$, and $B_\mu^{(3)}$ be the three terms that are added together in Equation (1), in order, so that $B_\mu = (B_\mu^{(1)} + B_\mu^{(2)} + B_\mu^{(3)}) n^\mu \mu!$. We have

$$\frac{B_\mu^{(1)} n^\mu \mu!}{B_0} = \left(\frac{9}{10}\right)^n n^\mu \mu!,$$

and we claim that this is at most $1/n^{16\mu}$. Since $\mu! \leq n^\mu$, it is sufficient to show $n^{18\mu} \leq (10/9)^n$, which is true when $\mu \leq \frac{n}{200 \log n}$.

For the next term, by Lemma 14 we have

$$\frac{B_\mu^{(2)} n^\mu \mu!}{B_0} \leq n^2 2^{-n^2/36} n^\mu \mu!.$$

To see that this is at most $1/n^{16\mu}$, it is sufficient to show $n^{19/200} \leq 2^{n/36}$ since $\mu \leq \frac{n}{200}$. This is indeed true for $n \geq 13$.

For the third term, by Lemma 14 we have

$$\frac{B_\mu^{(3)} n^\mu \mu!}{B_0} \leq n^3 2^{-n/2} n^\mu \mu!.$$

To see that this is at most $1/n^{16\mu}$, it is sufficient to show $n^{20\mu} \leq 2^{n/2}$, which is true when $\mu \leq \frac{n}{40 \log n}$. Adding together these three terms, we obtain $B_\mu/B_0 \leq 3/n^{16\mu}$. ◀

► **Lemma 16.** *For sufficiently large n , if $\frac{n}{200 \log n} < \mu \leq n$, then*

$$\frac{B_\mu}{B_0} \leq \frac{1}{n^{16\mu}}.$$

Proof. Lemma 14 implies $B_0 \geq 2^{n^2/4}$ for $n \geq 4$, so we have

$$\frac{B_\mu}{B_0} \leq n^{2n+1} 2^{-f(\mu)} n^\mu \mu!,$$

where $f(\mu) = \frac{\mu^2}{900} - \frac{\mu}{10}$. We claim that this expression is at most $1/n^{16\mu}$. Since $\mu! \leq n^\mu$, it is sufficient to show $n^{2n+1} n^{18\mu} \leq 2^{f(\mu)}$. When n is sufficiently large,¹ we have $\log^3 n \leq \frac{1}{200^2 \cdot 1800} \frac{n^2}{2n+1}$. Since $\mu \geq \frac{n}{200 \log n}$, this implies

$$n^{2n+1} \leq 2^{\mu^2/1800}. \quad (3)$$

When n is sufficiently large,² we also have $n \geq 18 \cdot 900 \cdot 200 \log^2 n + 90 \cdot 200 \log n$. Since $\mu \geq \frac{n}{200 \log n}$, this implies

$$n^{18\mu} \leq 2^{\mu^2/1800 - \mu/10}. \quad (4)$$

Multiplying Equations (3) and (4) gives us the desired bound of $n^{2n+1} n^{18\mu} \leq 2^{f(\mu)}$. ◀

By Lemmas 15 and 16, the above summation for $\mathbf{E}[T_1]$ is at most $T(n, 0) + O(1)$ since $T(n, \mu)$ is certainly at most $O(n^{16\mu})$. For an iteration in which we have chosen $\mu = 0$, when counting and sampling n -vertex labeled chordal graphs with automorphism $\pi = \text{id}$, we can simply run the counting and sampling algorithms of [14], rather than passing in $\pi = \text{id}$ as an input. Thus the expected running time $\mathbf{E}[T_1]$ of one iteration is at most $O(n^7)$ arithmetic operations. Lemmas 15 and 16 also immediately give us a bound on B/B_0 , since we have

$$\frac{B}{B_0} = \frac{B_0}{B_0} + \frac{B_2}{B_0} + \dots + \frac{B_n}{B_0} = O(1).$$

Therefore, by Lemma 13, the overall running time is at most $O(n^7)$ arithmetic operations in expectation.

4 Counting labeled chordal graphs with a given automorphism

In this section, we describe the algorithm for `COUNT_CHORDAL_LAB`(n, π), which counts the number of labeled chordal graphs with a given automorphism. In the full version of the paper, we prove correctness, analyze the running time, and derive the corresponding sampling algorithm (Theorem 7).

► **Theorem 6.** *There is a deterministic algorithm that given $n \in \mathbb{N}$ and $\pi \in S_n$, computes the number of labeled chordal graphs with vertex set $[n]$ for which π is an automorphism. This algorithm uses $O(2^{7\mu} n^9)$ arithmetic operations, where $\mu = |M_\pi|$.*

There is a known dynamic-programming algorithm for computing the number n -vertex labeled chordal graphs that uses $O(n^7)$ arithmetic operations, if we do not require the graphs to have a particular automorphism [14]. Our algorithm is closely based on that one, but we add more arguments and more details to each of the recurrences to keep track of the behavior of the automorphism. As was done in [14], we evaluate the recurrences top-down using memoization.

¹ This holds for $n \geq 2.6 \cdot 10^5$.

² This holds for $n \geq 3.3 \cdot 10^9$.

4.1 Reducing from counting chordal graphs to counting connected chordal graphs

For $k \in \mathbb{N}$, let $a(k)$ denote the number of labeled chordal graphs with vertex set $[k]$, and let $c(k)$ denote the number of *connected* labeled chordal graphs with vertex set $[k]$. The algorithm of [14] begins by reducing from counting chordal graphs to counting connected chordal graphs via the following recurrence, which appears as Lemma 3.13 in [14]. (We omit the “ ω -colorable” requirement since we will not need that here.)

► **Lemma 17** ([14]). *The number of labeled chordal graphs with vertex set $[k]$ is given by*

$$a(k) = \sum_{k'=1}^k \binom{k-1}{k'-1} c(k') a(k-k')$$

for all $k \in \mathbb{N}$.

Here k' stands for the number of vertices in the connected component that contains the label 1. The remaining connected components have a total of $k - k'$ vertices. Since this recurrence is relatively simple, most of the difficulty in the algorithm of [14] lies in the recurrences for counting *connected* chordal graphs. However, when counting graphs with a given automorphism, the step of reducing to connected graphs is already quite a bit more involved.

Suppose we are given $n \in \mathbb{N}$ and $\pi \in S_n$ as input. From now on, whenever we refer to n or π , we mean these particular values from the input.

We will first define $a(k, p, M)$ and $c(k, p, M)$, which are variations of $a(k)$ and $c(k)$ that only count graphs for which $\pi^p|_{V(G)}$ is an automorphism (see Definition 18). Here π^p stands for π to the power p , i.e., the permutation that arises from applying π a total of p times. The reason for raising π to a power will be apparent in the recurrence for $a(k, p, M)$. In the initial, highest-level recursive call, we will have $p = 1$ and thus $\pi^p = \pi$.

In [14], when counting the number of possibilities for a subgraph of size k' (for example, a connected component), the authors essentially relabel that subgraph to have vertex set $[k']$, so that one can count the number of possibilities using, for example, $c(k')$. In our algorithm, we want to relabel the vertices of each subgraph in a similar way. However, this time, we do not relabel the vertices that are moved by π . This ensures that the automorphism in each later recursive call will still be π , or a permutation closely related to π .

As a consequence, the vertex sets of the subgraphs that we wish to count become slightly more complicated. For example, suppose we wish to count the number of possibilities for a 5-vertex subgraph that originally contains the moved vertices $2, 8, 9 \in M_\pi$. Since we do not relabel the moved vertices, the resulting vertex set after relabeling is $\{1, 2, 3, 8, 9\}$ rather than $\{1, 2, 3, 4, 5\}$. More formally, suppose we have been given $k \in \mathbb{N}$ and $M \subseteq M_\pi$, where $|M| \leq k$. Let V be the set of the first $k - |M|$ natural numbers in $\mathbb{N} \setminus M_\pi$. We define $V_{k,M} := V \cup M$. For example, if $k = 5$ and $M = M_\pi = \{2, 8, 9\}$, then $V_{k,M} = \{1, 2, 3, 8, 9\}$. Intuitively, $V_{k,m}$ is the label set of size k whose intersection with M_π is M and that otherwise contains labels that are as small as possible.

For $\hat{\pi} \in S_n$ and $M \subseteq [n]$, recall that M is *invariant* under $\hat{\pi}$ if $\hat{\pi}(i) \in M$ for all $i \in M$. For $M', M \subseteq [n]$, we write $M' \subseteq_{\hat{\pi}} M$ to indicate that $M' \subseteq M$ and M' is invariant under $\hat{\pi}$. Intuitively, M' is a subset of M that respects the cycles of $\hat{\pi}$ by taking all or nothing of each cycle.

► **Definition 18.** *Suppose $k, p \in [n]$ and suppose $M \subseteq_{\pi^p} M_\pi$, where $|M| \leq k$. Let $a(k, p, M)$ (resp. $c(k, p, M)$) denote the number of labeled chordal graphs (resp. **connected** labeled chordal graphs) with vertex set $V_{k,M}$ for which $\pi^p|_{V(G)}$ is an automorphism.*

Our algorithm returns $a(n, 1, M_\pi)$. This is precisely the number of labeled chordal graphs with vertex set $[n]$ and automorphism π since $V_{n, M_\pi} = [n]$.

Suppose we have been given fixed values of the arguments k, p, M of $a(k, p, M)$. Let s be the smallest label in the vertex set $V_{k, M}$. For $k' \in [k]$, let $\mathcal{P}_{k'}$ be the family of sets $M' \subseteq_{\pi^p} M$ such that $|M| - k + k' \leq |M'| \leq k'$ and such that the following condition holds: if $s \in M$, then $s \in M'$, and otherwise, $|M'| \leq k' - 1$. This last condition will ensure that s belongs to the connected component of size k' in the recurrence for $a(k, p, M)$.

Suppose $C \subseteq [n]$, $\sigma \in S_n$. For an element $i \in C$, we say the *period* of i with respect to (C, σ) is the smallest positive integer j such that $\sigma^j(i) \in C$. Let \mathcal{Q} be the family of sets $C \subseteq M$ such that all elements of C have the same period $j \geq 2$ with respect to (C, π^p) , and such that $s \in C$. For a set $C \in \mathcal{Q}$, we write p_C to denote the period of the elements of C (with respect to (C, π^p)), and we let $C_\sigma := C \cup \sigma(C) \cup \dots \cup \sigma^{p_C-1}(C)$ denote the union of the sets that C is mapped to by powers of σ , where $\sigma = \pi^p$.

To compute $a(k, p, M)$, we use the following recurrence. The dot in front of the curly braces denotes multiplication. Note that we have $|M'| \leq k'$ and $|M \setminus M'| \leq k - k'$ by the definition of $\mathcal{P}_{k'}$.

► **Lemma 19.** *Let $k, p \in [n]$ and suppose $M \subseteq_{\pi^p} M_\pi$, where $|M| \leq k$. We have*

$$a(k, p, M) = \sum_{\substack{1 \leq k' \leq k \\ M' \in \mathcal{P}_{k'}}} c(k', p, M') a(k - k', p, M \setminus M') \cdot \begin{cases} \binom{k - |M|}{k' - |M'|} & \text{if } s \in M \\ \binom{k - 1 - |M|}{k' - 1 - |M'|} & \text{otherwise} \end{cases} \\ + \sum_{C \in \mathcal{Q}} c(|C|, p \cdot p_C, C) a(k - p_C |C|, p, M \setminus C_{\pi^p}).$$

For some intuition, suppose G is a graph counted by $a(k, p, M)$, and let C be the connected component of G that contains s . The first line of the recurrence for $a(k, p, M)$ covers the case when C is invariant under π^p . This case is analogous to Lemma 17. In the first summation, k' stands for $|C|$ and M' stands for the set of vertices in C that can be moved by π^p . If $s \in M$, then in addition to the vertices in M' , we need to choose $k' - |M'|$ additional vertices for C so that $|C| = k'$. For these, we must choose non-moved vertices, so there are $k - |M|$ possible vertices to choose from. If $s \notin M$, then we subtract 1 from each of the numbers in the binomial coefficient since we already know that s is a non-moved vertex in C .

The second line covers the case when C is not invariant under π^p , which means all of C is mapped to some other connected component of G by π^p . In this case, we have p_C components of size $|C|$ that are all isomorphic to C , and the rest of the graph has $k - p_C |C|$ vertices. We do not need a binomial coefficient in this case because all of the vertices in C are moved. There are $c(|C|, p \cdot p_C, C)$ possibilities for the edges of C since $\pi^{p \cdot p_C}$ is an automorphism of C . As an example, suppose $p = 1$. If we apply $\pi^p = \pi$ to the vertices of C a total of p_C times, then the image $\pi^{p_C}(C)$ is equal to C , although these two sets might not match up pointwise. To ensure that the image $\pi^{p_C}(C)$ matches up with the edges of C , we require that π^{p_C} is an automorphism of C . This is why we need the argument p in $a(k, p, M)$ and $c(k, p, M)$.

Note that for a graph G counted by $a(k, p, M)$, we have $M_{\pi^p} \cap V(G) \subseteq M_\pi \cap V(G) \subseteq M$ since $V(G) = V_{k, M}$. This means no vertex in $V(G) \setminus M$ is moved by π^p , so we are free to relabel these vertices in the proof of Lemma 19 without changing the automorphism. In the initial recursive call $a(n, 1, M_\pi)$, we have $p = 1$ and $M = M_\pi$, so $M_{\pi^p} \cap V(G) = M$. Later on in the algorithm, it is possible that some vertices in M are not actually moved by π^p . For example, in the previous paragraph, it could happen that $\pi^{p \cdot p_C}$ is the identity on C (and

then $p \cdot p_C$ becomes the new value of p). However, we always have $M \subseteq M_\pi$ by the definition of $a(k, p, M)$, so if $|M_\pi| = \mu$, then $|M| \leq \mu$. This fact will be useful for the running time analysis.

The proof of Lemma 19, along with the proofs of all of the following recurrences, can be found in the full version of the paper.

4.2 Recurrences for counting connected chordal graphs

To compute $c(k, p, M)$, we will define various counter functions that are analogous to those in [14]. First, we recall the counter functions from [14]. We refer to functions 1-4 in Definition 20 as g -functions, and we refer to the others as f -functions. See Figure 1 in [14] for an illustration of all of these functions.

► **Definition 20** ([14]). *The following functions count various subclasses of chordal graphs. The arguments t, x, ℓ, k, z are nonnegative integers, where $t \leq n$, $z \leq n$, $x + k \leq n$ for g -functions, and $x + \ell + k \leq n$ for f -functions. These also satisfy the domain requirements listed below.*

1. $g(t, x, k, z)$ is the number of labeled connected chordal graphs G with vertex set $[x + k]$ that evaporate in time at most t with exception set $X := [x]$, where X is a clique, with the following property: every connected component of $G \setminus X$ (if any) has at least one neighbor in $X \setminus [z]$. **Domain:** $t \geq 0$, $x \geq 1$, $z < x$.
2. $\tilde{g}(t, x, k, z)$ is the same as $g(t, x, k, z)$, except every connected component of $G \setminus X$ (if any) evaporates at time exactly t in G . Note: A graph with $V(G) = X$ would be counted because in that case, \tilde{g} is the same as g . **Domain:** $t \geq 1$, $x \geq 1$, $z < x$.
3. $\tilde{g}_p(t, x, k, z)$ is the same as $\tilde{g}(t, x, k, z)$, except no connected component of $G \setminus X$ sees all of X . **Domain:** $t \geq 1$, $x \geq 1$, $z < x$.
4. $\tilde{g}_1(t, x, k)$ and $\tilde{g}_{\geq 2}(t, x, k)$ are the same as $\tilde{g}(t, x, k, z)$, except every connected component of $G \setminus X$ sees all of X (hence we no longer require every component of $G \setminus X$ to have a neighbor in $X \setminus [z]$), and furthermore, for \tilde{g}_1 we require that $G \setminus X$ has exactly one connected component, and for $\tilde{g}_{\geq 2}$ we require that $G \setminus X$ has at least two components. **Domain for \tilde{g}_1 :** $t \geq 1$, $x \geq 0$. **Domain for $\tilde{g}_{\geq 2}$:** $t \geq 1$, $x \geq 1$.
5. $f(t, x, \ell, k)$ is the number of labeled connected chordal graphs G with vertex set $[x + \ell + k]$ that evaporate at time exactly t with exception set $X := [x]$, such that $G \setminus X$ is connected, $L_G(X) = \{x + 1, \dots, x + \ell\}$, and $X \cup L_G(X)$ is a clique. **Domain:** $t \geq 1$, $x \geq 0$, $\ell \geq 1$.
6. $\tilde{f}(t, x, \ell, k)$ is the same as $f(t, x, \ell, k)$, except every connected component of $G \setminus (X \cup L_G(X))$ evaporates at time exactly $t - 1$ in G , and there exists at least one such component, i.e., $X \cup L_G(X) \subsetneq V(G)$. **Domain:** $t \geq 2$, $x \geq 0$, $\ell \geq 1$.
7. $\tilde{f}_p(t, x, \ell, k)$ is the same as $\tilde{f}(t, x, \ell, k)$, except no connected component of $G \setminus (X \cup L_G(X))$ sees all of $X \cup L_G(X)$. **Domain:** $t \geq 2$, $x \geq 0$, $\ell \geq 1$.
8. $\tilde{f}_p(t, x, \ell, k, z)$ is the same as $\tilde{f}_p(t, x, \ell, k)$, except rather than requiring that $G \setminus X$ is connected, we require that $G \setminus [z]$ is connected. **Domain:** $t \geq 2$, $x \geq 0$, $\ell \geq 1$, $z \leq x$.

The counter functions for our algorithm will be similar to these, except we only want to count graphs with a particular automorphism. Therefore, we will have several more arguments in addition to the usual ones t, x, ℓ, k, z from Definition 20. As above, the argument p will indicate that π^p is the current automorphism. We also introduce the arguments M_X, M_L, M_Z , and M_K , each of which is a subset of M_π . Roughly, these sets specify which vertices – in $X, L := L_G(X), Z := [z]$, and the rest of the graph, respectively – can be moved by the current permutation (but the sets X, L , and Z will be modified slightly).

In Definition 20, the g -functions count graphs with vertex set $[x + k]$, and the f -functions count graphs with vertex set $[x + \ell + k]$. For our algorithm, we will make some adjustments to these vertex sets to ensure that the vertices moved by the current permutation still appear in the graph. This is similar to how we defined $V_{k,M}$ above. To do so, we define several symbols for these vertex sets and vertex subsets (V_{args} , etc.). Each of these depends on the list of arguments of the function in question, which we denote by $args$. For example, when defining g , we have $args = \begin{pmatrix} t & x & k & z \\ p & M_X & M_K & M_Z \end{pmatrix}$ (see Definition 21).

First, suppose $args$ comes from one of the g -functions in Definition 21. Let V_X be the set of the first $x - |M_X|$ natural numbers in $\mathbb{N} \setminus M_\pi$, and let V_K be the set of the first $k - |M_K|$ natural numbers in $\mathbb{N} \setminus (V_X \cup M_\pi)$. We define $X_{args} := V_X \cup M_X$ and $V_{args} := X_{args} \cup V_K \cup M_K$. Also, if z is included in $args$, then let V_Z be the set of the first $z - |M_Z|$ natural numbers in $\mathbb{N} \setminus M_\pi$. In this case, we define $Z_{args} := V_Z \cup M_Z$.

For the other case, suppose $args$ comes from one of the f -functions. Let V_X be the set of the first $x - |M_X|$ natural numbers in $\mathbb{N} \setminus M_\pi$, let V_L be the set of the first $\ell - |M_L|$ natural numbers in $\mathbb{N} \setminus (V_X \cup M_\pi)$, and let V_K be the set of the first $k - |M_K|$ natural numbers in $\mathbb{N} \setminus (V_X \cup V_L \cup M_\pi)$. We define $X_{args} := V_X \cup M_X$, $L_{args} = V_L \cup M_L$, and $V_{args} := X_{args} \cup L_{args} \cup V_K \cup M_K$. Also, if z is included in $args$, then let V_Z be the set of the first $z - |M_Z|$ natural numbers in $\mathbb{N} \setminus M_\pi$. In this case, we again define $Z_{args} := V_Z \cup M_Z$.

These vertex subsets still have the same sizes as they did in the original algorithm: the size of V_{args} is $x + k$ or $x + \ell + k$, $|X_{args}| = x$, $|L_{args}| = \ell$, the rest of the graph has size k , and $|Z_{args}| = z$. Just as we had $Z \subseteq X$ in the original algorithm, we now have $Z_{args} \subseteq X_{args}$. This is because we require $M_Z \subseteq M_X$ in Definition 21, and we also require $z - |M_Z| \leq x - |M_X|$, which implies $V_Z \subseteq V_X$.

For g -functions, we have $M_{\pi^p} \cap V_{args} \subseteq M_X \cup M_K$, and for f -functions, we have $M_{\pi^p} \cap V_{args} \subseteq M_X \cup M_L \cup M_K$, by the definition of V_{args} . We are now ready to define our new counter functions.

► **Definition 21.** *The following functions count various subclasses of chordal graphs. The arguments t, x, ℓ, k, z are nonnegative integers with the same domains as in Definition 20. We have $p \in [n]$, and the arguments M_X, M_L, M_K, M_Z are subsets of M_π . All other requirements for their domains are specified below.*

1. $g \begin{pmatrix} t & x & k & z \\ p & M_X & M_K & M_Z \end{pmatrix}, \tilde{g} \begin{pmatrix} t & x & k & z \\ p & M_X & M_K & M_Z \end{pmatrix}, \tilde{g}_p \begin{pmatrix} t & x & k & z \\ p & M_X & M_K & M_Z \end{pmatrix},$
 $\tilde{g}_1 \begin{pmatrix} t & x & k \\ p & M_X & M_K \end{pmatrix},$ and $\tilde{g}_{\geq 2} \begin{pmatrix} t & x & k \\ p & M_X & M_K \end{pmatrix}$ are the same as $g(t, x, k, z), \tilde{g}(t, x, k, z),$
 $\tilde{g}_p(t, x, k, z), \tilde{g}_1(t, x, k),$ and $\tilde{g}_{\geq 2}(t, x, k),$ respectively, except we only count graphs for which $\pi^p|_{V(G)}$ is an automorphism, and we make the following changes to the vertices of the graph: the vertex set is V_{args} rather than $[x + k]$, the exception set is X_{args} rather than $[x]$, and $[z]$ is replaced by Z_{args} .

Domain: $M_X \subseteq_{\pi^p} M_\pi, M_K \subseteq_{\pi^p} M_\pi \setminus M_X, M_Z \subseteq_{\pi^p} M_X, |M_X| \leq x, |M_K| \leq k, |M_Z| \leq z,$ and $z - |M_Z| \leq x - |M_X|.$

2. $f \begin{pmatrix} t & x & \ell & k \\ p & M_X & M_L & M_K \end{pmatrix}, \tilde{f} \begin{pmatrix} t & x & \ell & k \\ p & M_X & M_L & M_K \end{pmatrix}, \tilde{f}_p \begin{pmatrix} t & x & \ell & k \\ p & M_X & M_L & M_K \end{pmatrix},$ and
 $\tilde{f}_p \begin{pmatrix} t & x & \ell & k & z \\ p & M_X & M_L & M_K & M_Z \end{pmatrix}$ are the same as $f(t, x, \ell, k), \tilde{f}(t, x, \ell, k), \tilde{f}_p(t, x, \ell, k),$ and
 $\tilde{f}_p(t, x, \ell, k, z),$ respectively, except we only count graphs for which $\pi^p|_{V(G)}$ is an automorphism, and we make the following changes to the vertices of the graph: the vertex set is

46:16 Sampling Unlabeled Chordal Graphs in Expected Polynomial Time

V_{args} rather than $[x + \ell + k]$, the exception set is X_{args} rather than $[x]$, the last set to evaporate is L_{args} rather than $\{x + 1, \dots, x + \ell\}$, and $[z]$ is replaced by Z_{args} .

Domain: $M_X \subseteq_{\pi^p} M_\pi$, $M_L \subseteq_{\pi^p} M_\pi \setminus M_X$, $M_K \subseteq_{\pi^p} M_\pi \setminus (M_X \cup M_L)$, $M_Z \subseteq_{\pi^p} M_X$, $|M_X| \leq x$, $|M_L| \leq \ell$, $|M_K| \leq k$, $|M_Z| \leq z$, and $z - |M_Z| \leq x - |M_X|$.

For a graph G counted by one of these functions, $\pi^p|_{V(G)}$ is indeed a permutation of $V(G)$ since M_X and M_K (and M_L if needed) are invariant under π^p .

To compute $c(k, p, M)$, we consider all possibilities for the evaporation time. In the following recurrence, \tilde{g}_1 (with those particular arguments) counts the number of labeled connected chordal graphs with vertex set $V_{args} = V_{k, M}$ and automorphism $\pi^p|_{V(G)}$ that evaporate at time exactly t with empty exception set.

► **Lemma 22.** *Let $k, p \in [n]$ and suppose $M \subseteq_{\pi^p} M_\pi$, where $|M| \leq k$. We have*

$$c(k, p, M) = \sum_{t=1}^k \tilde{g}_1 \begin{pmatrix} t & 0 & k \\ p & \emptyset & M \end{pmatrix}.$$

To compute \tilde{g}_1 , we consider all possibilities for the size ℓ of L_{args} . The set M in the inner sum stands for the set of vertices in L_{args} that can be moved by π^p . Note that we have $|M| \leq \ell$ and $|M_K \setminus M| \leq k - \ell$ by the definition of \mathcal{I}_ℓ . For \tilde{g}_1 and all of the following functions, we recommend reading the analogous recurrences in [14] for further insight into what is happening with the arguments t, x, ℓ, k, z in each recursive call.

► **Lemma 23.** *For \tilde{g}_1 , we have*

$$\tilde{g}_1 \begin{pmatrix} t & x & k \\ p & M_X & M_K \end{pmatrix} = \sum_{\ell=1}^k \sum_{M \in \mathcal{I}_\ell} \binom{k - |M_K|}{\ell - |M|} f \begin{pmatrix} t & x & \ell & k - \ell \\ p & M_X & M & M_K \setminus M \end{pmatrix},$$

where $\mathcal{I}_\ell = \{M \subseteq_{\pi^p} M_K : |M_K| - k + \ell \leq |M| \leq \ell\}$.

To compute f , we consider all possibilities for the set of labels that appear in connected components of $G \setminus (X_{args} \cup L_{args})$ that evaporate at time exactly $t - 1$. These components correspond to the recursive call to \tilde{f} . The set M stands for the set of vertices in components that evaporate at time exactly $t - 1$ that can be moved by π^p . Note that we have $|M| \leq k'$ and $|M_K \setminus M| \leq k - k'$ by the definition of $\mathcal{I}_{k'}$. Since $|M_L| \leq \ell$, we also have $x - |M_X| \leq x + \ell - |M_X \cup M_L|$, which is required by the domain of g .

► **Lemma 24.** *For f , we have*

$$f \begin{pmatrix} t & x & \ell & k \\ p & M_X & M_L & M_K \end{pmatrix} = \sum_{k'=1}^k \sum_{M \in \mathcal{I}_{k'}} \binom{k - |M_K|}{k' - |M|} \tilde{f} \begin{pmatrix} t & x & \ell & k' \\ p & M_X & M_L & M \end{pmatrix} g \begin{pmatrix} t - 2 & x + \ell & k - k' & x \\ p & M_X \cup M_L & M_K \setminus M & M_X \end{pmatrix},$$

where $\mathcal{I}_{k'} = \{M \subseteq_{\pi^p} M_K : |M_K| - k + k' \leq |M| \leq k'\}$.

To compute g , we consider all possibilities for the set of labels that appear in connected components of $G \setminus X_{args}$ that evaporate at time exactly t . These components correspond to the recursive call to \tilde{g} . The set M stands for the set of vertices in components that evaporate at time exactly t that can be moved by π^p .

► **Lemma 25.** *For g , we have*

$$g \binom{t \quad x \quad k \quad z}{p \quad M_X \quad M_K \quad M_Z} = \sum_{k'=0}^k \sum_{M \in \mathcal{I}_{k'}} \binom{k - |M_K|}{k' - |M|} \tilde{g} \binom{t \quad x \quad k' \quad z}{p \quad M_X \quad M \quad M_Z} g \binom{t-1 \quad x \quad k-k' \quad z}{p \quad M_X \quad M_K \setminus M \quad M_Z},$$

where $\mathcal{I}_{k'} = \{M \subseteq_{\pi^p} M_K : |M_K| - k + k' \leq |M| \leq k'\}$.

For the next recurrence, we will need a few more definitions. Suppose we have been given fixed values of the arguments of \tilde{g} . Let s be the smallest label in $V_{args} \setminus X_{args}$. For $k' \in [k]$, let $\mathcal{P}_{k'}$ be the family of sets $M \subseteq_{\pi^p} M_K$ such that $|M_K| - k + k' \leq |M| \leq k'$ and such that the following condition holds: if $s \in M_K$, then $s \in M$, and otherwise, $|M| \leq k' - 1$. For $x' \in [x]$, let $\mathcal{I}_{x'} = \{M' \subseteq_{\pi^p} M_X : |M'| \leq x'\}$.

Let \mathcal{Q} be the family of pairs (C, M') , where $C \subseteq M_K$ and $M' \subseteq M_X$, such that all elements of C have the same period $p_C \geq 2$ with respect to (C, π^p) , such that $s \in C$, and such that M' is invariant under $\pi^{p \cdot p_C}$. For a set C from such a pair, we let $C_\sigma := C \cup \sigma(C) \cup \dots \cup \sigma^{p_C-1}(C)$, where $\sigma = \pi^p$.

To compute \tilde{g} , we consider all possibilities for the label set of the connected component C of $G \setminus X_{args}$ that contains s . In the definition of \tilde{g} , the components of $G \setminus X_{args}$ are similar enough (since they all evaporate at time t) that they can potentially be mapped to one another by π^p . Thus we consider two cases: either C is invariant under π^p , or C is mapped to some other component of $G \setminus X_{args}$ by π^p . We add together the summations from these two cases, in a similar way to the recurrence for $a(k, p, M)$. In the recurrence for \tilde{g} , k' stands for $|C|$, and x' stands for $|N(C)|$. The set M stands for the set of vertices in C that can be moved by π^p , and M' stands for the set of vertices in $N(C)$ that can be moved by π^p .

► **Lemma 26.** *For \tilde{g} , we have*

$$\begin{aligned} \tilde{g} \binom{t \quad x \quad k \quad z}{p \quad M_X \quad M_K \quad M_Z} &= \sum_{\substack{1 \leq k' \leq k \\ 1 \leq x' \leq x \\ M \in \mathcal{P}_{k'} \\ M' \in \mathcal{I}_{x'}}} \tilde{g}_1 \binom{t \quad x' \quad k'}{p \quad M' \quad M} \tilde{g} \binom{t \quad x \quad k-k' \quad z}{p \quad M_X \quad M_K \setminus M \quad M_Z} \\ &\quad \cdot \begin{cases} \binom{k - |M_K|}{k' - |M|} & \text{if } s \in M_K \\ \binom{k-1 - |M_K|}{k'-1 - |M|} & \text{otherwise} \end{cases} \\ &\quad \cdot \begin{cases} \binom{x - |M_X|}{x' - |M'|} & \text{if } M' \not\subseteq M_Z \\ \binom{x - |M_X|}{x' - |M'|} - \binom{z - |M_Z|}{x' - |M'|} & \text{otherwise} \end{cases} \\ &+ \sum_{\substack{1 \leq x' \leq x \\ (C, M') \in \mathcal{Q}}} \tilde{g}_1 \binom{t \quad x' \quad |C|}{p \cdot p_C \quad M' \quad C} \tilde{g} \binom{t \quad x \quad k - p_C |C| \quad z}{p \quad M_X \quad M_K \setminus C_{\pi^p} \quad M_Z} \\ &\quad \cdot \begin{cases} \binom{x - |M_X|}{x' - |M'|} & \text{if } M' \not\subseteq M_Z \\ \binom{x - |M_X|}{x' - |M'|} - \binom{z - |M_Z|}{x' - |M'|} & \text{otherwise.} \end{cases} \end{aligned}$$

To compute \tilde{f} , we consider three cases: either no component of $G \setminus (X_{args} \cup L_{args})$ sees all of $X_{args} \cup L_{args}$, exactly one component sees all of $X_{args} \cup L_{args}$, or at least two components see all of $X_{args} \cup L_{args}$. The recursive calls to \tilde{g}_1 and $\tilde{g}_{\geq 2}$ correspond to the component/component(s) that see all of $X_{args} \cup L_{args}$. In the second and third cases, the set M stands for the set of vertices that can be moved by π^p in components that see all of $X_{args} \cup L_{args}$.

► **Lemma 27.** For \tilde{f} , we have

$$\begin{aligned} \tilde{f} \binom{t \quad x \quad \ell \quad k}{p \quad M_X \quad M_L \quad M_K} &= \tilde{f}_p \binom{t \quad x \quad \ell \quad k}{p \quad M_X \quad M_L \quad M_K} \\ &+ \sum_{\substack{1 \leq k' \leq k \\ M \in \mathcal{I}_{k'}}} \binom{k - |M_K|}{k' - |M|} \tilde{g}_1 \binom{t-1 \quad x+\ell \quad k'}{p \quad M_X \cup M_L \quad M} \tilde{f}_p \binom{t \quad x \quad \ell \quad k-k'}{p \quad M_X \quad M_L \quad M_K \setminus M} \\ &+ \sum_{\substack{1 \leq k' \leq k \\ M \in \mathcal{I}_{k'}}} \binom{k - |M_K|}{k' - |M|} \tilde{g}_{\geq 2} \binom{t-1 \quad x+\ell \quad k'}{p \quad M_X \cup M_L \quad M} \tilde{g}_p \binom{t-1 \quad x+\ell \quad k-k' \quad x}{p \quad M_X \cup M_L \quad M_K \setminus M \quad M_X}, \end{aligned}$$

where $\mathcal{I}_{k'} = \{M \subseteq_{\pi^p} M_K : |M_K| - k + k' \leq |M| \leq k'\}$.

For the next recurrence, suppose we have been given fixed values of the arguments of $\tilde{g}_{\geq 2}$. Let s be the smallest label in $V_{args} \setminus X_{args}$, and let $\mathcal{P}_{k'}$ be defined as it was in the recurrence for \tilde{g} . Let \mathcal{Q} be the family of sets $C \subseteq M$ such that all elements of C have the same period $p_C \geq 2$ with respect to (C, π^p) , and such that $s \in C$. For a set $C \in \mathcal{Q}$, we let $C_\sigma := C \cup \sigma(C) \cup \dots \cup \sigma^{p_C-1}(C)$, where $\sigma = \pi^p$.

► **Lemma 28.** For $\tilde{g}_{\geq 2}$, we have

$$\begin{aligned} \tilde{g}_{\geq 2} \binom{t \quad x \quad k}{p \quad M_X \quad M_K} &= \\ &\sum_{\substack{1 \leq k' \leq k \\ M \in \mathcal{P}_{k'}}} \tilde{g}_1 \binom{t \quad x \quad k'}{p \quad M_X \quad M} \left(\tilde{g}_1 \binom{t \quad x \quad k-k'}{p \quad M_X \quad M_K \setminus M} + \tilde{g}_{\geq 2} \binom{t \quad x \quad k-k'}{p \quad M_X \quad M_K \setminus M} \right) \\ &\quad \cdot \begin{cases} \binom{k-|M_K|}{k'-|M|} & \text{if } s \in M_K \\ \binom{k-1-|M_K|}{k'-1-|M|} & \text{otherwise} \end{cases} \\ &+ \sum_{C \in \mathcal{Q}} \tilde{g}_1 \binom{t \quad x \quad |C|}{p \cdot p_C \quad M_X \quad C} \left(\tilde{g}_1 \binom{t \quad x \quad k-p_C|C|}{p \quad M_X \quad M_K \setminus C^\pi} + \tilde{g}_{\geq 2} \binom{t \quad x \quad k-p_C|C|}{p \quad M_X \quad M_K \setminus C^\pi} \right). \end{aligned}$$

To compute \tilde{g}_p , all we need to do is make a slight adjustment to the recurrence for \tilde{g} .

► **Lemma 29.** The recurrence for \tilde{g}_p is exactly the same as the recurrence for \tilde{g} in Lemma 26, except for the two sums over x' : In both summations, rather than summing over x' such that $1 \leq x' \leq x$, we sum over x' such that $1 \leq x' \leq x-1$.

To compute \tilde{f}_p , we first observe that when $z = x$, requiring $G \setminus Z_{args}$ to be connected is the same as requiring $G \setminus X_{args}$ to be connected.

► **Lemma 30.** We have $\tilde{f}_p \binom{t \quad x \quad \ell \quad k}{p \quad M_X \quad M_L \quad M_K} = \tilde{f}_p \binom{t \quad x \quad \ell \quad k \quad x}{p \quad M_X \quad M_L \quad M_K \quad M_X}$.

For the next \tilde{f}_p recurrence, we need one last round of definitions. Suppose we have been given fixed values of the arguments of \tilde{f}_p , including z and M_Z . Let s be the smallest label in $V_{args} \setminus (X_{args} \cup L_{args})$. For $k' \in [k]$, let $\mathcal{P}_{k'}$ be the family of sets $M \subseteq_{\pi^p} M_K$ such that $|M_K| - k + k' \leq |M| \leq k'$ and such that the following condition holds: if $s \in M_K$, then $s \in M$, and otherwise, $|M| \leq k' - 1$. For $0 \leq x' \leq x$, let $\mathcal{I}_{x'} = \{M' \subseteq_{\pi^p} M_X : |M'| \leq x'\}$. Also, for $0 \leq \ell' \leq \ell$, let $\mathcal{J}_{\ell'} = \{M' \subseteq_{\pi^p} M_L : |M'| \leq \ell'\}$.

Let \mathcal{Q} be the family of triples (C, M'_X, M'_L) , where $C \subseteq M_K$, $M'_X \subseteq M_X$, and $M'_L \subseteq M_L$, such that all elements of C have the same period $p_C \geq 2$ with respect to (C, π^p) , such that $s \in C$, and such that $M'_X \cup M'_L$ is invariant under π^{p_C} . For a set C from such a triple, we let $C_\sigma := C \cup \sigma(C) \cup \dots \cup \sigma^{p_C-1}(C)$, where $\sigma = \pi^p$.

► **Lemma 31.** For \tilde{f}_p with the argument z , we have

$$\begin{aligned}
 \tilde{f}_p \begin{pmatrix} t & x & \ell & k & z \\ p & M_X & M_L & M_K & M_Z \end{pmatrix} &= \sum_{\substack{1 \leq k' \leq k \\ 0 \leq x' \leq x \\ 0 \leq \ell' \leq \ell \\ 0 < x' + \ell' < x + \ell}} \sum_{\substack{M \in \mathcal{P}_{k'} \\ M'_X \in \mathcal{I}_{x'} \\ M'_L \in \mathcal{J}_{\ell'}}} \tilde{g}_1 \begin{pmatrix} t-1 & x'+\ell' & k' \\ p & M'_X \cup M'_L & M \end{pmatrix} \\
 &\cdot \begin{pmatrix} \ell - |M_L| \\ \ell' - |M'_L| \end{pmatrix} \cdot \begin{cases} \binom{k-|M_K|}{k'-|M|} & \text{if } s \in M_K \\ \binom{k-1-|M_K|}{k'-1-|M|} & \text{otherwise} \end{cases} \\
 &\cdot \begin{cases} \binom{x-|M_X|}{x'-|M'_X|} & \text{if } \ell' > 0 \text{ or } M'_X \not\subseteq M_Z \\ \binom{x-|M_X|}{x'-|M'_X|} - \binom{z-|M_Z|}{x'-|M'_X|} & \text{otherwise} \end{cases} \\
 &\cdot \begin{cases} \tilde{f}_p \begin{pmatrix} t & x+\ell' & \ell-\ell' & k-k' & z \\ p & M_X \cup M'_L & M_L \setminus M'_L & M_K \setminus M & M_Z \end{pmatrix} & \text{if } \ell' < \ell \\ \tilde{g}_p \begin{pmatrix} t-1 & x+\ell & k-k' & z \\ p & M_X \cup M'_L & M_K \setminus M & M_Z \end{pmatrix} & \text{otherwise} \end{cases} \\
 + \sum_{\substack{0 \leq x' \leq x \\ 0 \leq \ell' \leq \ell \\ 0 < x' + \ell' < x + \ell \\ (C, M'_X, M'_L) \in \mathcal{Q}}} \tilde{g}_1 \begin{pmatrix} t-1 & x'+\ell' & |C| \\ p \cdot p_C & M'_X \cup M'_L & C \end{pmatrix} \cdot \begin{pmatrix} \ell - |M_L| \\ \ell' - |M'_L| \end{pmatrix} \\
 \cdot \begin{cases} \binom{x-|M_X|}{x'-|M'_X|} & \text{if } \ell' > 0 \text{ or } M'_X \not\subseteq M_Z \\ \binom{x-|M_X|}{x'-|M'_X|} - \binom{z-|M_Z|}{x'-|M'_X|} & \text{otherwise} \end{cases} \\
 \cdot \begin{cases} \tilde{f}_p \begin{pmatrix} t & x+\ell' & \ell-\ell' & k-p_C|C| & z \\ p & M_X \cup M'_L & M_L \setminus M'_L & M_K \setminus C^\pi & M_Z \end{pmatrix} & \text{if } \ell' < \ell \\ \tilde{g}_p \begin{pmatrix} t-1 & x+\ell & k-p_C|C| & z \\ p & M_X \cup M'_L & M_K \setminus C^\pi & M_Z \end{pmatrix} & \text{otherwise.} \end{cases}
 \end{aligned}$$

See the full version of the paper for more intuition behind the recurrences for $\tilde{g}_{\geq 2}$ and \tilde{f}_p . The base cases for all of these counter functions are the same as in [14] since they only depend on the arguments t, x, ℓ, k, z .

5 Conclusion

We built upon the algorithm of Wormald for generating random unlabeled graphs to design an algorithm that, given n , generates a random unlabeled chordal graph on n vertices in expected polynomial time. This serves as a proof of concept that one can obtain a sampling algorithm for unlabeled graphs from a GI-complete graph class \mathcal{G} using the following two ingredients: (1) an FPT algorithm for counting labeled graphs in \mathcal{G} with a given automorphism π parameterized by the number of moved points of π and (2) a bound on the probability that a labeled graph in \mathcal{G} has a given automorphism. A few potential candidates for this are bipartite graphs, strongly chordal graphs, and chordal bipartite graphs, all of which are GI-complete. An additional open problem would be to design a uniform, or approximately uniform, sampling algorithm – either for unlabeled chordal graphs or general unlabeled graphs – that runs in expected polynomial time even when we condition on the output graph.

References

- 1 Edward A. Bender, L. Bruce Richmond, and Nicholas C. Wormald. Almost all chordal graphs split. *Journal of the Australian Mathematical Society*, 38(2):214–221, 1985.
- 2 Vladislav Bína and Jiří Přibíl. Note on enumeration of labeled split graphs. *Commentationes Mathematicae Universitatis Carolinae*, 56(2):133–137, 2015.
- 3 Jean R.S. Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.
- 4 Manuel Bodirsky, Clemens Gröpl, and Mihyun Kang. Sampling unlabeled biconnected planar graphs. In *Algorithms and Computation: 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005. Proceedings 16*, pages 593–603. Springer, 2005. doi:10.1007/11602613_60.
- 5 Manuel Bodirsky, Clemens Gröpl, and Mihyun Kang. Generating labeled planar graphs uniformly at random. *Theoretical Computer Science*, 379(3):377–386, 2007. doi:10.1016/J.TCS.2007.02.045.
- 6 Manuel Bodirsky, Clemens Gröpl, and Mihyun Kang. Generating unlabeled connected cubic planar graphs uniformly at random. *Random Structures & Algorithms*, 32(2):157–180, 2008. doi:10.1002/RSA.20206.
- 7 John D. Dixon and Herbert S. Wilf. The random selection of unlabeled graphs. *Journal of Algorithms*, 4(3):205–213, 1983. doi:10.1016/0196-6774(83)90021-4.
- 8 Éric Fusy. Uniform random sampling of planar graphs in linear time. *Random Structures & Algorithms*, 35(4):464–522, 2009. doi:10.1002/RSA.20275.
- 9 Andrew Gainer-Dewar and Ira M. Gessel. Counting unlabeled k -trees. *Journal of Combinatorial Theory, Series A*, 126:177–193, 2014. doi:10.1016/J.JCTA.2014.05.002.
- 10 Pu Gao and Nicholas Wormald. Uniform generation of random regular graphs. *SIAM Journal on Computing*, 46:1395–1427, 2017. doi:10.1137/15M1052779.
- 11 Pu Gao and Nicholas Wormald. Uniform generation of random graphs with power-law degree sequences. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1741–1758, 2018. doi:10.1137/1.9781611975031.114.
- 12 Fănică Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972. doi:10.1137/0201013.
- 13 András Hajnal and János Surányi. Über die auflösung von graphen in vollständige teilgraphen. *Ann. Univ. Sci. Budapest, Eötvös Sect. Math*, 1:113–121, 1958.
- 14 Úrsula Hébert-Johnson, Daniel Lokshtanov, and Eric Vigoda. Counting and sampling labeled chordal graphs in polynomial time, 2023. doi:10.48550/arXiv.2308.09703.
- 15 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986. doi:10.1016/0304-3975(86)90174-X.
- 16 Walter Oberschelp. Kombinatorische anzahlbestimmungen in relationen. *Mathematische Annalen*, 174:53–78, 1967.
- 17 OEIS Foundation Inc. Entry A058862 in the On-Line Encyclopedia of Integer Sequences, 2024. Published electronically at <http://oeis.org>.
- 18 Toshiki Saitoh, Yota Otachi, Katsuhisa Yamanaka, and Ryuhei Uehara. Random generation and enumeration of bipartite permutation graphs. *Journal of Discrete Algorithms*, 10:84–97, 2012. doi:10.1016/J.JDA.2011.11.001.
- 19 Herbert S. Wilf. The uniform selection of free trees. *Journal of Algorithms*, 2(2):204–207, 1981. doi:10.1016/0196-6774(81)90021-3.
- 20 Nicholas C. Wormald. Generating random unlabelled graphs. *SIAM Journal on Computing*, 16(4):717–727, 1987. doi:10.1137/0216048.