

---

# Distribution Learning Meets Graph Structure

## Sampling: Supplementary Material

---

**Arnab Bhattacharyya**

University of Warwick

arnab.bhattacharyya@warwick.ac.uk

**Sutanu Gayen**

IIT Kanpur

sutanugayen@gmail.com

**Philips George John**

CNRS@CREATE & Dept of Computer Science

National University of Singapore

philips.george.john@u.nus.edu

**Sayantana Sen**

Centre for Quantum Technologies

National University of Singapore

sayantan789@gmail.com

**N. V. Vinodchandran**

University of Nebraska-Lincoln

vinod@cse.unl.edu

**Organization of the appendices** The appendices are organized as follows. In Appendix A, we present the preliminaries required for this work. Appendix B establishes the connection between regret in online learning to KL divergence in the scenario of agnostic learning of distributions. It also presents several necessary techniques from online learning along with the EWA and RWM algorithms that will be used later in our work. In Appendix C, we present our results on learning chordal-structured distributions. In Appendix D, we discuss our results on learning tree-structured distributions and present our alternative proper learning algorithm. In Appendix E, we give the lower bound of learning tree-structured distributions. In Appendix F, we design efficient learning algorithms for Bayes nets over graphs with bounded vertex cover. In Appendix G, we outline how our algorithms can be adapted to efficiently compute maximum likelihood.

## A Preliminaries

For integers  $0 < m \leq n$ , let  $[n]$  denote the set  $\{1, \dots, n\}$ , and let  $[m, n]$  denote the set  $\{m, m+1, \dots, n\}$ . For any  $\eta > 0$ , let  $\exp_\eta(u)$  denote  $e^{-\eta u}$ . For concise expressions and readability, we use the asymptotic complexity notion of  $\tilde{O}(\cdot)$ , where we hide poly-logarithmic dependencies of the parameters. By stating i.i.d samples from a distribution  $P$ , we mean independently and identically distributed samples from  $P$ . For a positive integer  $\ell$ ,  $\text{Unif}([ \ell ])$  denotes the uniform distribution on the set  $[ \ell ]$ , where each  $i \in [ \ell ]$  is chosen with equal probability of  $1/\ell$ .

### A.1 Probability Distributions

We let  $\Delta(\mathcal{D})$  denote the set of probability distributions over the elements of a set  $\mathcal{D}$ . Let  $P$  and  $Q$  be two such distributions over  $\mathcal{D}$ :

- The *KL-divergence* between  $P$  and  $Q$  is defined as:  $D_{\text{KL}}(P, Q) = \sum_{x \in \mathcal{D}} P(x) \log \frac{P(x)}{Q(x)}$ .
- The *total variation (TV) distance* between  $P$  and  $Q$  is defined as:  $d_{\text{TV}}(P, Q) = \sum_{x \in \mathcal{D}} |P(x) - Q(x)|$ .

**Lemma A.1** (Pinkser’s inequality). *Let  $P$  and  $Q$  be two probability distributions defined over the same sample space  $\mathcal{D}$ . Then the following holds:*

$$d_{TV}(P, Q) \leq \sqrt{\frac{D_{KL}(P||Q)}{2}}$$

We study distributions over high-dimensional spaces, which can require an exponential amount of space to represent in general. So, for computational efficiency, we focus on distributions from which we can generate samples efficiently. This notion is formally defined below.

**Definition A.2** (Efficiently samplable distribution). A distribution  $P$  is said to be *efficiently samplable* if there exists a probabilistic Turing machine  $M$  which on input  $0^k$  produces a string  $x$  such that  $|\Pr[M(0^k) = x] - P(x)| \leq 2^{-k}$  and  $M$  runs in time  $\text{poly}(|x| + k)$ .

Next we define graphical models of interest in this work. For a directed graph  $G$  on  $n$  nodes, we will identify its set of nodes with  $[n]$ . The underlying undirected graph of  $G$  is called the *skeleton*. For any  $i \in [n]$ , let  $\text{pa}_G(i)$  denote the set of the parents of node  $i$  and  $\text{nd}_G(i)$  denote the set of its non-descendants. The subscript  $G$  may be removed if it is clear from the context.

Let us start with the definition of Bayesian networks.

**Definition A.3** (Bayesian networks). A probability distribution  $P$  over  $n$  variables  $X_1, \dots, X_n$  is said to be a *Bayesian network* (*Bayes net in short*) on a *directed acyclic graph*  $G$  with  $n$  nodes if<sup>1</sup> for every  $i \in [n]$ ,  $X_i$  is conditionally independent of  $X_{\text{nd}(i)}$  given  $X_{\text{pa}(i)}$ . Equivalently,  $P$  admits the factorization:

$$P(x) = \Pr_{X \sim P}[X = x] = \prod_{i=1}^n \Pr_{X \sim P}[X_i = x_i \mid \forall j \in \text{pa}(i), X_j = x_j] \quad \text{for all } x. \quad (1)$$

It is well-known that Bayesian networks are efficiently samplable.

Now we define tree-structured distributions, a subclass of Bayesian networks defined above.

**Definition A.4** (Tree-structured distribution). Let  $T$  be a tree, and  $G$  be any rooted orientation of  $T$ . A probability distribution  $P$  is said to be  *$T$ -structured* if it is a Bayesian network on  $G$ . A distribution  $P$  is said to be *tree-structured* if it is  $T$ -structured for some tree  $T$ .

Now we define the notion of polytree-structured distributions, which generalizes tree-structured distributions defined before.

**Definition A.5** (Polytree-structured distribution). A directed acyclic graph (DAG)  $G$  is said to be a *polytree* if the skeleton of  $G$  is a forest. For a positive integer  $k \in \mathbb{N}$ ,  $G$  is said to be a  *$k$ -polytree* if the in-degree of each node in  $G$  is at most  $k$ . A distribution  $P$  is said to be a  *$(k)$ -polytree-structured* if it is a Bayes net defined over some  $(k)$ -polytree  $G$ .

Finally, we define the notion of Chordal-structured distributions, a class of Bayes nets that generalizes polytree-structured distributions and will be crucially used in this work.

**Definition A.6** (Chordal-structured distribution). An undirected graph  $G$  is said to be *chordal* if every cycle of length at least 4 has a chord, that is, an edge that connects two non-adjacent vertices on the cycle. A distribution  $P$  is said to be *chordal-structured* if it is a Bayes net defined over a DAG whose skeleton is chordal.

Clearly, any tree-structured distribution is polytree-structured, and any polytree-structured distribution is chordal-structured.

**Definition A.7** (Add-one distribution). For a directed acyclic graph  $G$  with vertex set  $[n]$  and a set of samples  $S = \{x^{(1)}, \dots, x^{(|S|)}\} \subseteq [k]^n$ , the *add-one* or *Laplace distribution* with  $G$ -structure given the samples  $S$ , denoted by  $P_{G,S}^+$ , is a Bayes net on  $G$  and is defined as follows

$$P_{G,S}^+(x) = \prod_{v \in [n]} P_{S,v,\text{pa}_G(v)}^+(x_v | x_{\text{pa}_G(v)})$$

<sup>1</sup>We use the notation  $X_S$  to denote  $\{X_i : i \in S\}$  for a set  $S \subseteq [n]$ .

where all the node distributions  $P_{S,v,\text{pa}(v)}^+$  are computed as

$$P_{S,v,\text{pa}(v)}^+(z_v | z_{\text{pa}(v)}) = \frac{\left| \left\{ x^{(i)} \in S : x_v^{(i)} = z_v \wedge x_w^{(i)} = z_w \forall w \in \text{pa}(v) \right\} \right| + 1}{\left| \left\{ x^{(i)} \in S : x_w^{(i)} = z_w \forall w \in \text{pa}(v) \right\} \right| + k} \quad \forall z \in [k]^n.$$

That is,  $P_{G,S}^+$  is the Bayes net on  $G$  where all the node distributions (of  $X_v \mid X_{\text{pa}(v)}$ ) are defined w.r.t the (conditional) add-one or Laplace estimates computed from the samples  $S = \{x^{(1)}, \dots, x^{(|S|)}\}$  for each fixing of the parents' values.

## A.2 PAC Distribution Learning

A *distribution learning* algorithm takes as input a sequence of i.i.d. samples generated from a probability distribution  $P$  and outputs a description  $\Theta$  of a distribution  $\hat{P}_\Theta$  as an estimate for  $P$ . In the following,  $\mathcal{C}$  is a family of probability distributions over  $[k]^n$ , and  $P$  is a distribution over  $[k]^n$  not necessarily in  $\mathcal{C}$ . Let us first define the notion of approximation of a distribution that will be used in this work.

**Definition A.8** ( $(\varepsilon, A)$ -approximation of a distribution). A distribution  $\hat{P}$  is said to be an  $(\varepsilon, A)$ -approximation for  $P$  with respect to  $\mathcal{C}$  if:

$$D_{\text{KL}}(P \parallel \hat{P}) \leq A \cdot \inf_{Q \in \mathcal{C}} D_{\text{KL}}(P \parallel Q) + \varepsilon.$$

When  $A = 1$ ,  $\hat{P}$  is said to be an  $\varepsilon$ -approximation for  $P$ . Note that when  $P \in \mathcal{C}$ , if  $\hat{P}$  is an  $\varepsilon$ -approximation for  $P$ , then  $d_{\text{TV}}(P, \hat{P}) \leq \sqrt{\frac{\varepsilon}{2}}$  using Pinsker's inequality (Lemma A.1).

Now we proceed to define the notion of PAC-learning with respect to KL divergence.

**Definition A.9** (PAC-learning in KL divergence). A distribution learning algorithm is said to<sup>2</sup> be an *agnostic PAC-learner* for  $\mathcal{C}$  with sample complexity  $m_{\mathcal{C}}(n, k, \varepsilon, \delta)$  and running time  $t_{\mathcal{C}}(n, k, \varepsilon, \delta)$ , if for all distributions  $P$  over  $[k]^n$  and all  $\varepsilon, \delta \in (0, 1)$ , given  $\varepsilon, \delta$ , and a sample set of size  $m = m_{\mathcal{C}}(n, k, \varepsilon, \delta)$  drawn i.i.d. from  $P$ , the algorithm runs for time  $t \leq t_{\mathcal{C}}(n, k, \varepsilon, \delta)$  and outputs the description  $\Theta$  of a distribution  $\hat{P}_\Theta$  such that with probability at least  $1 - \delta$ ,  $\hat{P}_\Theta$  is an  $\varepsilon$ -approximation for  $P$  with respect to  $\mathcal{C}$  (where the probability is taken over the samples as well as the algorithm's randomness).

If  $P$  is restricted to be in  $\mathcal{C}$ , the algorithm is said to be a *realizable PAC-learner* for  $\mathcal{C}$ . If the output  $\hat{P}_\Theta$  is guaranteed to be in  $\mathcal{C}$ , the algorithm is said to be a *proper PAC-learner* for  $\mathcal{C}$ ; otherwise the learner is called an *improper PAC-learner*.

## A.3 Online Learning

The framework of prediction with experts is setup as follows; the formulation we follow here is based on Chapter 2 of [6]. The goal is to design an algorithm  $\mathcal{A}$  that predicts an unknown sequence  $x^{(1)}, x^{(2)}, \dots$  of elements of an *outcome space*  $\mathcal{X}$ . The algorithm's predictions  $\hat{P}_1, \hat{P}_2, \dots$  belong to a *decision space*  $\mathcal{D}$  (which in our case will be the probability simplex on  $\mathcal{X}$ ). The algorithm  $\mathcal{A}$  makes its predictions sequentially, and the quality of its predictions is benchmarked against a set of reference predictions called *experts*. At each time instant  $t$ ,  $\mathcal{A}$  has access to the set of expert predictions  $\mathcal{E}$ , where  $\mathcal{E}$  is a fixed subset of  $\mathcal{D}$ .  $\mathcal{A}$  makes its own prediction  $\hat{P}_t$  based on the expert predictions. Finally, after the prediction  $\hat{P}_t$  is made, the true outcome  $x^{(t)}$  is revealed, and the algorithm  $\mathcal{A}$  incurs a loss  $\ell(\hat{P}_t, x^{(t)})$ , where  $\ell : \mathcal{D} \times \mathcal{X} \rightarrow \mathbb{R}$  is a non-negative *loss function*.

Note that  $x^{(1)}, x^{(2)}, \dots$  form an arbitrary sequence. Hence, naturally, the loss incurred by the algorithm depends on how well the experts fit these outcomes.

Now we are ready to define the notion of regret.

<sup>2</sup>In the learning theory literature, when  $A > 1$ , such a guarantee is also sometimes called *semi-agnostic learning*.

**Definition A.10** (Regret). Given the notation above, the *cumulative regret* (or simply *regret*) over  $T$  steps is defined as:

$$\text{Reg}_T(\mathcal{A}; \mathcal{E}) = \sum_{t=1}^T \ell(\hat{P}_t, x^{(t)}) - \min_{E \in \mathcal{E}} \sum_{t=1}^T \ell(E, x^{(t)})$$

Since  $\mathcal{E}$  is usually fixed, we often use the shorthand  $\text{Reg}_T(\mathcal{A})$ . The *average regret* is defined as  $\frac{1}{T} \text{Reg}_T(\mathcal{A})$ . The total  $T$  steps are also sometimes referred to as the *horizon* of the algorithm.

Our work utilizes two standard online learning algorithms: Exponentially Weighted Average (EWA) and Randomized Weighted Majority (RWM). Both algorithms assume that the expert set is finite; the first algorithm is deterministic, while the second one is probabilistic.

**Fact A.11.** In both EWA and RWM algorithms, when run with a hyperparameter  $\eta > 0$  and a set of experts  $\mathcal{E} = \{E_1, \dots, E_N\}$ , the *weight of the  $i$ -th expert* ( $i \in [N]$ ) *after time step  $t$*  ( $t \in [T]$ ) is denoted by  $w_{i,t}$  defined as follows:

$$w_{i,t} \triangleq \exp \left( -\eta \sum_{s=1}^t \ell(E_i, x^{(s)}) \right).$$

---

**Algorithm 1:** EWA forecaster

---

**Input :** Experts  $\mathcal{E} = \{E_1, \dots, E_N\}$ , parameter  $\eta$ , horizon  $T$ .

- 1  $w_{i,0} \leftarrow 1$  for each  $i \in [N]$ .
  - 2 **for**  $t \leftarrow 1$  **to**  $T$  **do**
  - 3      $\hat{P}_t \leftarrow \frac{\sum_{i=1}^N w_{i,t-1} E_i}{\sum_{j=1}^N w_{j,t-1}}.$
  - 4     **Output**  $\hat{P}_t$ .
  - 5     Observe outcome  $x^{(t)}$ .
  - 6     **for**  $i \in [N]$  **do**
  - 7          $w_{i,t} \leftarrow w_{i,t-1} \cdot \exp(-\eta \cdot \ell(E_i, x^{(t)})).$
- 

---

**Algorithm 2:** RWM forecaster

---

**Input :** Experts  $\mathcal{E} = \{E_1, \dots, E_N\}$ , parameter  $\eta$ , horizon  $T$ .

- 1  $w_{i,0} \leftarrow 1$  for each  $i \in [N]$ .
  - 2 **for**  $t \leftarrow 1$  **to**  $T$  **do**
  - 3     Sample  $\hat{P}_t \in \mathcal{E}$  where  $\Pr[\hat{P}_t = E_i] = \frac{w_{i,t-1}}{\sum_{j=1}^N w_{j,t-1}}.$
  - 4     **Output**  $\hat{P}_t$ .
  - 5     Observe outcome  $x^{(t)}$ .
  - 6     **for**  $i \in [N]$  **do**
  - 7          $w_{i,t} \leftarrow w_{i,t-1} \cdot \exp(-\eta \cdot \ell(E_i, x^{(t)})).$
- 

Before proceeding to give the regret bounds for EWA and RWM algorithms, we need the notion of exp-concave loss functions.

**Definition A.12** (Exp-concave loss function). Let  $\mathcal{D}$  be a convex decision space,  $\mathcal{X}$  be a sample space, and let  $\alpha \in \mathbb{R}$  be a parameter. A loss function  $\ell : \mathcal{D} \times \mathcal{X} \rightarrow \mathbb{R}$  is said to be  $\alpha$ -*exp-concave* if the function  $f_x : \mathcal{D} \rightarrow \mathbb{R}$ ,  $f_x(P) = \exp(-\alpha \ell(P, x))$  is concave for all  $x \in \mathcal{X}$ .

The following regret bound holds for EWA with *exp-concave* loss functions. Importantly for us,  $\ell(P, x) = -\log(P(x))$  is exp-concave for  $0 \leq \alpha \leq 1$ .

**Lemma A.13** ([6], Theorem 3.2). *With  $N = |\mathcal{E}|$  experts and an  $\alpha$ -exp-concave loss function  $\ell(P, x)$ , for any sequence of outcomes  $x^{(1)}, \dots, x^{(T)}$ , the regret of the EWA forecaster with the parameter  $\eta = \alpha$  is bounded as follows:*

$$\text{Reg}_T(\text{EWA}; \mathcal{E}) \leq \frac{\log N}{\eta}.$$

The regret bound below for RWM assumes the loss function is bounded.

**Lemma A.14** ([6], Lemma 4.2). *With  $N = |\mathcal{E}|$  experts and a loss function bounded in  $[-L_{\max}, L_{\max}]$ , for any sequence of outcomes  $x^{(1)}, \dots, x^{(T)}$ , the expected regret of the RWM forecaster with the parameter  $\eta = \sqrt{8(\log N)/T}$  is bounded as follows:*

$$\mathbb{E}[\text{Reg}_T(\text{RWM}; \mathcal{E})] \leq L_{\max} \sqrt{\frac{T \log N}{2}}.$$

Moreover, with probability at least  $1 - \delta$ , the following holds:

$$\text{Reg}_T(\text{RWM}; \mathcal{E}) \leq L_{\max} \left( \sqrt{\frac{T \log N}{2}} + \sqrt{\frac{T}{2} \log \frac{1}{\delta}} \right).$$

## B Agnostic Learning of Bayesian Networks

In this section, we study the notions and our results on the agnostic learning of Bayesian networks in general. Our first result in Section B.1 shows that we can leverage the multiplicative weight update method to get an agnostic learning guarantee in reverse KL distance in terms of the regret of the RWM or EWA algorithm. Thereafter, we can plug in the standard regret bounds for EWA and RWM to get an agnostic learning guarantee for our output distributions with high probability. These results are given in Lemma B.4 and B.6 respectively. Next we apply our general learning result to the problem of learning Bayesian networks which requires us to define an appropriate set of experts. Towards this, we define one expert for each possible DAGs on  $n$  nodes of in-degree at most  $d$ . The distribution of such an expert will be fixed to the add-1 distribution at each conditional distribution defined using a previous set of samples. A prior work by Bhattacharyya et al. ([2]) has established that this add-1 estimator will have a near-optimal sample complexity of agnostically learning fixed-structure Bayesian networks in KL divergence. Combining everything together, we get interesting new sample-complexity upper bounds of agnostically learning Bayes nets in both improper and proper learning setting using EWA and RWM algorithms respectively in Theorems B.11 and B.12.

**Theorem B.1.** *Let  $\mathcal{C}$  be the family of distributions over  $[k]^n$  that can be defined as Bayes nets over DAGs with  $n$  nodes and maximum in-degree  $d$ . There exists a PAC-learner for  $\mathcal{C}$  in the realizable setting using  $\tilde{O}(nk^{d+1}\epsilon^{-1})$  samples that returns a mixture of Bayes nets from  $\mathcal{C}$  which is an  $\epsilon$ -approximation of the input distribution with probability at least  $2/3$ .*

The sample complexity in Theorem B.1 is *nearly optimal*<sup>3</sup> for constant error probability. The same approach also yields polynomial sample complexity for agnostic learning in the non-realizable setting with exponentially small error probability and for proper learning with polynomially small error probability. While these bounds are no longer optimal, we will use them later to design the first polynomial time algorithms for learning Bayes nets over large classes of DAGs.

We now describe the relation between regret and KL divergence, which is crucially used throughout this work.

### B.1 Connection between Regret and KL divergence

In the prediction with experts framework, let us fix the decision space  $\mathcal{D}$  to be the probability simplex on the outcome space  $\mathcal{X}$  and the loss function to be  $\ell(P, x) = -\log P(x)$  for the distribution  $P$  defined over the probability simplex on  $\mathcal{X}$ . Then we have the following general connection between the expected average regret and the expected KL divergence when the outcomes  $x^{(1)}, x^{(2)}, \dots$  are i.i.d samples from a distribution  $P^*$ .

**Lemma B.2.** *If  $\mathcal{A}$  is a (possibly randomized) online learning algorithm that receives i.i.d samples  $x^{(1)}, \dots, x^{(T)}$  from a probability distribution  $P^*$  on  $\mathcal{X}$  and makes predictions  $\hat{P}_1, \dots, \hat{P}_T \in \mathcal{D} \subseteq \Delta(\mathcal{X})$  with respect to an expert set  $\mathcal{C} \subseteq \mathcal{D}$ ,*

$$\frac{\mathbb{E}_{x^{(1)}, \dots, x^{(T)}} \mathbb{E}_{t \sim \text{Unif}([T])} [\text{D}_{\text{KL}}(P^* \| \hat{P}_t)]}{\mathbb{E}_{x^{(1)}, \dots, x^{(T)}}} \leq \frac{1}{T} \mathbb{E}_{x^{(1)}, \dots, x^{(T)}} [\text{Reg}_T(\mathcal{A}; \mathcal{C})] + \min_{P \in \mathcal{C}} \text{D}_{\text{KL}}(P^* \| P).$$

<sup>3</sup>Note that for learning with respect to total variation distance, the corresponding sample complexity of  $\tilde{O}(nk^{d+1}\epsilon^{-2})$  is known ([4]), but the proof uses the tournament method which does not apply to KL divergence as it is not a metric.

Note that if  $\mathcal{A}$  is a randomized algorithm, both the left and right hand side of the inequality in Lemma B.2 are random variables.

*Proof.* From the definition of regret (Definition A.10), we have:

$$\text{Reg}_T(\mathcal{A}; \mathcal{C}) \triangleq \sum_{t=1}^T \log \frac{1}{\widehat{P}_t(x^{(t)})} - \min_{P \in \mathcal{C}} \sum_{t=1}^T \log \frac{1}{P(x^{(t)})}.$$

Thus, the expectation over the samples of the average regret for  $T$  rounds would be the following:

$$\begin{aligned} \frac{1}{T} \mathbb{E}_{x^{(1)}, \dots, x^{(T)} \sim P^*} [\text{Reg}_T(\mathcal{A}; \mathcal{C})] &= \mathbb{E}_{x^{(1)}, \dots, x^{(T)}} \frac{1}{T} \cdot \left( \sum_{t=1}^T \log \frac{1}{\widehat{P}_t(x^{(t)})} - \min_{P \in \mathcal{C}} \sum_{t=1}^T \log \frac{1}{P(x^{(t)})} \right) \\ &\geq \frac{1}{T} \mathbb{E}_{x^{(1)}, \dots, x^{(T)}} \left[ \sum_{t=1}^T \log \frac{1}{\widehat{P}_t(x^{(t)})} \right] - \min_{P \in \mathcal{C}} \mathbb{E}_{x \sim P^*} \left[ \log \frac{1}{P(x)} \right] \\ &= \frac{1}{T} \mathbb{E}_{x^{(1)}, \dots, x^{(T)}} \left[ \sum_{t=1}^T \log \frac{1}{\widehat{P}_t(x^{(t)})} \right] - \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) - H(P^*) \end{aligned} \quad (2)$$

where the second-last step is by applying Jensen's inequality, as  $\min$  is a concave function, and also uses the fact that  $x^{(1)}, \dots, x^{(T)} \sim P^*$ . Also,  $H(P^*)$  denotes the entropy of the distribution  $P^*$ . Thus we have the following:

$$\begin{aligned} \frac{1}{T} \mathbb{E}_{x^{(1)}, \dots, x^{(T)}} [\text{Reg}_T(\mathcal{A}; \mathcal{C})] &\geq \frac{1}{T} \mathbb{E}_{x^{(1)}, \dots, x^{(T)}} \left[ \sum_{t=1}^T \log \frac{1}{\widehat{P}_t(x^{(t)})} \right] - \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) - H(P^*) \\ &= \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{x^{(1)}, \dots, x^{(t-1)}} \mathbb{E}_{x \sim P^*} \left[ \log \frac{1}{\widehat{P}_t(x)} \middle| x^{(1)}, \dots, x^{(t-1)} \right] - H(P^*) \\ &\quad - \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) \\ &= \frac{1}{T} \sum_t \mathbb{E}_{x^{(1)}, \dots, x^{(t-1)}} \left[ D_{\text{KL}}(P^* \| \widehat{P}_t) \right] - \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) \\ &= \mathbb{E}_{x^{(1)}, \dots, x^{(T-1)}} \left[ \frac{1}{T} \sum_{t=1}^T D_{\text{KL}}(P^* \| \widehat{P}_t) \right] - \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) \end{aligned}$$

In the second line, inside the summation over  $t$ , the expectation does not include  $x^{(t)}$  because the prediction  $\widehat{P}_t$  does not depend upon  $x^{(t)}$ , as  $\widehat{P}_t$  is predicted before  $x^{(t)}$  is revealed.  $\square$

We immediately have the following corollary:

**Corollary B.3.** *In the same setup as Lemma B.2 above, if  $\frac{1}{T} \text{Reg}_T(\mathcal{A}; \mathcal{C}) \leq \rho$ ,*

$$\mathbb{E}_{x^{(1)}, \dots, x^{(T)}} \left[ D_{\text{KL}} \left( P^* \| \frac{1}{T} \sum_{t=1}^T \widehat{P}_t \right) \right] \leq \rho + \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P).$$

*Proof.* This follows from applying the fact that KL-divergence is convex in its second argument.  $\square$

Now, we apply these results to the EWA and RWM algorithms. Let us start with the results for the EWA algorithm.

**Lemma B.4.** *In the same setup as Lemma B.2 above, suppose that for every  $P \in \mathcal{C}$ ,  $\min_{x \in \mathcal{X}} P(x) \geq \tau$  holds.*

- *If  $\mathcal{A}$  is the EWA algorithm run with parameter  $\eta = 1$ , then we have the following:*

$$\mathbb{E}_{x^{(1)}, \dots, x^{(T)}} \left[ D_{\text{KL}} \left( P^* \| \frac{1}{T} \sum_{t=1}^T \widehat{P}_t \right) \right] \leq \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) + \frac{\log |\mathcal{C}|}{T}.$$

- For any  $\varepsilon, \delta \in (0, 1)$ , let  $\mathcal{A}$  be the EWA algorithm run with parameter  $\eta = \frac{\varepsilon}{2\sqrt{T} \log(1/\tau) \sqrt{\log(1/\delta)}}$ . Then with probability at least  $1 - \delta$ , the following holds:

$$D_{\text{KL}} \left( P^* \parallel \frac{1}{T} \sum_{t=1}^T \hat{P}_t \right) \leq \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \parallel P) + \frac{2 \log |\mathcal{C}| \log(1/\tau) \sqrt{\log(1/\delta)}}{\varepsilon \sqrt{T}} + \varepsilon.$$

*Proof.* Recall that our loss function  $\ell(P, x) = -\log P(x)$  is  $\eta$ -exp-concave for all  $0 \leq \eta \leq 1$ . Hence, we can apply Lemma A.13 for any such  $\eta$ . The first item follows from Corollary B.3 by setting  $\eta = 1$ .

For the second item, we use the method of bounded differences. Define the function  $f$ :

$$f(X^{(1)}, \dots, X^{(T)}) = D_{\text{KL}} \left( P^* \parallel \frac{1}{T} \sum_{t=1}^T \hat{P}_t \right)$$

**Claim B.5.** For any  $X^{(1)}, X^{(2)}, \dots, X^{(T)}, X^{(1)'} \in \mathcal{X}$ ,

$$|f(X^{(1)}, X^{(2)}, \dots, X^{(T)}) - f(X^{(1)'}, X^{(2)}, \dots, X^{(T)})| \leq 2\eta \log \frac{1}{\tau}.$$

*Proof.* Let  $\hat{P}_t$  and  $\hat{P}_t'$  be the outputs from the EWA algorithm after  $t$  rounds using the samples  $X^{(1)}, X^{(2)}, \dots, X^{(T)}$  and  $X^{(1)'}, X^{(2)}, \dots, X^{(T)}$  respectively. Then,  $\hat{P}_t = \sum_{Q \in \mathcal{C}} w_{t,Q} Q$  and  $\hat{P}_t' = \sum_{Q \in \mathcal{C}} w'_{t,Q} Q$ .

By definition of the EWA algorithm and the choice of our loss function,  $w_{t,Q} = \frac{(\prod_{s \leq t} Q(X^{(s)}))^\eta}{\sum_{R \in \mathcal{C}} (\prod_{s \leq t} R(X^{(s)}))^\eta}$  and  $w'_{t,Q} = \frac{(Q(X^{(1)')}) \prod_{2 \leq s \leq t} Q(X^{(s)})^\eta}{\sum_{R \in \mathcal{C}} R(X^{(1)'}) (\prod_{2 \leq s \leq t} R(X^{(s)}))^\eta}$ . Since for every  $Q \in \mathcal{C}$ ,  $\tau \leq \frac{Q(X^{(1)})}{Q(X^{(1)'})} \leq \frac{1}{\tau}$ ,  $\tau^{2\eta} \leq \frac{w_{t,Q}}{w'_{t,Q}} \leq \tau^{-2\eta}$  and hence for every  $y \in \mathcal{X}$ ,  $\tau^{2\eta} \leq \frac{\hat{P}_t'(y)}{\hat{P}_t(y)} \leq \tau^{-2\eta}$ . Therefore:

$$\left| f(X^{(1)'}, \dots, X^{(T)}) - f(X^{(1)}, \dots, X^{(T)}) \right| = \left| \sum_{y \in \mathcal{X}} P^*(y) \log \left( \frac{\sum_{t=1}^T \hat{P}_t'(y)}{\sum_{t=1}^T \hat{P}_t(y)} \right) \right| \leq 2\eta \log \frac{1}{\tau}.$$

□

Applying the McDiarmid's inequality,

$$\Pr \left[ \left| D_{\text{KL}}(P^* \parallel \hat{P}) - \mathbb{E} \left[ D_{\text{KL}}(P^* \parallel \hat{P}) \right] \right| \geq \varepsilon \right] \leq 2 \exp \left( -\frac{2\varepsilon^2}{4T\eta^2 \log^2 \tau^{-1}} \right) \leq \delta,$$

by our choice of  $\eta$ . The second item then follows from Lemma A.13, Corollary B.3, and the above. □

Now let us discuss the guarantees of the RWM algorithm.

**Lemma B.6.** In the same setup as Lemma B.2 above, suppose that  $\mathcal{C} = \{P_1, \dots, P_N\}$  and that for every  $P_i \in \mathcal{C}$ ,  $\min_{x \in \mathcal{X}} P_i(x) \geq \tau$  holds.

- If  $\mathcal{A}$  is the RWM algorithm run with the parameter  $\eta = \sqrt{(8 \log N)/T}$  — giving proper predictions  $\hat{P}_1, \dots, \hat{P}_T \in \mathcal{C}$  — and  $\hat{P} \leftarrow \hat{P}_t \in \mathcal{C}$  where  $t$  is sampled uniformly from  $[T]$ , then we have:

$$\mathbb{E}_{\mathcal{A}, x^{(1)}, \dots, x^{(T)}} \left[ D_{\text{KL}}(P^* \parallel \hat{P}) \right] \leq \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \parallel P) + \frac{\log(1/\tau) \sqrt{T \log |\mathcal{C}|}}{T}.$$

- With the same algorithm  $\mathcal{A}$  as above, for any  $\delta \in (0, 1)$ , the following bound holds with probability  $\geq 1 - \delta$  over the randomness of **RWM**, the random choice of  $t$  from  $[T]$ , and the randomness of the samples  $x^{(1)}, \dots, x^{(T)}$ :

$$D_{\text{KL}}(P^* \|\hat{P}) \leq \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) + \frac{\log(1/\tau) \left( \sqrt{\log |\mathcal{C}|} + \sqrt{\log(2/\delta)} \right)}{\delta \sqrt{T}}.$$

*Proof.* The first item follows from Lemma B.2, taking the expectation of both sides over the randomness of  $\mathcal{A}$ , and the expected regret bound in Lemma A.14. Note that we can take  $L_{\max} = \log(1/\tau)$  since  $P(x) \geq \tau$  for all  $x \in \mathcal{X}$  and  $P \in \mathcal{C}$ , by the assumption of the lemma.

The RWM algorithm (see Algorithm 2) can equivalently be viewed as follows. Sample  $U_1, \dots, U_T \sim \text{Uniform}((0, 1])$  independently. At each  $t \in [T]$ , predict  $\hat{P}_t \leftarrow P_{i_t} \in \mathcal{C}$ , where  $i_t \in [N]$  is the index such that  $U_t \in \left( \frac{\sum_{j=1}^{i_t-1} w_{j,t-1}}{\sum_{k=1}^N w_{k,t-1}}, \frac{\sum_{j=1}^{i_t} w_{j,t-1}}{\sum_{k=1}^N w_{k,t-1}} \right]$ . Note that modulo the choice of  $U = (U_1, \dots, U_T)$ , the RWM algorithm is deterministic.

From Lemma A.14 (high probability bound), with probability  $\geq 1 - \delta/2$  over the random choice of  $U$ ,  $\text{Regret}_T(\mathcal{A}; \mathcal{C}) \leq \log(1/\tau) \left( \sqrt{\frac{T \log |\mathcal{C}|}{2}} + \sqrt{\frac{T}{2} \log \frac{2}{\delta}} \right)$ . Conditioning on this event  $\mathcal{E}$ , applying Lemma B.2, we get

$$\begin{aligned} \mathbb{E}_{x^{(1)}, \dots, x^{(T)}} \mathbb{E}_{t \sim \text{Unif}([T])} \left[ \mathcal{Z}_{X,t} \triangleq D_{\text{KL}}(P^* \|\hat{P}_t) - \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) \mid \mathcal{E} \right] \\ \leq \frac{\log(1/\tau) \left( \sqrt{\log |\mathcal{C}|} + \sqrt{\log(2/\delta)} \right)}{2\sqrt{T}} \end{aligned}$$

Note that, since  $\hat{P}_t \in \mathcal{C}$  for all  $t \in [T]$ ,  $\mathcal{Z}_{X,t} \geq 0$  for all  $x^{(1)}, \dots, x^{(T)} \in \mathcal{X}$  and  $t \in [T]$ . Thus, we can use Markov's inequality to show that, conditioned on the event  $\mathcal{E}$ , with probability  $\geq 1 - \delta/2$  over the randomness of  $x^{(1)}, \dots, x^{(T)}$  and the random choice of  $t \sim \text{Unif}([T])$ ,

$$D_{\text{KL}}(P^* \|\hat{P}) - \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) \leq \frac{\log(1/\tau) \left( \sqrt{\log |\mathcal{C}|} + \sqrt{\log(2/\delta)} \right)}{\delta \sqrt{T}}. \quad (3)$$

Taking a union bound with event  $\bar{\mathcal{E}}$  and the failure event of the above bound (3), we can see that (3) holds with probability  $\geq 1 - \delta$  over the randomness of algorithm  $\mathcal{A}$ , the random choice of  $t$ , and the randomness of the samples  $x^{(1)}, \dots, x^{(T)}$ . This completes the proof of the lemma.  $\square$

## B.2 Discretization

We will typically want to design agnostic PAC-learners for classes of distributions  $\mathcal{C}$  that are infinite, e.g., tree-structured distributions, etc. However, to apply the results of the previous section, we first need to *finitize*  $\mathcal{C}$ . We make use of the properties of the add-one distribution — also explored in [2] — to do this finitization more efficiently compared to using a  $\varepsilon$ -cover over each set of node distributions (the cover method [15, 9], applied to Bayes nets). This of course means that we make use of additional samples from the ground truth distribution  $P^*$  to construct the discretization. However, we show that this can be done without affecting our asymptotic sample complexity (up to logarithmic factors).

Suppose we are interested in the class  $\mathcal{C}$  of  $\mathcal{G}$ -structured Bayes nets, where  $\mathcal{G} = \{G_1, \dots, G_M\}$  is a *finite set* of directed acyclic graphs (DAGs), all with indegree  $\leq d$  —  $\mathcal{G}$  may be the set of all rooted trees on  $[n]$  (with  $d = 1$ ), all DAGs having a particular (undirected) skeleton, all DAGs on  $[n]$  with indegree  $\leq d$ , etc. We define the discretization  $\mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}}$  to be the set of add-one distributions constructed from a set of  $s_{\text{AO}}(\varepsilon, \delta) \leq \tilde{O} \left( \frac{d^3 n k^{d+1} \log^2(nk/\varepsilon\delta)}{\varepsilon} \right)$  samples from  $P^*$  (see Definition A.7 and Theorem B.8) for each DAG  $G \in \mathcal{G}$ . Note that all the distributions in  $\mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}}$  are constructed using a single set of samples from  $P^*$ .



**Definition B.7.** Suppose  $\mathcal{G} = \{G_1, \dots, G_M\}$  is a set of DAGs on  $n$  nodes with maximum indegree  $d$ , and  $P^*$  is a distribution on  $[k]^n$ . For  $\varepsilon > 0$  and  $\delta \in (0, 1)$ , the finitization  $\mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}}$  is constructed as  $\mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}} \triangleq \{\hat{P}_{G_i, S}^+ : i \in [M]\}$  (see Theorem B.8), where  $S$  is a set of  $s_{\text{AO}}(\varepsilon, \delta) \triangleq \Theta\left(\frac{nk^{d+1}}{\varepsilon} \log\left(\frac{n^{d+1}k^{d+1}}{\delta}\right) \log\left(\frac{nk^{d+1}}{\varepsilon} \log\left(\frac{1}{\delta}\right)\right)\right)$  i.i.d samples from  $P^*$ .

For determining the number of samples  $s_{\text{AO}}(\varepsilon, \delta)$  used to construct the finitization, we use the following refined variant of Theorem 6.2, in [2] (where the guarantee holds uniformly for all DAGs on  $[n]$  with indegree  $\leq d$ ).

**Theorem B.8.** Let  $\mathcal{G}$  denote the set of all DAGs on  $[n]$  with maximum indegree  $\leq d$ , and  $P^*$  is a distribution on  $[k]^n$ . For any  $G \in \mathcal{G}$ , let  $\hat{P}_{G, S}^+$  denote the add-one distribution (see Definition A.7) with  $G$ -structure constructed given a set  $S$  of i.i.d samples from  $P^*$ . Then, if  $|S| \geq s_{\text{AO}}(\varepsilon, \delta) \triangleq \Theta\left(\frac{nk^{d+1}}{\varepsilon} \log\left(\frac{n^{d+1}k^{d+1}}{\delta}\right) \log\left(\frac{nk^{d+1}}{\varepsilon} \log\left(\frac{1}{\delta}\right)\right)\right)$ , the distribution  $\hat{P}_{G, S}^+$  satisfies

$$D_{\text{KL}}(P^* \| \hat{P}_{G, S}^+) \leq \min_{G\text{-structured distributions } Q} D_{\text{KL}}(P^* \| Q) + \varepsilon$$

for every  $G \in \mathcal{G}$ , with probability  $\geq 1 - \delta$  over the samples  $S$ .

*Proof.* As in the proof of Theorem 6.2 in [2], for any DAG  $G \in \mathcal{G}$ , we can write  $D_{\text{KL}}(P^* \| \hat{P}_{G, S}^+) - D_{\text{KL}}(P^* \| P_G^*)$  as:

$$\sum_{v \in [n]} \sum_{x \in [k]^{\text{pa}_G(v)}} P^*(X_{\text{pa}_G(v)} = x) D_{\text{KL}}\left(P^*(X_v | X_{\text{pa}_G(v)} = x) \| \hat{P}_{G, S}^+(X_v | X_{\text{pa}_G(v)} = x)\right), \quad (4)$$

where  $P_G^*$  is a distribution that minimizes  $D_{\text{KL}}(P^* \| P)$  among all  $G$ -structured distributions  $P$ .

Note that this decomposition depends only on the set of parents of each node  $v \in [n]$ . Thus to upper bound the LHS for all DAGs in  $\mathcal{G}$ , we just need to upper bound  $\sum_{x \in [k]^U} P^*(X_U = x) D_{\text{KL}}\left(P^*(X_v | X_U = x) \| \hat{P}_{G, S}^+(X_v | X_U = x)\right)$  for all  $v$  and for all subsets  $U \subseteq [n]$  with  $|U| \leq d$ .

The number of possible subsets is  $\binom{n+d}{d} \leq \left(\frac{2en}{d}\right)^d \leq O((2n)^d)$  since  $d \leq n - 1$ . Not all such configurations will give valid DAGs in  $\mathcal{G}$ , but a bound for all  $v, U$  is sufficient to upper bound the LHS for every  $G \in \mathcal{G}$ .

Now we can proceed as in the proof of Theorem 1.4 of [2]. Let  $N = |S|$  (total number of samples from  $P^*$ ) and  $r$  be the number of samples from  $P^*(X_v | X_U = x)$  for a particular  $v \in [n]$ ,  $U \subseteq [n]$  and  $x \in [k]^U$ . Using the guarantee for the add-one estimator at each node (Theorem 6.1, [2]) and splitting into two cases ( $\mathbb{E}[r] > 15 \log(1/\delta)$  and  $\mathbb{E}[r] \leq 15 \log(1/\delta)$ ) as in the proof of (Theorem 1.4, [2]) gives us that

$$P^*(X_U = x) D_{\text{KL}}\left(P^*(X_v | X_U = x) \| \hat{P}_{G, S}^+(X_v | X_U = x)\right) \leq O\left(\frac{k \log\left(\frac{k}{\delta}\right) \log N}{N}\right)$$

with probability  $\geq 1 - \delta$  over the  $N$  samples.

Rescaling  $\delta$  to  $\delta' = \frac{\delta}{n(2kn)^d}$  and using a union bound gives the above inequality for all choices of  $v \in [n]$ ,  $U \subseteq [n]$  with  $|U| \leq d$  and  $x \in [k]^U$  together, with probability  $\geq 1 - n(2kn)^d \delta' = 1 - \delta$  over the samples.

Using this upper bound for all terms in equation (4) along with the fact that  $|\text{pa}_G(v)| \leq d$  for all  $G \in \mathcal{G}$  gives (w.p  $\geq 1 - \delta$ )

$$\forall G \in \mathcal{G} : D_{\text{KL}}(P^* \| \hat{P}_{G, S}^+) - D_{\text{KL}}(P^* \| P_G^*) \leq O\left(\frac{nk^{d+1} \log\left(\frac{nk(2kn)^d}{\delta}\right) \log N}{N}\right).$$

Choosing  $N = \Theta\left(\frac{nk^{d+1}}{\varepsilon} \log\left(\frac{n^{d+1}k^{d+1}}{\delta}\right) \log\left(\frac{nk^{d+1}}{\varepsilon} \log\left(\frac{1}{\delta}\right)\right)\right)$  proves the theorem.  $\square$

This finitization of the class of  $\mathcal{G}$ -structured Bayes nets has the following useful properties.

**Lemma B.9.** Any  $P \in \mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}}$  (where DAGs in  $\mathcal{G}$  have indegree  $\leq d$ ) satisfies  $\min_{x \in [k]^n} P(x) \geq \left( \frac{\varepsilon}{C_d n k^{d+1} \log^2 \left( \frac{nk}{\varepsilon \delta} \right)} \right)^n$ , for some constant  $C_d > 0$  that depends on  $d$ .

*Proof.* Each distribution  $P \in \mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}}$  is an add-one distribution (see Definition A.7) constructed from  $s_{\text{AO}}(\varepsilon, \delta)$  samples of  $P^*$ . Each such distribution is a Bayes net on an  $n$ -vertex DAG and hence the joint probability is a product of the  $n$  node (conditional) probabilities. Since each node distribution  $X_v | X_{\text{pa}(v)}$  is an add-one estimate, the probability value (for each  $X_v = x_v | X_{\text{pa}(v)} = x_{\text{pa}(v)}$ ) is at least  $\frac{1}{s_{x_{\text{pa}(v)}+k}} \geq \frac{1}{s_{\text{AO}}(\varepsilon, \delta) + k}$ , where  $s_{x_{\text{pa}(v)}}$  is the number of samples consistent with  $X_{\text{pa}(v)} = x_{\text{pa}(v)}$ . Hence,

$$\begin{aligned} P(x) &\geq \prod_{v \in [n]} \frac{1}{s_{x_{\text{pa}(v)}} + k} \geq \left( \frac{1}{s_{\text{AO}}(\varepsilon, \delta) + k} \right)^n \stackrel{(1)}{\geq} \Theta \left( \frac{\varepsilon}{d^3 n k^{d+1} \log \left( \frac{nk}{\delta} \right) \log \left( \frac{nk}{\varepsilon} \log \frac{1}{\delta} \right)} \right)^n \\ &\geq \left( \frac{\varepsilon}{C_d n k^{d+1} \log^2 \left( \frac{nk}{\varepsilon \delta} \right)} \right)^n, \end{aligned}$$

where the inequality (1) follows from  $s_{\text{AO}}(\varepsilon, \delta) = \Theta \left( \frac{nk^{d+1}}{\varepsilon} \log \left( \frac{n^{d+1} k^{d+1}}{\delta} \right) \log \left( \frac{nk^{d+1}}{\varepsilon} \log \frac{1}{\delta} \right) \right)$  (see Definition B.7).  $\square$

**Lemma B.10.** Let  $\mathcal{C}^{\mathcal{G}} \subseteq \Delta([k]^n)$  be the set of all Bayes nets defined on any DAG  $G \in \mathcal{G}$ , where  $\mathcal{G}$  is a finite set of DAGs on  $[n]$ , and all DAGs in  $\mathcal{G}$  have maximum indegree  $d$ . For any  $P^* \in \Delta([k]^n)$ , the following holds with probability  $\geq 1 - \delta$  over the samples used in the construction of  $\mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}}$ :

$$\min_{P \in \mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}}} D_{\text{KL}}(P^* \| P) \leq \min_{P \in \mathcal{C}^{\mathcal{G}}} D_{\text{KL}}(P^* \| P) + \varepsilon.$$

*Proof.* Take  $\mathcal{G} = \{G_1, \dots, G_M\}$  and  $\mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}} = \{\hat{P}_{G_1, S}^+, \dots, \hat{P}_{G_M, S}^+\}$  for a set of samples  $S$  with  $|S| \geq s_{\text{AO}}(\varepsilon, \delta)$ . Applying Theorem B.8, we have that for all DAGs  $G_i \in \mathcal{G}$  (where  $i$  is not dependent on the samples  $S$ ), with probability  $\geq 1 - \delta$ ,

$$D_{\text{KL}}(P^* \| \hat{P}_{G_i, S}^+) \leq \min_{G_i\text{-structured distributions } Q} D_{\text{KL}}(P^* \| Q) + \varepsilon. \quad (5)$$

Let  $P'$  denote a distribution in  $\mathcal{C}^{\mathcal{G}}$  that minimizes  $D_{\text{KL}}(P^* \| P')$ . By definition of  $\mathcal{C}^{\mathcal{G}}$ ,  $P'$  will be  $G_{i^*}$ -structured for some  $i^* \in [M]$  which is fixed as a function of  $P^*$  (and independent of the samples  $S$ ), which implies that,

$$\min_{P \in \mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}}} D_{\text{KL}}(P^* \| P) \leq D_{\text{KL}}(P^* \| \hat{P}_{G_{i^*}, S}^+) \stackrel{(1)}{\leq} D_{\text{KL}}(P^* \| P') + \varepsilon \stackrel{(2)}{=} \min_{P \in \mathcal{C}^{\mathcal{G}}} D_{\text{KL}}(P^* \| P) + \varepsilon.$$

Note that the equality (2) holds by definition of  $P'$  as the KL minimizer, and the penultimate inequality (1) holds since  $P'$  is  $G_{i^*}$ -structured, applying (5) with  $i = i^*$ .  $\square$

### B.3 Sample Complexity for learning Bayes Nets

We can now collect the tools developed in Sections B.1 and B.2 to prove Theorem B.1 from the introduction. The algorithm that yields the first part of Theorem B.1 (improper learning) is quite simple to describe (Algorithm 3 below), corresponding to EWA forecaster with  $\eta = 1$ . In the following theorem, we specify the  $\mathcal{N}$  and  $T$  to take so as to obtain desired guarantees.

**Theorem B.11** (Bayes net improper learning). *Let  $\mathcal{C}$  be the class of distributions over  $[k]^n$  that can be defined as Bayes nets over any unknown DAG with  $n$  nodes and in-degree  $\leq d$ .*

- *There is an algorithm that for all  $P^* \in \Delta([k]^n)$  and all  $\varepsilon \in (0, 1)$ , receives  $m = O \left( \frac{d^3 n k^{d+1}}{\varepsilon} \log^2 \frac{nk}{\varepsilon} \right)$  i.i.d. samples from  $P^*$  and returns a distribution  $\hat{P}$  that is with probability at least  $2/3$ , an  $(\varepsilon, 4)$ -approximation for  $P^*$  with respect to  $\mathcal{C}$ .*

---

**Algorithm 3:** EWA-based learning for Bayes nets

---

**Input** :  $\mathcal{N} = \{P_1, \dots, P_N\}, T$ , hyperparameter  $\eta > 0$ .  
**Output** : Sampler for  $\hat{P}$ .

```
1  $w_{i,0} \leftarrow 1$  for each  $i \in [N]$ .
2 for  $t \leftarrow 1$  to  $T$  do
3   Observe sample  $x^{(t)} \sim P^*$ .
4   for  $i \in [N]$  do
5      $w_{i,t} \leftarrow w_{i,t-1} \cdot P_i(x^{(t)})^\eta$ .
6 function EWA-SAMPLER()
7   Sample  $t \leftarrow [T]$  uniformly at random.
8   Sample  $i \sim [N]$  with probability  $\frac{w_{i,t-1}}{\sum_{j \in [N]} w_{j,t-1}}$ .
9   return  $x \sim P_i$ .
10 return EWA-SAMPLER /* This is a sampler for  $\hat{P}$ . */
```

---

---

**Algorithm 4:** RWM-based learning for Bayes nets

---

**Input** :  $\mathcal{N} = \{P_1, \dots, P_N\}, T$ , hyperparameter  $\eta > 0$ .  
**Output** :  $\hat{P} \in \mathcal{N}$ .

```
1  $w_{i,0} \leftarrow 1$  for each  $i \in [N]$ .
2 for  $t \leftarrow 1$  to  $T$  do
3   Sample  $i_t$  from  $[N]$  with  $\Pr(i_t = i) = \frac{w_{i,t-1}}{\sum_{j \in [N]} w_{j,t-1}}$ .
4   Observe sample  $x^{(t)} \sim P^*$ .
5   for  $i \in [N]$  do
6      $w_{i,t} \leftarrow w_{i,t-1} \cdot P_i(x^{(t)})^\eta$ .
7 Sample  $t$  uniformly from  $[T]$ .
8 return  $\hat{P} \leftarrow P_{i_t}$ .
```

---

• There is a realizable PAC-learner for  $\mathcal{C}$  with sample complexity  $O\left(\frac{d^3 nk^{d+1}}{\varepsilon \delta} \log^2 \frac{nk}{\varepsilon \delta}\right)$ .

• There is an agnostic PAC-learner for  $\mathcal{C}$  with sample complexity

$$\tilde{O}\left(\max\left\{\frac{d^4 n^4 \log^2(n)}{\varepsilon^4} \log^2\left(\frac{nk d \log^2(nk/\varepsilon \delta)}{\varepsilon}\right) \log(1/\delta), \frac{d^3 nk^{d+1}}{\varepsilon} \log\left(\frac{nk}{\delta}\right) \log\left(\frac{nk}{\varepsilon} \log \frac{1}{\delta}\right)\right\}\right).$$

*Proof.* Let  $\mathcal{G}$  be the collection of all DAGs on  $n$  nodes that have in-degree  $d$ , then we can say that  $|\mathcal{G}| \leq (n!)^{d+1}$ . Suppose  $\mathcal{N} = \mathcal{N}_{\varepsilon/8, \delta/2}^{\mathcal{G}}$ , using the notation of Definition B.7. By construction, we have

$$\log(|\mathcal{N}|) \leq \log(|\mathcal{G}|) \leq (d+1) \log n! \leq (d+1)n \log(n), \quad (6)$$

and the sample complexity of the learning algorithm will be  $s_{\text{AO}}\left(\frac{\varepsilon}{8}, \frac{\delta}{2}\right)$  (to construct  $\mathcal{N}$ ) +  $T$  (the number of online rounds), where

$$s_{\text{AO}}(\varepsilon/8, \delta/2) \leq O\left(\frac{d^3 nk^{d+1}}{\varepsilon} \log\left(\frac{nk}{\delta}\right) \log\left(\frac{nk}{\varepsilon} \log \frac{1}{\delta}\right)\right) \text{ (using Definition B.7 and Theorem B.8).} \quad (7)$$

By Lemma B.10 and the definition of  $\mathcal{N}$  above,

$$\min_{P \in \mathcal{N}} D_{\text{KL}}(P^* \| P) \leq \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) + \varepsilon/8, \quad (8)$$

with probability  $\geq 1 - \delta/2$  over the samples used to construct  $\mathcal{N}$ .

We begin with the first item. Here, we can take  $\delta = \frac{1}{8}$  for the purpose of constructing  $\mathcal{N}$ . We run EWA-based learning (Algorithm 3) using  $\mathcal{N} = \mathcal{N}_{\varepsilon/8, 1/16}^{\mathcal{G}}$  as the expert set,  $\eta = 1$ , and loss function  $\ell(P, x) = -\log P(x)$ . By the first part of Lemma B.4 and Markov's inequality, with probability at

least 3/4:

$$D_{\text{KL}} \left( P^* \parallel \frac{1}{T} \sum_{t=1}^T \hat{P}_t \right) \leq 4 \left( \min_{P \in \mathcal{N}} D_{\text{KL}}(P^* \parallel P) + \frac{\log |\mathcal{N}|}{T} \right). \quad (9)$$

Using (6), and combining (8) and (9) with a union bound — choosing  $T = \frac{8(d+1)n \log(n)}{\varepsilon} \geq 8 \frac{\log(|\mathcal{N}|)}{\varepsilon}$  — now ensures that right-hand-side in (9) is at most  $4 \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \parallel P) + \varepsilon$  with probability at least  $1 - (\frac{1}{4} + \frac{1}{16}) \geq \frac{2}{3}$ . The sample complexity will be dominated by  $s_{\text{AO}}(\frac{\varepsilon}{8}, \frac{1}{16}) \leq O\left(\frac{d^3 n k^{d+1}}{\varepsilon} \log^2\left(\frac{nk}{\varepsilon}\right)\right)$  (see Definition B.7).

For the second item, let  $\mathcal{N}' = \mathcal{N}_{\varepsilon\delta/4, \delta/2}^{\mathcal{G}}$  as the expert set for the algorithm. By Lemma B.10, if  $P^* \in \mathcal{C}$ ,  $\min_{P \in \mathcal{N}'} D_{\text{KL}}(P^* \parallel P) \leq \varepsilon\delta/4$  with probability  $\geq 1 - \delta/2$ . We can now run the same argument as above with  $\varepsilon$  being  $\varepsilon\delta/4$  (applying Markov's inequality with error probability  $\delta/2$ ) to obtain the result.

We now prove the third item. Suppose  $\varepsilon, \delta \in (0, 1)$ . By Lemma B.9 and  $|\mathcal{G}| \leq (n!)^{d+1} \leq n^{nd}$ , for every  $P \in \mathcal{N}$ ,  $\min_{x \in [k]^n} P(x) \geq \Theta\left(\frac{\varepsilon}{d^2 n k^{d+1} \log\left(\frac{nk}{\delta}\right) \log\left(\frac{nk}{\varepsilon} \log \frac{1}{\delta}\right)}\right) = \tau$ . Thus,  $\log(1/\tau) \leq \Theta\left(n \log\left(\frac{d^2 n k^{d+1} \log^2\left(\frac{nk}{\varepsilon\delta}\right)}{\varepsilon}\right)\right)$ . We run EWA using  $\mathcal{N}$  as the expert set,  $\eta = \frac{\varepsilon}{4\sqrt{T} \log(1/\tau) \sqrt{\log(1/\delta)}} = \Theta\left(\frac{\varepsilon}{4\sqrt{T} n \log\left(\frac{d^2 n k^{d+1} \log^2\left(\frac{nk}{\varepsilon\delta}\right)}{\varepsilon}\right) \sqrt{\log(2/\delta)}}\right)$  and the same loss function.

By the second part of Lemma B.4, we get that with probability at least  $1 - \delta/2$ :

$$D_{\text{KL}} \left( P^* \parallel \frac{1}{T} \sum_{t=1}^T \hat{P}_t \right) \leq \min_{P \in \mathcal{N}} D_{\text{KL}}(P^* \parallel P) + \Theta \left( \frac{4dn^2 \log(n) \log\left(\frac{d^2 n k^{d+1} \log^2\left(\frac{nk}{\varepsilon\delta}\right)}{\varepsilon}\right) \sqrt{\log(2/\delta)}}{\varepsilon \sqrt{T}} \right) + \frac{\varepsilon}{2} \quad (10)$$

Using (6) and (8) along with a union bound, if  $T = \Theta\left(d^4 n^4 \log^2(n) \varepsilon^{-4} \log^2\left(\frac{nk d \log^2(nk/\varepsilon\delta)}{\varepsilon}\right) \log(1/\delta)\right)$ , then the right-hand-side of (10) is at most  $\min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \parallel P) + \varepsilon$  and the bound will hold with probability  $\geq 1 - \delta$ . Here, the sample complexity will be

$$T + s_{\text{AO}}(\varepsilon/8, \delta/2) \leq O\left(\frac{d^4 n^4 \log^2(n)}{\varepsilon^4} \log^2\left(\frac{nk d \log^2(nk/\varepsilon\delta)}{\varepsilon}\right) \log(1/\delta) + \frac{d^3 n k^{d+1}}{\varepsilon} \log\left(\frac{nk}{\delta}\right) \log\left(\frac{nk}{\varepsilon} \log \frac{1}{\delta}\right)\right).$$

□

The above theorem is interesting in the context of improper learning of Bayes nets with constant in-degree  $d$ , because it gives a nearly-optimal sample complexity (optimal up to logarithmic factors) for realizable PAC learning with constant success probability (second part) and for getting a  $(\varepsilon, 3)$ -approximation with constant success probability (first part).

While the next theorem does not have significant advantages for general in-degree  $d$  Bayesian networks due to the less efficient nature of Algorithm 4 and its higher sample complexity compared to established algorithms such as Chow-Liu for proper learning, we prove it to lay the groundwork for subclasses of Bayesian networks (such as trees, polytrees, and chordal graphs). For these subclasses, the sampling method employed in Algorithm 4 can be optimized to yield efficient algorithms for proper learning.

**Theorem B.12** (Bayes net proper learning). *Let  $\mathcal{C}$  be the class of distributions over  $[k]^n$  that can be defined as Bayes nets over any unknown DAG with  $n$  nodes and in-degree  $\leq d$ . There is a proper agnostic*

$(\varepsilon, \delta)$ -PAC-learner for  $\mathcal{C}$  with sample complexity  $O\left(\max\left\{\frac{d^3 n^3 \log^3\left(\frac{nk}{\varepsilon\delta}\right)}{\varepsilon^2 \delta^2}, \frac{d^3 nk^{d+1}}{\varepsilon} \log^2\left(\frac{nk}{\varepsilon\delta}\right)\right\}\right)$  for any  $\delta \in (0, 1)$  and  $\varepsilon > 0$ .

*Proof.* Define  $\mathcal{N} = \mathcal{N}_{\varepsilon/2, \delta/2}^{\mathcal{G}}$  — where  $\mathcal{G}$  is the collection of all DAGs on  $n$  nodes with in-degree  $\leq d$  — the same way as in the proof of Theorem B.11. We run the RWM-based learning algorithm (Algorithm 4) with  $\mathcal{N}$  as the expert set,  $\eta = \sqrt{(8 \log |\mathcal{N}|)/T}$ , and the log loss function. By the second part of Lemma B.6, and noting that:

- (i) The maximum loss at any round is at most (using Lemma B.9; see also the third item in the proof of Theorem B.11)

$$L_{\max} = \log(1/\tau) \leq O\left(n \log\left(\frac{d^3 nk^{d+1} \log^2\left(\frac{nk}{\varepsilon\delta}\right)}{\varepsilon}\right)\right).$$

- (ii) the size of the cover  $\mathcal{N}$  satisfies  $\log |\mathcal{N}| \leq \log |\mathcal{G}| \leq (d+1)n \log(n)$ .

we have that, for any  $\delta > 0$ , with probability  $\geq 1 - \delta/2$  over the random samples as well as the randomness of the algorithm, we have:

$$D_{\text{KL}}(P^* \|\hat{P}) \leq \underbrace{\min_{P \in \mathcal{N}} D_{\text{KL}}(P^* \| P) + C \left( \frac{\left( n \log\left(\frac{d^3 nk^{d+1} \log^2\left(\frac{nk}{\varepsilon\delta}\right)}{\varepsilon}\right) \right) (\sqrt{(d+1)n \log(n)} + \sqrt{\log(4/\delta)})}{\delta \sqrt{T}} \right)}_{\triangleq \text{ErrorBound}},$$

where  $C$  is some constant  $> 1$ .

Using the above inequality with a union bound, along with Lemma B.10, the definition of  $\mathcal{C}$ , and the fact that  $\mathcal{N} = \mathcal{N}_{\varepsilon/2, \delta/2}^{\mathcal{G}}$ , we have that

$$D_{\text{KL}}(P^* \|\hat{P}) \leq \min_{P \in \mathcal{C}} D_{\text{KL}}(P^* \| P) + \text{ErrorBound} + \varepsilon/2,$$

with probability  $\geq 1 - \delta$  over the samples and the randomness of RWM.

Now if we choose any  $T \geq \Theta\left(\frac{d^2 n^2 \log^2\left(\frac{nk d \log^2(nk/\varepsilon\delta)}{\varepsilon}\right) (dn \log(n) + \log(4/\delta))}{\varepsilon^2 \delta^2}\right)$ , we get

$D_{\text{KL}}(P^* \|\hat{P}) \leq \varepsilon/2 + \varepsilon/2 = \varepsilon$  with probability  $\geq 1 - \delta$  (this bound for  $T$  uses the fact that  $(\sqrt{x} + \sqrt{y})^2 \leq 2(x + y)$  by the AM-GM inequality). The total sample complexity of the algorithm will be  $T + s_{\text{AO}}(\varepsilon/2, \delta/2)$  (see Theorem B.8 and Lemma B.10). This gives us the

$$O\left(\frac{d^3 n^3 \log^3\left(\frac{nk d}{\varepsilon\delta}\right)}{\varepsilon^2 \delta^2} + \frac{d^3 nk^{d+1}}{\varepsilon} \log^2\left(\frac{nk}{\varepsilon\delta}\right)\right)$$

sample complexity for proper agnostic  $(\varepsilon, \delta)$ -PAC learning.  $\square$

## C Learning Chordal-structured distributions

In this section, we show that *chordal-structured distributions* can be learnt efficiently. Before proceeding to describe our results, we first define some notions that will be used in our results and proofs.

### C.1 Preliminaries about Chordal Graphs

Given an undirected graph  $G = (V, E)$  and subsets  $S, T \subseteq V$ , we let  $G[S]$  denote the induced subgraph of  $G$  on  $S$ ,  $E(S)$  denote the edge set of  $G[S]$ , and  $E(S, T)$  denote the set of edges with one

endpoint in  $S$  and the other in  $T$ . For a vertex  $v$  of  $G$ , let  $\text{Nbr}_G(v)$  denote the set of adjacent vertices of  $v$  in  $G$  (vertices  $u$  such that  $\{u, v\} \in E(G)$ ).

An undirected graph is *chordal* if every cycle of length at least 4 contains a chord, that is an edge connecting two vertices of the cycle which is not part of the cycle.

**Definition C.1** (Clique tree). Let  $G = (V, E)$  be a graph with vertex set  $V$  and edge set  $E$ . The *clique tree*, denoted by  $\mathcal{T}^G$ , of  $G$  is a tree that has the maximal cliques of  $G$  as its vertices and for every two maximal cliques  $C$  and  $C'$ , each clique on the path from  $C$  to  $C'$  in  $\mathcal{T}^G$  contains  $C \cap C'$ .  $\mathcal{T}^G$  has the *induced subtree* property: for every vertex  $v \in V$ , the set of nodes in  $\mathcal{T}^G$  that contains  $v$  forms a connected subtree of  $\mathcal{T}^G$ . The *treewidth* of  $G$  equals one less than the size of the largest maximal clique in  $G$ .

Let  $G$  be a chordal graph, and let  $\mathcal{T}^G$  be a clique tree of  $G$ . Fix an arbitrary maximal clique  $C_r \in V(\mathcal{T}^G)$  and we view  $\mathcal{T}^G$  as a tree rooted at  $C_r$ . We denote this *rooted clique tree* as  $\mathcal{T}_{C_r}^G$  and as  $\mathcal{T}_{C_r}$  if  $G$  is clear from context. For any maximal clique  $C$ , let  $\text{pa}_{\mathcal{T}_{C_r}}(C)$  denote the parent of  $C$  in the rooted tree  $\mathcal{T}_{C_r}$ , and let  $\mathcal{T}_C$  denote the subtree of  $\mathcal{T}_{C_r}$  rooted at  $C$ . Let  $V[\mathcal{T}_C]$  denote the vertex set of  $G[\mathcal{T}_C]$ , that is,  $\bigcup_{C' \in V(\mathcal{T}_C)} C'$ . For notational convenience, we will use  $\mathcal{T}_C$  to denote both the subtree of  $\mathcal{T}^G$  as well the vertex set  $V[\mathcal{T}_C]$  when the usage will be clear from the context. Thus,  $G[\mathcal{T}_C]$  denotes the subgraph of  $G$  induced by the vertices in  $V[\mathcal{T}_C]$ .

**Definition C.2** (Separator set). Let  $G$  be a chordal graph, and let  $\mathcal{T}_{C_r}$  be a rooted clique tree of  $G$  for a maximal clique  $C_r$ . For  $C \in V(\mathcal{T}_{C_r})$ , the *separator* of  $C$  with respect to  $\mathcal{T}_{C_r}$  is defined as follows:

$$\text{Sep}(C) = C \cap \text{Pa}_{\mathcal{T}_{C_r}}(C)$$

Next we define the notion of link set, which is crucially used in our proofs.

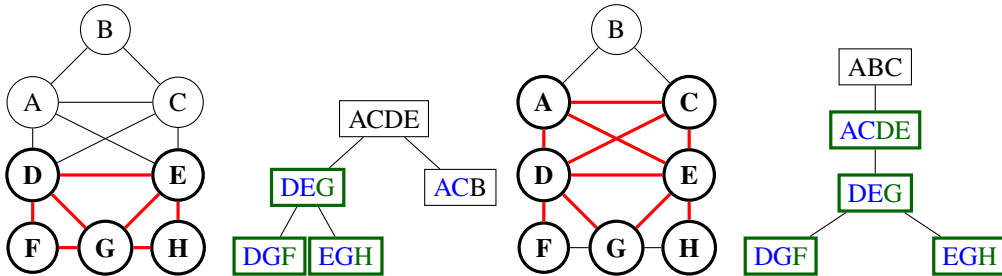
**Definition C.3** (Link set). Let  $G = (V, E)$  be a chordal graph, and  $\mathcal{T}_{C_r}$  be a rooted clique tree of  $G$  for a maximal clique  $C_r$ . For  $C \in V(\mathcal{T}_{C_r})$ , the *link set* of  $C$  is defined as follows:

$$\text{Link}(C) = E(C, V[\mathcal{T}_C])$$

An *orientation* of an edge set  $F$  assigns a direction to each edge in  $F$ ; an orientation is *acyclic* if it does not give rise to a directed cycle. The *indegree* of an orientation is the maximum number of incident edges which are oriented inward at any vertex. Two acyclic orientations on edge sets  $F$  and  $F'$  are said to be *consistent* with each other if they agree on the edges in  $F \cap F'$ .

**Definition C.4** (Indegree-Bounded Acyclic Orientations). Let  $G = (V, E)$  be an undirected graph. For any subset of edges  $F \subseteq E$  and integer  $d \geq 0$ ,  $\text{AO}_d(F)$  denotes the set of all acyclic orientations of  $F$  with indegree at most  $d$ . Given an orientation  $\mathcal{O}$  that assigns orientations to a subset of  $E$ , we let  $\text{AO}_d(F; \mathcal{O})$  denote the subset of  $\text{AO}_d(F)$  that is consistent with  $\mathcal{O}$ . If  $F$  corresponds to the edge set of a subgraph  $H$ , then we also use the notations  $\text{AO}_d(H)$  and  $\text{AO}_d(H, \mathcal{O})$  respectively.

Figure 1: In the left panel, a chordal graph and a clique tree decomposition with reference clique  $C = \text{DEG}$ . In the left panel, the edges of  $\text{Link}(C)$  are in red and the vertices  $V[\mathcal{T}_C]$  are in bold. In the right panel, the nodes of  $\mathcal{T}_C$  are in green, and the separator vertices in each node of the clique tree are colored in blue. The same chordal graph and a different clique tree decomposition with reference clique  $C = \text{ACDE}$ . The colors have the same meaning as in the left.



We will use below some standard observations about chordal graphs and acyclic orientations; for the sake of completeness, we give their proofs.

**Lemma C.5** (Lemma 1 of [1], restated). *Let  $G = (V, E)$  be a chordal graph and consider a clique tree  $\mathcal{T}_{C_r}$  of  $G$  rooted at a node  $C_r$ . Let  $C$  be a node in  $\mathcal{T}_{C_r}$  and  $C_1, \dots, C_\ell$  be its children in  $\mathcal{T}_{C_r}$ . Then the edge sets of the graphs  $G[\mathcal{T}_{C_1} \setminus \text{Sep}(C_1)], \dots, G[\mathcal{T}_{C_\ell} \setminus \text{Sep}(C_\ell)]$  are mutually disjoint.*

*Proof.* We will prove this by contradiction. Let us assume that there exists  $i \neq j$  with  $i, j \in [\ell]$  such that  $G[\mathcal{T}_{C_i} \setminus \text{Sep}(C_i)]$  and  $G[\mathcal{T}_{C_j} \setminus \text{Sep}(C_j)]$  share an edge  $e = \{u, v\}$ . This implies that both  $G[\mathcal{T}_{C_i}]$  and  $G[\mathcal{T}_{C_j}]$  contain the edge  $e$ . However, this would imply  $C \supseteq \{u, v\}$  by the *inducted subtree property* (since any pair of nodes in  $\mathcal{T}_{C_i}$  and  $\mathcal{T}_{C_j}$  can only be connected in  $\mathcal{T}^G$  through  $C$ ), and thus that  $e$  will be present in both  $E(\text{Sep}(C_i))$  and  $E(\text{Sep}(C_j))$ . This contradicts our assumption on  $e$ .  $\square$

**Lemma C.6** (See Theorem 11 in Chapter 4 of [13]). *Let  $G$  be a chordal graph which is acyclically orientable with indegree  $\leq d$ , and consider a rooted clique tree  $\mathcal{T}_{C_r}$  of  $G$ . Consider a non-leaf node  $C \in \mathcal{T}_{C_r}$  and  $C_1, \dots, C_\ell$  be the set of children of  $C$  in the rooted tree. Consider an acyclic orientation  $\mathcal{O}_C$  of  $\text{Link}(C)$  with indegree  $\leq d$ , and let  $\mathcal{O}_{\text{Sep}(C_i)}$  be the orientation  $\mathcal{O}_C$  restricted to  $E(\text{Sep}(C_i), \mathcal{T}_{C_i})$  for every  $i \in [\ell]$ . Then there exists a bijection between  $\text{AO}_d(\mathcal{T}_C, \mathcal{O}_C)$  and  $\text{AO}_d(\mathcal{T}_{C_1}, \mathcal{O}_{\text{Sep}(C_1)}) \times \dots \times \text{AO}_d(\mathcal{T}_{C_\ell}, \mathcal{O}_{\text{Sep}(C_\ell)})$ .*

*Proof.* Consider any acyclic orientation  $\mathcal{O}$  of  $\mathcal{T}_C$  with indegree  $\leq d$  consistent with a given acyclic orientation  $\mathcal{O}_C$  of  $\text{Link}(C)$ . Any subset of an acyclic orientation  $\mathcal{O}$  will be acyclic as well, and indegree bounds will be preserved. Hence  $\mathcal{O}$  restricted to each  $\mathcal{T}_{C_i}$  will give acyclic orientations  $\mathcal{O}_i$  of  $\mathcal{T}_{C_i}$  with indegree  $\leq d$ . Given these consistent orientations  $\mathcal{O}_1, \dots, \mathcal{O}_\ell$  of the subtrees, and knowing that  $\mathcal{O}$  is consistent with  $\mathcal{O}_C$ , we can reconstruct  $\mathcal{O}$ . This suffices to argue the injectivity.

In order to prove surjectivity, let us consider acyclic orientations  $\mathcal{O}_i \in \text{AO}_d(\mathcal{T}_{C_i}, \mathcal{O}_{\text{Sep}(C_i)})$  for every  $i \in [\ell]$ . Now let us consider the following orientation  $\mathcal{O}$  by taking the union of  $\mathcal{O}_C$  and  $\bigcup_{i=1}^\ell \mathcal{O}_i$ . Following Lemma C.5, we know that  $\mathcal{O}$  is consistent with  $\mathcal{O}_C$ . Now we would like to prove that  $\mathcal{O}$  is an acyclic orientation and has indegree  $\leq d$ .

We will prove acyclicity by contradiction. Assume that there is at least one cycle in  $\mathcal{O}$ . Consider the shortest cycle among all those cycles in  $\mathcal{O}$ . The cycle must contain at least two edges in  $E(C)$ , because if it contains one or no edges of  $E(C)$ , then it is contained inside some  $\mathcal{T}(C_i)$  which is impossible since  $\mathcal{O}_i$  is acyclic. Consider two vertices  $u, v \in C$  which are in the cycle but not adjacent in the cycle. In  $C$ , there exists an edge between  $u$  and  $v$ , with  $\mathcal{O}_C$  orienting it either  $(u, v)$  or  $(v, u)$ . Whichever the case, we can shorten the cycle by taking a shortcut on the edge, which contradicts the assumption that we are considering the shortest cycle. This implies that  $\mathcal{O}$  is acyclic.

To argue that the indegree of  $\mathcal{O}$  is  $\leq d$ , note that any  $v$  which is common to  $\mathcal{T}_{C_i}$  and  $\mathcal{T}_{C_j}$  for  $i \neq j$  will belong to  $C$  as well. Hence the orientation of all incident edges of  $v$  will be fixed by  $\mathcal{O}_C$  itself, and combining  $\mathcal{O}_1, \dots, \mathcal{O}_\ell$  will not increase the indegree beyond  $d$  since they are all consistent with  $\mathcal{O}_C$ . Thus the mapping is surjective as well. Combining the above, we have the proof of the lemma.  $\square$

**Lemma C.7.** *Let  $C$  and  $C_i$  be two cliques of  $G$  so that  $C$  is a node in  $\mathcal{T}^G$  and  $C_i$  is a child of  $C$  in  $\mathcal{T}^G$ . Then there are no edges between  $V[\mathcal{T}_{C_i}] \setminus C$  and  $C \setminus C_i$ .*

*Proof.* Assume for the sake of contradiction that  $\{u, v\}$  is an edge between  $u \in V[\mathcal{T}_{C_i}] \setminus C$  and  $v \in C \setminus C_i$ . In particular, say  $u \in K$  where  $K$  is a clique in  $\mathcal{T}_{C_i}$ . Then, by the definition of the clique tree decomposition, (i) the edge is contained neither in  $C$  nor in any of the cliques in the subtree rooted at  $C_i$  (since  $u \notin C$  and  $v \in C \setminus C_i$ ), but (ii) the edge is part of some maximal-clique  $C'$ . Hence,  $C'$  and  $K$  must be separated by  $C$  in the clique tree. But this is a contradiction, since  $u \in C'$  and  $u \in K$  but  $u \notin C$ .  $\square$

## C.2 EWA and RWM with Chordal Experts

In this section, we consider distributions on chordal graphs which can be oriented acyclically with indegree  $\leq d$ . This assumption is sufficient to bound the size of each maximal clique.

**Remark C.8.** Let  $G$  be an undirected chordal graph. If there exists an acyclic orientation of  $G$  with indegree  $\leq d$ , then for any clique tree decomposition  $\mathcal{T}_{C_r}$  of  $G$  (rooted at a maximal clique  $C_r$ ), all the nodes of  $\mathcal{T}_{C_r}$  (maximal cliques of  $G$ ) have cardinality  $\leq d + 1$ .

*Proof.* By assumption, we have an acyclic orientation with indegree  $\leq d$  for each clique  $C \in V(\mathcal{T}_{C_r})$ . If we topologically order the nodes of  $C$ , the last node will have indegree  $|C| - 1 \leq d$ , which implies  $|C| \leq d + 1$ .  $\square$

We will establish the following theorem.

**Theorem C.9.** *Let  $G$  be an undirected chordal graph, and suppose  $k$  and  $d$  are fixed constants. Let  $\mathcal{L}_d^G$  be the family of distributions over  $[k]^n$  that can be defined as Bayes nets over DAGs having skeleton  $G$  and with indegree  $\leq d$ . Given sample access from  $P^* \in \Delta([k]^n)$ , and a parameter  $\varepsilon > 0$ , there exist the following:*

- (i) *An agnostic PAC-learner for  $\mathcal{L}_d^G$  using  $\tilde{O}\left(\max\left\{\frac{d^4 n^4}{\varepsilon^4} \log^4\left(\frac{nk d}{\varepsilon}\right) \log^2(1/\delta), \frac{d^3 n k^{d+1}}{\varepsilon} \log^2\left(\frac{nk}{\varepsilon \delta}\right)\right\}\right)$  samples that is improper and returns an efficiently-samplable mixture of distributions from  $\mathcal{L}_d^G$ .*
- (ii) *An agnostic PAC-learner `LearnChordalDist` for  $\mathcal{L}_d^G$  using  $\tilde{O}\left(\max\left\{\frac{d^3 n^3}{\varepsilon^2 \delta^2}, \frac{d^3 n k^{d+1}}{\varepsilon} \log^2\left(\frac{nk}{\varepsilon \delta}\right)\right\}\right)$  samples and  $\text{poly}(n)$  running time that is proper and returns a distribution from  $\mathcal{L}_d^G$ .*

The sample complexity guarantees follow from Theorem B.11 and from Theorem B.12. What remains to be justified is that the algorithms run efficiently and the distribution output by the improper learning algorithm can be sampled efficiently. Below, let  $x^{(1)}, \dots, x^{(T)}$  be the samples drawn from  $P^*$  for the online phase, and let  $S_{\mathcal{N}}$  denote the set of  $s_{\text{AO}}(\varepsilon, \delta)$  samples — for appropriate choices of  $\varepsilon, \delta$  as used in Theorem B.11 and Theorem B.12 — used to construct the finite expert set  $\mathcal{N}$  used in the algorithms..

Let  $\mathcal{T}_{C_r}^G$  be a clique tree decomposition of  $G$  rooted at a maximal clique  $C_r$ . We can assume without loss of generality that  $G$  has at least one acyclic orientation with indegree  $\leq d$ ; otherwise,  $\mathcal{L}_d^G$  is empty and the problem is trivial. Hence, by Remark C.8, every node in  $\mathcal{T}_{C_r}^G$  has at most  $d + 1$  vertices of  $G$ .

**Claim C.10.** *For any  $C \in V(\mathcal{T}_{C_r})$ ,  $|\text{AO}_d(\text{Link}(C))| \leq \binom{n+d}{d}^{d+1}$ .*

*Proof.* Note that there are at most  $d + 1$  vertices in  $C$  and each such vertex has at most  $n$  incident edges, of which at most  $d$  can be incoming. Combining the above, we have the claim.  $\square$

For an orientation  $\mathcal{O}$  of  $G[\mathcal{T}_C]$  and a node  $v \in V[\mathcal{T}_C]$ , let  $\text{in}_C(v, \mathcal{O})$  denote the set of in-neighbors of  $v$  in  $G[\mathcal{T}_C]$  with respect to  $\mathcal{O}$ . We define the *weight*, with respect to the clique  $C$ , of a node  $v \in V[\mathcal{T}_C]$ , acyclic orientation  $\mathcal{O}$  of  $G[\mathcal{T}_C]$  and time-step  $t \in [T]$  as follows:

$$\text{wt}_C(v, \mathcal{O}, t) \triangleq \prod_{s=1}^t \exp_{\eta} \log P_{S_{\mathcal{N}}, v, \text{in}_C(v, \mathcal{O})}^+ \left( x_v^{(s)} \mid x_{\text{in}_C(v, \mathcal{O})}^{(s)} \right). \quad (11)$$

where  $P_{S_{\mathcal{N}}, v, \text{in}_C(v, \mathcal{O})}^+$  is the add-one distribution for the node  $v$  with nodes in  $\text{in}_C(v, \mathcal{O})$  chosen as the parents of  $v$  (see Definition A.7), computed from the first set of samples  $S_{\mathcal{N}}$  used to construct  $\mathcal{N}$ .

At any point of our sampling algorithm, we only compute the add-one distribution for a node  $v$  with respect to a fixed set of  $\leq d$  parents for the node  $v$ , which are fixed (and adjusted) recursively. Note that this distribution can be computed by performing a single pass over the  $\text{poly}(n, k, \varepsilon, \log(1/\delta))$  many samples (for constant  $d$ ).

For all  $C \in V(\mathcal{T}^G)$  and for all  $\mathcal{O}_C \in \text{AO}_d(\text{Link}(C))$ , and for all  $t \in [T]$ , we will store an entry in a table  $\text{Table}[C, \mathcal{O}_C, t]$ .

$$\text{Table}[C, \mathcal{O}_C, t] \triangleq \sum_{\mathcal{O} \in \text{AO}_d(G[\mathcal{T}_C], \mathcal{O}_C)} \prod_{v \in V[\mathcal{T}_C]} \text{wt}_C(v, \mathcal{O}, t). \quad (12)$$

Note that, according to this definition,  $\text{Table}[C_r, \emptyset, t]$  gives the total weight of all indegree  $\leq d$  acyclic orientations of  $G$  after observing  $t$  samples. The sum is over an exponential-sized set; nevertheless, we will be able to use dynamic programming to compute it efficiently.



Let us start with the case that  $C$  is a leaf node of  $\mathcal{T}^G$ . Note that, if  $C$  is a leaf node of  $\mathcal{T}^G$ ,  $\text{AO}_d(G[\mathcal{T}_C], \mathcal{O}_C) = \{\mathcal{O}_C\}$ , where  $\mathcal{O}_C$  is an acyclic orientation of  $C$ . So, the corresponding table entry can be directly computed for any  $\mathcal{O}_C$  in  $\text{poly}(n, t, k, 1/\varepsilon)$  time for constant  $d$ . This completes the base case of our dynamic programming.

Now let us assume that  $C$  has  $\ell$  children  $C_1, \dots, C_\ell$  in  $\mathcal{T}^G$ . For  $\mathcal{O}_C \in \text{AO}_d(\text{Link}(C))$ , we show that  $\text{Table}[C, \mathcal{O}_C, t]$  can be inductively computed in terms of the table entries  $\text{Table}[C_i, \mathcal{O}_{C_i}, t]$  for  $1 \leq i \leq \ell$ , where each  $\mathcal{O}_{C_i} \in \text{AO}_d(\text{Link}(C_i))$ , so that we obtain a bottom-up dynamic programming algorithm for counting weighted acyclic orientations with indegree  $\leq d$  starting from the leaf nodes of  $\mathcal{T}^G$ . By Lemma C.6, we first break each acyclic orientation  $\mathcal{O}$  of  $\mathcal{T}_C$  into  $\mathcal{O}_1, \dots, \mathcal{O}_\ell$  (acyclic orientations of the child subtrees  $\mathcal{T}_{C_i}$ ) which can be combined consistently with an orientation  $\mathcal{O}_C \in \text{AO}_d(\text{Link}(C))$ .

The table entry  $\text{Table}[C, \mathcal{O}_C, t]$  can be expressed as follows:

$$\begin{aligned} \text{Table}[C, \mathcal{O}_C, t] &\triangleq \sum_{\mathcal{O} \in \text{AO}_d(G[\mathcal{T}_C], \mathcal{O}_C)} \prod_{v \in V[\mathcal{T}_C]} \text{wt}_C(v, \mathcal{O}, t) \\ &= \sum_{\mathcal{O}_1 \in \text{AO}_d(G[\mathcal{T}_{C_1}], \mathcal{O}_C), v \in V[\mathcal{T}_C]} \prod_{j=1}^{\ell} \text{wt}_C(v, \cup_{j=1}^{\ell} \mathcal{O}_j, t) \\ &\quad \vdots \\ &\quad \mathcal{O}_\ell \in \text{AO}_d(G[\mathcal{T}_{C_\ell}], \mathcal{O}_C) \\ &= \prod_{v \in C} \text{wt}_C(v, \mathcal{O}_C, t) \sum_{\substack{\mathcal{O}_1 \in \text{AO}_d(G[\mathcal{T}_{C_1}], \mathcal{O}_C), \\ \vdots \\ \mathcal{O}_\ell \in \text{AO}_d(G[\mathcal{T}_{C_\ell}], \mathcal{O}_C)}} \left( \prod_{i=1}^{\ell} \prod_{v \in V[\mathcal{T}_{C_i}] \setminus C} \text{wt}_{C_i}(v, \mathcal{O}_i, t) \right) \quad (13) \end{aligned}$$

where in the last line, we used the fact that  $V[\mathcal{T}_{C_i}] \setminus C$  are disjoint for distinct  $i$ , and also that, there are no edges between  $V[\mathcal{T}_{C_i}] \setminus C$  and  $C \setminus C_i$  by Lemma C.7.

For each  $i$ , let  $\text{Cons}_i(\mathcal{O}_C) \subseteq \text{AO}_d(\text{Link}(C_i))$  be the set of acyclic orientations of  $\text{Link}(C_i)$  of indegree at most  $d$  that are consistent with  $\mathcal{O}_C$ . From (13), we can write  $\text{Table}[C, \mathcal{O}_C, t]$  as:

$$\begin{aligned} &\prod_{v \in C \setminus (C_1 \cup \dots \cup C_\ell)} \text{wt}_C(v, \mathcal{O}_C, t) \times \\ &\sum_{\forall i, \mathcal{O}_{C_i} \in \text{Cons}_i(\mathcal{O}_C)} \sum_{\forall i, \mathcal{O}_i \in \text{AO}_d(G[\mathcal{T}_{C_i}], \mathcal{O}_{C_i})} \left( \left( \prod_{i=1}^{\ell} \prod_{v \in V[\mathcal{T}_{C_i}] \setminus \text{Sep}(C_i)} \text{wt}_{C_i}(v, \mathcal{O}_i, t) \right) \right. \quad (14) \\ &\quad \left. \cdot \left( \prod_{v \in \cup_{i=1}^{\ell} \text{Sep}(C_i)} \text{wt}_C(v, \mathcal{O}_C, t) \right) \right). \end{aligned}$$

We can now state a recurrence for  $\text{Table}[C, \mathcal{O}_C, t]$  using the notation above.

**Lemma C.11.** *For any  $C \in V(\mathcal{T}_{C_r})$ ,  $\mathcal{O}_C \in \text{AO}_d(\text{Link}(C))$ ,  $t \in [T]$ , if  $C_1, \dots, C_\ell$  are the children of  $C$  in  $\mathcal{T}_{C_r}$ , then:*

$$\text{Table}[C, \mathcal{O}_C, t] = \prod_{v \in C \setminus \cup_i C_i} \text{wt}_C(v, \mathcal{O}_C, t) \sum_{\substack{\forall i, \mathcal{O}_{C_i} \\ \in \text{Cons}_i(\mathcal{O}_C)}} \left( \prod_{i=1}^{\ell} \text{Table}[C_i, \mathcal{O}_{C_i}, t] \right) \Xi(\mathcal{O}_C, \mathcal{O}_{C_1}, \dots, \mathcal{O}_{C_\ell}),$$

where:

$$\Xi(\mathcal{O}_C, \mathcal{O}_{C_1}, \dots, \mathcal{O}_{C_\ell}) = \prod_{v \in \cup_i \text{Sep}(C_i)} \frac{\text{wt}_C(v, \mathcal{O}_C, t)}{\prod_{j \in [\ell]: C_j \ni v} \text{wt}_{C_j}(v, \mathcal{O}_{C_j}, t)}.$$

*Proof.* We argue by induction. For the base case, when  $C$  is a leaf, we have  $\text{Table}[C, \mathcal{O}_C, t] = \prod_{v \in C} \text{wt}_C(v, \mathcal{O}_C, t)$  which agrees with (12). Otherwise, we inductively use (12) on each

Table $[C_i, \mathcal{O}_{C_i}, t]$  and check that (14), and therefore (12), holds for Table $[C, \mathcal{O}_C, t]$ . The only fact we need here is that if  $v \in \text{Sep}(C_i)$ , all its incident edges belong to  $\text{Link}(C_i)$  and hence their orientations are fixed by  $\mathcal{O}_{C_i}$ .  $\square$

The pseudocode for counting the number of acyclic orientations with maximum indegree  $d$  is described in Algorithm 5.

---

**Algorithm 5:** CountChordalDist ( $G, \mathcal{T}_{C_r}, \eta, S_N, \text{SampleList} = [x^{(1)}, \dots, x^{(t)}], t$ )

---

**Input:** A known chordal skeleton  $G$  which is acyclically orientable with indegree  $d$ , a rooted clique tree  $\mathcal{T}_{C_r}$  of  $G$ , hyperparameter  $\eta$ , a list of samples  $[x^{(1)}, \dots, x^{(t)}]$  from  $P^*$ , time step  $t$ .

**Output:** 3-dimensional table Table.

```

1 Table  $\leftarrow \emptyset$ .
2 NumberOfLevels  $\leftarrow$  the number of levels in  $\mathcal{T}^G$ .
3 for every leaf node  $C \in V(\mathcal{T}_C)$  do
4   for every  $\mathcal{O}_C \in \text{AO}_d(C)$  do
5     for every  $v \in V(G_C)$  do
6        $\text{wt}_C(v, \mathcal{O}_C, t)$  from the samples  $S_N$  according to Equation (11).
7     Table $[C, \mathcal{O}_C, t] = \prod_{v \in C} \text{wt}_C(v, \mathcal{O}_C, t)$  according to Equation (12).
8 // The root is at level 0, and the lowest leaf is at level NumberOfLevels.
9 for  $j = \text{NumberOfLevels} - 1$  to 0 do
10    $\mathcal{S} \leftarrow$  the nodes at level  $j$  in  $\mathcal{T}^G$ .
11   for every non-leaf  $C \in \mathcal{S}$  do
12     Let the children of  $C$  be  $C_1, \dots, C_\ell$ , which are at level  $j + 1$  by definition.
13     for every  $\mathcal{O}_C \in \text{AO}_d(\text{Link}(C))$  do
14       Compute Table $[C, \mathcal{O}_C, t]$  according to Lemma C.11.
15 Return Table.

```

---

Once Table $[\cdot, \cdot, t]$  has been constructed, we can sample an orientation of the chordal graph  $G$  at time step  $t$ . The idea is to go down from the root of  $\mathcal{T}_{C_r}$  to its leaves, level-by-level, with each iteration orienting the link of a maximal clique while being consistent with the orientations sampled so far. For each clique  $C \in V(\mathcal{T}_{C_r})$  with parent  $C_p$ , we suppose that the edges in  $\text{Link}(C_p)$  have been oriented fully. We use the table Table $[\cdot, \cdot, t]$  to sample in a consistent way with the already oriented edges. To argue correctness, we again appeal to Lemma C.6 to see that once a parent clique has been oriented, its children can be oriented independently as long as they are consistent with the parent's orientation. Algorithm 6 gives the pseudocode.

Suppose  $D$  is the orientation of  $G$  that is sampled by SamplingChordalDist for a uniformly chosen value of  $t \in [T]$ . Following Definition A.7, we return the add-one distribution  $\hat{P}$ , with respect to the structure  $D$  and the samples  $S_N$ , as the Bayes net. The proper learning algorithm is summarized as pseudocode in Algorithm 7. The improper learning algorithm is similar, but it uses EWA in place of RWM, and so, it samples a random  $\hat{P}$  when generating samples instead of during the learning phase.

## D Learning Tree-structured distributions

Let us start with the definition of arborescence, which will be used throughout this section.

**Definition D.1** (Arborescence). A directed graph  $G = (V, E)$  is an *out-arborescence* rooted at  $v \in V$  if its skeleton (underlying undirected graph) is a tree (acyclic and connected) and there is a unique directed path from  $v$  to  $w$  for every  $w \in V \setminus \{v\}$ . An *in-arborescence* rooted at  $v$  is similar, but with unique directed paths from all  $w \in V \setminus \{v\}$  to  $v$ .

Given a vertex set  $V = [n]$ , let  $\mathcal{G}_T$  denote the set of out-arborescences rooted at node 1. Note that each  $G \in \mathcal{G}_T$  will have  $m = n - 1$  edges. Any tree-structured distribution  $P$  on  $[k]^n$  is  $G$ -structured for some  $G \in \mathcal{G}_T$ . Let  $\mathcal{C}^{\text{TREE}}$  denote the set of all tree-structured distributions on  $[k]^n$ . Now we have the following lemma.

---

**Algorithm 6:** SamplingChordalDist ( $G, d, \eta, S_{\mathcal{N}}, \text{SampleList} = [x^{(1)}, \dots, x^{(t)}], t$ )

---

**Input:** A chordal skeleton  $G$  which is acyclically orientable with indegree  $d$ , hyperparameter  $\eta > 0$ , a list of samples  $x^{(1)}, \dots, x^{(t)}$  from distribution  $P^*$  on  $[k]^n$ , time step  $t \in [T]$

**Output:** A DAG with skeleton  $G$ .

- 1 Construct a clique tree  $\mathcal{T}^G$  of  $G$  using the algorithm of [12].
  - 2  $r \leftarrow \text{root}(\mathcal{T}^G)$ .
  - 3  $L \leftarrow \text{Leaves}(\mathcal{T}^G)$ .
  - 4  $\text{Stack} \leftarrow \emptyset$ .
  - 5 Call the counting algorithm CountChordalDist ( $G, \mathcal{T}^G, \eta, S_{\mathcal{N}}, \text{SampleList}$ ) to obtain a 3-d DP table Table.
  - 6 Push  $(r, \emptyset)$  to the Stack.
  - 7 **while** Stack is not empty **do**
  - 8     Pop  $(B, \mathcal{O}_p)$  from Stack, where  $B$  is a node in  $\mathcal{T}^G$  and  $\mathcal{O}_p \in \text{AO}_d(\text{Link}(\text{pa}_{\mathcal{T}^G}(B)))$  if  $B \neq r$ .
  - 9     **for every orientation**  $\mathcal{O} \in \text{AO}(\text{Link}(B), \mathcal{O}_p)$  **do**
  - 10         
$$P_B(\mathcal{O}) \leftarrow \frac{\text{Table}[B, \mathcal{O}, t]}{\sum_{\mathcal{O}' \in \text{AO}(\text{Link}(B), \mathcal{O}_p)} \text{Table}[B, \mathcal{O}', t]}$$
  - 11         Sample  $\mathcal{O}_B \in \text{AO}(\text{Link}(B))$  from the distribution  $P_B$ .
  - 12         Fix the orientation of  $\text{Link}(B)$  to be  $\mathcal{O}_B$ .
  - 13         **for each child**  $C$  of  $B$  in  $\mathcal{T}^G$  **do**
  - 14             Push  $(C, \mathcal{O}_B)$  to Stack.
  - 15 Return all the orientations of the edges in  $G$ .
- 

---

**Algorithm 7:** LearnChordalDist (RWM-based proper learning of chordal-structured distributions)

---

**Input :** Sample access to  $P^*$ , Chordal skeleton  $G$ , indegree parameter  $d$ , hyperparameter  $\eta > 0, T$ .

**Output :** A chordal-structured distribution  $\hat{P}$ .

- 1 Let  $S_{\mathcal{N}} \leftarrow s_{\text{AO}}(\varepsilon, \delta)$  samples from  $P^*$  (see Theorem B.8).
  - 2 Sample  $t \leftarrow \{1, \dots, T\}$  uniformly at random.
  - 3  $\text{OnlineSampleList} \leftarrow \emptyset$ .
  - 4 **for**  $s \leftarrow 1$  to  $t$  **do**
  - 5     Observe sample  $x^{(s)} \sim P^*$ .
  - 6      $\text{OnlineSampleList} \leftarrow \text{OnlineSampleList} \cup \{x^{(s)}\}$ .
  - 7 Call SamplingChordalDist ( $G, d, \eta, S_{\mathcal{N}}, \text{OnlineSampleList}, t$ ) to obtain a DAG  $D$ .
  - 8 **return** the Bayes net  $\hat{P} \triangleq$  the add-one distribution with structure  $D$  computed from samples  $S_{\mathcal{N}}$  (Definition A.7).
-

**Lemma D.2.** Consider the finite set  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}} = \mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}_T}$  (see Definition B.7) — where  $\mathcal{G}_T$  is the set of all 1-rooted out-arborescences on vertex set  $[n]$  — of tree-structured Bayes net distributions on  $[k]^n$  (see Definition A.3). Then, for any distribution  $P^* \in \Delta([k]^n)$ ,

$$\min_{Q \in \mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}} D_{\text{KL}}(P^* \| Q) \leq \min_{Q \in \mathcal{C}^{\text{TREE}}} D_{\text{KL}}(P^* \| Q) + \varepsilon,$$

with probability  $\geq 1 - \delta$  over the samples used to construct  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$ . The size of  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$  is bounded by  $\log |\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}| \leq \log |\mathcal{G}_T| \leq n \log(n)$ .

*Proof.* By the definition of tree-structured distributions, every tree-structured distribution will be  $G$ -structured for some  $G \in \mathcal{G}_T$ . Thus, by the construction of  $\mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}_T}$  (see Definition B.7 and Lemma B.10), we will have that  $\min_{P \in \mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}_T}} D_{\text{KL}}(P^* \| P) \leq \min_{P \in \mathcal{C}^{\text{TREE}}} D_{\text{KL}}(P^* \| P) + \varepsilon$  w.p  $\geq 1 - \delta$ . We also have  $|\mathcal{N}_{\varepsilon, \delta}^{\mathcal{G}_T}| \leq |\mathcal{G}_T|$  by construction, and the final bound follows from Cayley’s formula.  $\square$

### D.1 Sampling from the EWA/RWM Distribution

In this section, we describe our approach for sampling tree-structured distribution using EWA and RWM algorithms.

Following Definition A.3, each tree-structured distribution  $P$  in  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$  can be factored as  $(G, V_P)$  where  $G$  is the underlying rooted arborescence and  $V_P = (p_e : e \in E(G), p_1)$  is the vector of functions where  $p_e(x_i, x_j)$  for  $e = (i, j)$  corresponds to  $\Pr_{X \sim P}(X_j = x_j \mid X_i = x_i)$  and  $p_1(x_1)$  corresponds to  $\Pr_{X \sim P}(X_1 = x_1)$ , where  $X_1$  denotes the root node of  $G$ . By construction of  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$  (Lemma D.2), each  $p_i$  will be the add-one distribution (see Definition A.7 and Definition B.7) of  $X_i | X_{\text{pa}(i)}$ , computed using  $s_{\text{AO}}(\varepsilon, \delta)$  samples from  $P^*$  (see Definition B.7).

Let us start with our result for EWA algorithm.

**Lemma D.3.** Let  $\hat{P}_t$  be the output of the EWA algorithm run with the expert set  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$ , parameter  $\eta$ , and horizon  $T \geq t$  for time step  $t$ . For any sequence of observed samples  $x^{(1)}, \dots, x^{(T)} \in [k]^n$ , one can generate a sample from  $\hat{P}_t$  in polynomial time (w.r.t  $n, k, 1/\varepsilon$  and  $\log(1/\delta)$ ).

*Proof.* In the EWA algorithm,  $\hat{P}_t$  is a mixture of the distributions in  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$ . For a distribution  $P \in \mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$  that factors into  $(G, (p_e : e \in E(G), p_1))$  as above (where  $p_1$  and all  $p_e : e \in E(G)$  are add-one distributions (see Definition A.7 and Definition B.7)), let the weight of  $P$  in  $\hat{P}_t$  be  $\omega(P)$ . We have the following (note that, in the analysis below, we use  $q_1, q_e : e \in E(H)$  for the add-one distributions w.r.t DAG  $H$  to distinguish them from  $p_1, p_e : e \in E(G)$  (w.r.t DAG  $G$ )):

$$\begin{aligned} \omega(P) &= \frac{\prod_{s < t} \exp_{\eta} \log \left( p_1(x_1^{(s)}) \cdot \prod_{e \in E(G)} p_e(x_e^{(s)}) \right)}{\sum_{H \in \mathcal{G}_T} \prod_{s < t} \exp_{\eta} \log \left( q_1(x_1^{(s)}) \cdot \prod_{e \in E(H)} q_e(x_e^{(s)}) \right)} \\ &= \frac{\prod_{s < t} \exp_{\eta} \log p_1(x_1^{(s)}) \cdot \prod_{e \in E(G)} \prod_{s < t} \exp_{\eta} \log p_e(x_e^{(s)})}{\prod_{s < t} \exp_{\eta} \log q_1(x_1^{(s)}) \cdot \sum_{H \in \mathcal{G}_T} \prod_{e \in E(H)} \prod_{s < t} \exp_{\eta} \log q_e(x_e^{(s)})} \\ &= \frac{\prod_{s < t} \exp_{\eta} \log p_1(x_1^{(s)})}{\prod_{s < t} \exp_{\eta} \log q_1(x_1^{(s)})} \cdot \prod_{e \in E(G)} \frac{\prod_{s < t} \exp_{\eta} \log p_e(x_e^{(s)})}{\sum_{H \in \mathcal{G}_T} \prod_{e \in E(H)} \prod_{s < t} \exp_{\eta} \log q_e(x_e^{(s)})}, \end{aligned} \tag{15}$$

where we note that  $p_1 = q_1$  always (both are add-one distributions from the same set of samples, and node 1 is the root node in both  $G$  and  $H$  by definition of  $\mathcal{G}_T$ ).

We can interpret (15) as follows: the whole product gives the probability of sampling structure  $G$  (and hence a Bayes net distribution  $P$  on  $[k]^n$  when  $G$  is combined with the appropriate add-one distribution) as a product distribution over  $G$ ’s edges. Hence, we can sample  $G$  edge-wise,

recursively following the above product distribution, as long as we can sample from the edge-probability distributions efficiently. The difficulty lies in the fact that the normalization factors involve exponentially many terms (going over each  $H \in \mathcal{G}_T$ ).

$G$  is sampled from a weighted mixture of spanning arborescences of the complete directed graph on  $n$  nodes, where the weight of a spanning arborescence  $H$  is proportional to  $\prod_{e \in E(H)} w(e)$  and

$w(e) = \prod_{s < t} \exp_{\eta} \log q_e(x_e^{(s)})$  for any  $e = (i, j) \in [n]^2$  ( $q_e$  is an add-one distribution). Sampling weighted arborescences is known to be in polynomial time ([3, 7, 14, 8]), using Tutte's theorem; for completeness, we describe the algorithm in Appendix D.3. Once  $G$  is sampled, we can construct  $P$  from a sufficiently large set  $S_N$  of samples from  $P^*$  ( $|S_N| = s_{AO}(\varepsilon, \delta)$ , see definition B.7). First, we fix the distribution  $p_1$  (of  $X_1$ ) and then we follow the edge structure of  $G$  to fix the distribution of each  $X_i | X_{pa(i)}$  as the appropriate add-one distribution (see Definition A.7).

Once  $P$  is constructed, a sample from  $P$  can be generated in polynomial time (in  $n, k$ ) as  $P$  is tree-structured. This completes the proof of the lemma.  $\square$

Now let us state our result for RWM algorithm.

**Lemma D.4.** *If the RWM algorithm is run with the expert set  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$ , parameter  $\eta$ , and horizon  $T$ , the prediction  $\hat{P}_t$  at time step  $t \leq T$  can be computed in polynomial time, for any sequence of observed samples  $x^{(1)}, \dots, x^{(T)} \in [k]^n$ .*

*Proof.* This follows directly from the argument used to prove Lemma D.3, since in RWM,  $\hat{P}_t$  is sampled during the execution of the learning algorithm rather than during the sampling stage as in EWA.  $\square$

## D.2 Learning Guarantees

In this section, we will state the learning guarantees for tree-structured distributions for both the improper and proper settings. Let us start by describing our result for the improper setting.

**Theorem D.5.** *Let  $P^*$  be an unknown distribution over  $[k]^n$ , and  $\varepsilon, \delta \in (0, 1)$  be parameters. Given sample access to  $P^*$ , there exist algorithms outputting efficiently-sampleable distributions (which are mixtures of trees) giving the following guarantees:*

- (i) *With probability at least  $2/3$ , the output distribution  $Q$  is an  $(\varepsilon, 4)$ -approximation of  $P^*$ ; the sample complexity is  $\tilde{O}\left(\frac{nk^2 \log^2(nk/\varepsilon)}{\varepsilon}\right)$ .*
- (ii) *A realizable  $(\varepsilon, \delta)$ -PAC learner for  $\mathcal{C}^{\text{TREE}}$  with sample complexity  $\tilde{O}\left(\frac{nk^2 \log^2(nk/\delta\varepsilon)}{\delta\varepsilon}\right)$ .*
- (iii) *An agnostic  $(\varepsilon, \delta)$ -PAC learner for  $\mathcal{C}^{\text{TREE}}$  with sample complexity  $\tilde{O}\left(n^4 \varepsilon^{-4} \log^4(nk/\varepsilon\delta) + nk^2 \varepsilon^{-1} \log^2(nk/\varepsilon\delta)\right)$ .*

*Proof.* The algorithm in all cases is the EWA-based improper learning algorithm (Algorithm 3) instantiated with appropriate  $\mathcal{N} = \mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$  as in the proof of Theorem B.11. The correctness and sample complexities in each case follow from Theorem B.11. The fact that the output distribution is efficiently sampleable is shown in Lemma D.3.  $\square$

In the realizable case (where  $P^*$  is a tree-structured distribution), this sample complexity bound matches the following lower-bound in [5, Appendix A.2] up to log-factors for constant error probability.

**Lemma D.6** (Restatement of lower bound result in Appendix A.2 of [5]). *Let  $P^*$  be an unknown tree-structured distribution defined over  $\{0, 1\}^n$ , and  $\varepsilon \in (0, 1)$  be a parameter. Any algorithm that outputs a distribution  $Q$  which is an  $\varepsilon$ -approximation of  $P^*$  with constant probability requires  $\Omega(\frac{n}{\varepsilon})$  samples from  $P^{*4}$ .*

<sup>4</sup>The lower bound stated in [5] holds for general Bayes net distributions of indegree  $\leq d$ . For tree-structured distributions, we have  $d = 1$ . Moreover, their lower bound is with respect to TV-distance, which translates to KL-divergence due to Pinsker's inequality.

Now we will state the guarantees for proper learning of the tree-structured distributions.

**Theorem D.7.** *Let  $P^*$  be an unknown distribution over  $[k]^n$ , and  $\varepsilon, \delta \in (0, 1)$  be parameters. Given sample access to  $P^*$ , there exists a proper agnostic  $(\varepsilon, \delta)$ -PAC-learner for  $\mathcal{C}^{\text{TREE}}$  that takes  $\tilde{O}\left(\frac{n^3}{\varepsilon^2 \delta^2} \log^3\left(\frac{nk}{\varepsilon \delta}\right) + \frac{nk^2}{\varepsilon} \log^2\left(\frac{nk}{\varepsilon \delta}\right)\right)$  samples from  $P^*$  and runs in time  $O(\text{poly}(n, k, 1/\varepsilon, 1/\delta))$ .*

*Proof.* The algorithm is the RWM-based proper learning algorithm (Algorithm 4) applied with  $\mathcal{N} = \mathcal{N}_{\varepsilon/2, \delta/2}^{\text{TREE}}$ ,  $\eta = \sqrt{(8 \log |\mathcal{N}|)/T}$  and the log loss function. The algorithm, by construction, will output a tree-structured distribution in  $\mathcal{N}$ . The correctness (agnostic PAC guarantee) and sample complexity follow from exactly the same argument as in Theorem B.12, taking  $d = 1$  since tree-structured distributions are Bayes nets with in-degree 1. The fact that the RWM-computation can be done in  $\text{poly}(n, k, \frac{1}{\varepsilon}, \frac{1}{\delta})$  time, in spite of having exponentially many distributions in  $\mathcal{N}$ , follows from the fact that the sampling from the weight distribution, for each time step  $t$ , can be done efficiently when  $\mathcal{N} = \mathcal{N}_{\varepsilon/2, \delta/2}^{\text{TREE}}$  (see Lemma D.4).  $\square$

### D.3 Proof of Lemma D.3

Given a vertex set  $V = [n]$ ,  $\mathcal{G}_T$  denotes the set of all spanning out-arborescences rooted at node 1.  $\mathcal{C}^{\text{TREE}}$  denotes the set of all tree-structured distributions on  $[k]^n$ .  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$  denotes the finite set of distributions (see Lemma D.2 and Definition B.7) that *finitizes*  $\mathcal{C}^{\text{TREE}}$  with error  $\varepsilon$  in  $D_{\text{KL}}$  and failure probability  $\delta$ . Let  $S_{\mathcal{N}}$  denote the set of samples used to construct  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$ . By construction, each tree-structured distribution  $P$  in  $\mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$  can be factored as  $(G, V_P)$  where  $G$  is the underlying rooted arborescence and  $V_P = (p_e : e \in E(G), p_1)$  is the vector of functions where  $p_e(x_i, x_j)$  for  $e = (i, j)$  corresponds to  $\Pr_{X \sim P_{S_{\mathcal{N}}, j, \{i\}}^+}(X_j = x_j \mid X_i = x_i)$  and  $p_1(x_1)$  corresponds to  $\Pr_{X \sim P_{S_{\mathcal{N}}, 1, \emptyset}^+}(X_1 = x_1)$  (here, as in Definition A.7,  $P_{S_{\mathcal{N}}, v, \{\text{pa}(v)\}}^+$  denotes the add-one distribution computed from  $S_{\mathcal{N}}$  for node  $v$  when the parent set is  $\{\text{pa}(v)\}$ ).

In the proof of Lemma D.3, we derive the following expression for  $\omega_t(P)$ , the weight assigned to a tree-structured distribution  $P \cong (G, (p_e : e \in E(G), p_1))$  in the discretized set  $\mathcal{N} = \mathcal{N}_{\varepsilon, \delta}^{\text{TREE}}$  by the EWA/RWM algorithms at step  $t$ .

$$\omega_t(P) = \prod_{e \in E(G)} \frac{\prod_{s < t} \exp_{\eta} \log p_e(x_e^{(s)})}{\sum_{H \in \mathcal{G}_T} \prod_{e \in E(H)} \prod_{s < t} \exp_{\eta} \log q_e(x_e^{(s)})}, \quad (16)$$

where  $p_e : e \in E(G)$  denotes the add-one-distributions w.r.t structure  $G$ , and  $q_e : e \in E(H)$  denotes the add-one distributions w.r.t structure  $H$  (all distributions computed from the samples  $S_{\mathcal{N}}$  with  $|S_{\mathcal{N}}| = s_{\text{AO}}(\varepsilon, \delta) \leq \tilde{O}\left(\frac{nk^2 \log^2(nk/\varepsilon \delta)}{\varepsilon}\right)$ , see Definition B.7).

This gives the following high-level algorithm for sampling from the EWA/RWM distribution.

- (i) Sample a spanning arborescence  $G \in \mathcal{G}_T$  (rooted at node 1), where  $\Pr(G) \propto \prod_{e \in E(G)} \mathbf{w}(e)$

and  $\mathbf{w}(e) \triangleq \left(\prod_{s < t} P_{S_{\mathcal{N}}, j, \{i\}}^+(x_e^{(s)})\right)^{\eta}$  for any  $e = (i, j) \in [n]^2$ . This can be done in polynomial time (even though there are exponentially many such arborescences) by applying Tutte's matrix-tree theorem to weighted digraphs ([3, 7, 14, 8]). We describe this approach in the algorithm `SAMPLINGARBORESCENCE`, which we describe and analyze later. We invoke

$$G \leftarrow \text{SAMPLINGARBORESCENCE}(G_0 \leftarrow K_n, w \leftarrow \mathbf{w}, r \leftarrow 1),$$

where  $K_n$  is the complete graph on the vertex set  $[n]$  viewed as a digraph, which returns  $G \in \mathcal{G}_T$  (rooted at 1) with the required sampling probability.

- (ii) For each  $e = (i, j) \in E(G)$  (fixed in the preceding step), set the node distribution of node  $j$ ,  $\hat{p}_j(x_j | x_i)$ , to be the add-one distribution  $P_{S_{\mathcal{N}}, j, \{i\}}^+$ . Also let  $\hat{p}_1$  be the add-one distribution  $P_{S_{\mathcal{N}}, 1, \emptyset}^+$ . Finally, let  $P$  be the Bayes net  $(G, (\hat{p}_1, \dots, \hat{p}_n))$ .

Once  $P$  is sampled, a sample from  $P$  can be generated in polynomial time as  $P$  is tree-structured.

**Correctness** The correctness of the above algorithm follows from considering each factor of Equation (16) and noting that the steps (i)-(ii) described above are independent.

**Running Time** The `SAMPLINGARBORESCENCE` call in step (i), in this case (invoked with  $K_n$ ), requires  $O(n^5)$  time, and the weights required (for all the edges in  $K_n$ ) can be computed in  $O(n^2k \times |s_{AO}(\varepsilon, \delta)|)$  time. The add-one distributions used in step (ii) can again be computed in  $O(nk \cdot s_{AO}(\varepsilon, \delta))$  time (one pass over the samples, with fixed  $G$  giving the parent, for each node). which gives an overall time complexity which is  $\tilde{O}(n^8k^3/\varepsilon\delta)$ .

### The SamplingArborescence algorithm

**Preliminaries** Let  $G = ([n], E, w)$  be a connected weighted directed graph on  $n$  vertices  $\{1, \dots, n\}$  with  $m$  edges  $e_1, \dots, e_m$ .  $w : E \rightarrow \mathbb{R}_{>0}$  is a positive weight function. Let  $A_w(G)$  denote the  $n \times n$  weighted vertex adjacency matrix of  $G$ . For every  $i, j \in [n]$ ,  $A_w(G)$  is defined as follows:

$$[A_w(G)]_{i,j} = \begin{cases} w(e_k), & \text{if } e_k \text{ is the directed edge from } i \text{ to } j \\ 0, & \text{if there is no directed edge from } i \text{ to } j \end{cases}$$

Let  $N_{\text{in},w}(G)$  be the weighted *inward* incidence matrix of order  $m \times n$  defined as follows: For every  $i \in [n]$  and  $k \in [m]$ , we have:

$$[N_{\text{in},w}(G)]_{k,i} = \begin{cases} \sqrt{w(e_k)}, & \text{if directed edge } e_k \text{ points to vertex } i \\ 0, & \text{otherwise} \end{cases}$$

Similarly, we can define the weighted *outward* incidence matrix  $M_{\text{out},w}(G)$  of order  $n \times m$  as follows, for  $i \in [n]$  and  $k \in [m]$ :

$$[M_{\text{out},w}(G)]_{i,k} = \begin{cases} \sqrt{w(e_k)}, & \text{if directed edge } e_k \text{ points from vertex } i \\ 0, & \text{otherwise} \end{cases}$$

The *indegree* matrix  $D_{\text{in},w}(G)$  is a  $n \times n$  diagonal matrix such that, for all  $i \in [n]$ ,  $[D_{\text{in},w}(G)]_{i,i}$  is equal to the sum of the weights of all incoming edges to vertex  $i$ . Similarly, the *outdegree* matrix  $D_{\text{out},w}(G)$  is a  $n \times n$  diagonal matrix such that for all  $i \in [n]$ ,  $[D_{\text{out},w}(G)]_{i,i}$  is equal to the sum of the weights of all outgoing edges from vertex  $i$ .

Now we can define the *Laplacian matrices*  $L_{1,w}(G)$  and  $L_{2,w}(G)$  associated with digraph  $G$  as follows:

$$L_{1,w}(G) \triangleq D_{\text{in},w}(G) - A_w(G), \text{ and } L_{2,w}(G) = D_{\text{out},w}(G) - A_w^\top(G). \quad (17)$$

For all these associated matrices, we omit  $G$  and denote the matrix  $A_w(G)$  as  $A_w$  etc. when  $G$  is clear from the context. Now we have the following relationships between the Laplacian and incidence matrices:

**Claim D.8** (Equation (10) of [8]). (i)  $L_{1,w} = (N_{\text{in},w}^\top - M_{\text{out},w})N_{\text{in},w}$ .

(ii)  $L_{2,w} = (M_{\text{out},w} - N_{\text{in},w}^\top)M_{\text{out},w}^\top$ .

For a vertex  $r \in [n]$ , let  $L_{1,w}^r$  be the  $(n-1) \times (n-1)$  matrix obtained by removing the  $r$ -th row and  $r$ -th column of  $L_{1,w}$ . Similarly, we also define  $N_{\text{in},w}^r$  and  $M_{\text{out},w}^r$ . Then we have the following:

**Claim D.9** (Equation (11) of [8]). (i)  $L_{1,w}^r = ((N_{\text{in},w}^r)^\top - M_{\text{out},w}^r)N_{\text{in},w}^r$ .

(ii)  $L_{2,w}^r = ((M_{\text{out},w}^r - (N_{\text{in},w}^r)^\top))(M_{\text{out},w}^r)^\top$ .

**Definition D.10** (Weight of a graph). Let  $G = (V, E, w)$  be a weighted (directed or undirected) graph, where  $w : E \rightarrow \mathbb{R}_{>0}$ . Let  $H$  be any subgraph of  $G$ . The weight of graph  $H$  is defined as the *product* of the weights of its edges, i.e.

$$w(H) = \prod_{e \in E(H)} w(e).$$

**Definition D.11** (Contraction and deletion). If  $G = (V, E, w)$  is a weighted undirected graph (not necessarily simple) and  $e = \{i, j\} \in E$ , the graph  $G/e$  obtained by *contracting* edge  $e$  is a weighted graph on  $V \setminus \{i, j\} \cup \{\langle i, j \rangle\}$  where

- Vertices  $i$  and  $j$  are removed, and a new vertex  $\langle i, j \rangle$  is added.
- All edges  $\{i, j\}$  (in case of parallel edges) are removed in  $G/e$ .
- Edges  $\{u, v\}$  where  $u, v \in V \setminus \{i, j\}$  are preserved in  $G/e$  with the same weight.
- Every edge  $\{u, v\} \in E$  where  $u \in \{i, j\}$  and  $v \in V \setminus \{i, j\}$  becomes an edge  $\{\langle i, j \rangle, v\}$  in  $G/e$  with the same weight.

Similarly the graph  $G \setminus e$  obtained by *deleting* edge  $e$  is just  $(G \setminus e) = (V, E \setminus e, w|_{E \setminus e})$  where the edge  $e$  is deleted (including all parallel edges) from  $G$ .

**Definition D.12** (Contraction edge mapping). With the above definition of contraction (with parallel edges kept), if  $G$  is a graph and  $G' = G \setminus e$  for  $e = \{i, j\} \in E(G)$ , we can map each edge  $e' \in E(G')$  *injectively* to an edge  $e \in E(G)$  that caused its inclusion in  $G'$ . We denote this mapping by  $f_{G':G} : E(G') \rightarrow E(G)$ .

The following theorem is a generalization of Tutte's matrix-tree theorem to weighted graphs.

**Lemma D.13** (Theorem 3 of [8]). *Let  $G = (V, E, w)$  be a weighted directed graph on  $n$  vertices  $[n]$  with positive weight function  $w : E \rightarrow \mathbb{R}_{>0}$ . Then, for any  $r \in [n]$ , the sum of the weights of the weighted outgoing (incoming) spanning arborescences rooted at vertex  $r$  is equal to  $\det(L_{1,w}^r)$  ( $\det(L_{2,w}^r$ ), respectively).*

Now the next corollary follows as any undirected graph can be viewed as a directed graph with edge  $\{i, j\}$  corresponding to a pair of directed edges  $(i, j)$  and  $(j, i)$ .

**Corollary D.14.** *If  $G = K_n$  (the complete graph on  $n$  vertices) and  $w : [n]^2 \rightarrow \mathbb{R}_{>0}$  is any positive weight function,*

$$\det(L_{1,w}^1) = \sum_{G \in \mathcal{G}_T} \prod_{e \in E(G)} w(e),$$

where  $\mathcal{G}_T$ , as defined earlier, is the set of all spanning out-arborescences on  $[n]$  rooted at vertex 1.

**The algorithm** If  $G$  is a graph and  $v \in V(G)$  be a vertex of  $G$ , let  $\mathcal{A}_{G,v}$  denote the set of out-arborescences on  $V(G)$  which are subgraphs of  $G$  (viewed as a digraph), rooted at  $v$ , and span the set of nodes reachable from  $v$ . Note that this set will not be empty unless  $v$  is an isolated vertex.

**Definition D.15** (Product arborescence distribution). Let  $G = (V, E, w)$  be a weighted graph where  $w : E \rightarrow \mathbb{R}_{\geq 0}$  is a weight function on the edges. For any  $v \in V(G)$ , a distribution  $\mu$  on  $\mathcal{A}_{G,v}$  is said to be the *product arborescence distribution* if

$$\Pr_{G \sim \mu}(G) \propto \prod_{e \in G} w(e) \text{ for all } G \in \mathcal{A}_{G,v}.$$

**Claim D.16.** *Suppose  $G = (V, E, w)$  is a weighted digraph with non-negative weights,  $r \in V(G)$  and  $\mu$  is a product arborescence distribution on  $\mathcal{A}_{G,r}$ . Then*

$$\Pr_{T \sim \mu}(e \in T) = 1 - \frac{\det(L_{1,w}^r(G \setminus e))}{\det(L_{1,w}^r(G))}.$$

*Proof.* Since  $\Pr_{\mu}(T) \propto \prod_{e \in T} w(e)$ , we have

$$\Pr_{T \sim \mu}(e \in T) = \frac{\sum_{T \in \mathcal{A}_{G,r} : T \ni e} w(T)}{\sum_{T \in \mathcal{A}_{G,r}} w(T)} = 1 - \frac{\sum_{T \in \mathcal{A}_{G,r} : T \not\ni e} w(T)}{\sum_{T \in \mathcal{A}_{G,r}} w(T)} = 1 - \frac{\det(L_{1,w}^r(G \setminus e))}{\det(L_{1,w}^r(G))},$$

where the first equality follows from Definition D.15 and Definition D.10, and the third equality follows from applying Lemma D.13 to  $G \setminus e$  and  $G$ .  $\square$

Now we are ready to prove the correctness of the algorithm `SAMPLINGARBORESCENCE`.



---

**Algorithm 8:** The SAMPLINGARBORESCENCE algorithm

---

**Input:**  $G_0$  a directed graph,  $w : E(G_0) \rightarrow \mathbb{R}_{\geq 0}$  weight function,  $r \in V(G_0)$  root node.

**Output:** Arborescence  $G_T \in \mathcal{A}_{G_0, r}$  with  $\Pr(\text{output } G_T) \propto \prod_{e \in E(G_T)} w(e)$ .

```
1 Let  $G_T \leftarrow \emptyset$ .
2 Choose an arbitrary ordering of the  $m$  edges of  $G_0$ , and let  $\text{EdgeList} \leftarrow [e_1, \dots, e_m]$ .
3 Let  $\text{ContractionMapping}[e_i] \leftarrow e_i$  for each  $i \in [m]$ .
4 Let  $j \leftarrow 0$ .
5 while  $r$  has an outgoing edge in  $G_j$  do
6   Let  $e = (r, x)$  be the first remaining outgoing edge.
7   Compute  $p_e \leftarrow \frac{\det(L_{1,w}^r(G_j \setminus e))}{\det(L_{1,w}^r(G_j))}$ .
8   Sample  $u \sim \text{Unif}((0, 1])$ .
9   if  $u \leq p_e$  (with probability  $p_e$ ) then
10     $G_{j+1} \leftarrow G_j \setminus e$  (delete edge  $e$ ).
11     $j \leftarrow j + 1$ .
12  else
13     $G_{j+1} \leftarrow G_j / e$  (contract edge  $e$ ).
14    Add  $\text{ContractionMapping}[e]$  to  $G_T$  (this will be an edge of  $G_0$ ).
15    /* Reconstruct  $\text{ContractionMapping}$  for mapping  $E(G_{j+1})$  to  $E(G_0)$ . */
16     $\text{CM}' \leftarrow$  empty dictionary.
17    for each edge  $e \in E(G_{j+1})$  do
18       $\text{CM}'[e] \leftarrow \text{ContractionMapping}[f_{G_{j+1}:G_j}(e)]$  (see Definition D.12).
19     $\text{ContractionMapping} \leftarrow \text{CM}'$ 
20     $r \leftarrow \langle r, x \rangle \in V(G_{j+1})$  (the newly-contracted vertex).
21     $j \leftarrow j + 1$ .
22 return  $G_T$ .
```

---

**Claim D.17.** *There is an algorithm SAMPLINGARBORESCENCE that, if invoked with input  $(G, w, r)$  for a digraph  $G, w : E(G) \rightarrow \mathbb{R}_{\geq 0}$ , and  $r \in V(G)$ , returns a random arborescence  $G_T$  sampled from the product arborescence distribution on  $\mathcal{A}_{G, r}$ . The algorithm runs in time  $O(|E||V|^3)$  in the real RAM model.*

*Proof.* The pseudocode for the algorithm is given as Algorithm 8. Observe that, if  $G_T \in \mathcal{A}_{G, r}$ , then it is actually sampled from the product arborescence distribution (Definition D.15) by Claim D.16, since each edge  $e$  is added to  $G_T$  with the correct probability  $1 - p_e$ .

It remains to ensure that  $G_T$  is indeed a spanning arborescence. Let  $r_j$  denote the value of  $r$  used in iteration  $j$  of the while loop (lines 5-20).  $r_j \in V(G_j)$  by construction. Let  $\mathcal{S}(r_j) \subseteq V(G_0)$  denote  $\{r_j\}$  if  $V(G_j) = V(G_0)$  (no contractions) and the set of nodes in  $G_0$  corresponding to the contracted-vertex in  $G_j$  otherwise.

By construction, when  $|\mathcal{S}(r_j)| > 1$ , the algorithm maintains (in  $G_T$ ) an out-arborescence rooted at  $r = r_0$  that spans  $\mathcal{S}(r_j)$ . This is because when the algorithm selects an edge  $e = (r_j, x) \in E(G_j)$  and takes  $G_{j+1} = G_j / (r_j, x)$ , this edge will correspond to an edge  $(v, x)$  in the original graph  $G_0$  where  $v \in \mathcal{S}(r_j)$  and  $x \notin \mathcal{S}(r_j)$ . This will ensure that there are no (undirected) cycles formed when adding  $(v, x)$  to  $G_T$  and the rooted arborescence invariant will be maintained in iteration  $j + 1$  with  $\mathcal{S}(r_{j+1}) = \mathcal{S}(r_j) \sqcup \{x\}$ . Note that when an edge  $e$  is selected for contraction in iteration  $j$ , the algorithm maintains a mapping  $\text{ContractionMapping}$  from the edges in  $G_j$  to the edges in  $G_0$  and adds the original edge (in  $G_0$ ) to  $G_T$ . The other property of the algorithm is that it does not *disconnect* the graph — an edge  $e$  which is a *cut edge* will be present in all arborescences and hence will have  $p_e = 0$ , ensuring that it will never be selected for deletion (in line 9). So, the algorithm will terminate when  $\mathcal{S}(r_j)$  is the set of nodes reachable from  $r = r_0$ , and  $G_T$  will have the required spanning arborescence.

For the time complexity bound, note that the while loop (lines 5-20) will run at most  $|E|$  times, while the expensive step inside the while loop will be the two determinant computations in line 7 (to compute  $p_e$ ). These can be done in  $O(|V|^3)$  time in the real RAM model, by Gaussian elimination etc.

Contraction, deletion etc. require only  $O(|V| + |E|) \leq O(|V|^2)$  time. This gives us the  $O(|E||V|^3)$  bound.  $\square$

## E Lower bound for learning tree-structured distributions

In this section, we show a lower bound of  $\Omega(\frac{n}{\varepsilon})$  holds for our tree structure learning results. This follows from [2]. We are including it here for completeness.

**Theorem E.1.** *Given sample access to an unknown tree-structured discrete distribution defined over  $[k]^n$ , and  $\varepsilon > 0$  be a parameter. With probability at least  $9/10$ ,  $\Omega(\frac{n}{\varepsilon})$  samples are necessary for any realizable PAC learner for learning  $\mathcal{C}^{\text{TREE}}$ .*

We will prove the above result for  $k = 2$ , that is, for distributions defined over  $\{0, 1\}^n$ . The lower bound immediately extends to distributions over  $[k]^n$ .

We will first prove the above result for  $n = 3$  nodes. Then we will extend the result for  $n = 3\ell$ , for some positive integer  $\ell$ .

**Definition E.2** (Hellinger distance). Let  $P$  and  $Q$  be two probability distributions defined over the same sample space  $\mathcal{D}$ . The Hellinger distance between  $P$  and  $Q$  is defined as follows:

$$H(P, Q) = \sqrt{\frac{1}{2} \sum_{x \in \mathcal{D}} \left( \sqrt{P(x)} - \sqrt{Q(x)} \right)^2}$$

**Fact E.3.** Let  $P$  and  $Q$  be two probability distributions defined over the same sample space  $\mathcal{D}$ . Then the following holds:

$$H(P, Q) \leq \sqrt{\frac{1}{2} D_{\text{KL}}(P||Q)}$$

We will also need the following folklore result.

**Fact E.4.** Let  $P$  be an unknown discrete distribution defined over  $\{0, 1\}^3$ , and  $\varepsilon > 0$  be a parameter. In order to output a distribution  $\hat{P}$  defined over  $\{0, 1\}^3$  such that  $H(P, \hat{P}) \leq \sqrt{\varepsilon}$ ,  $\Omega(\frac{1}{\varepsilon})$  samples from  $P$  are necessary.

**Lemma E.5.** *There exist three tree-structured distributions  $P_1, P_2, P_3$  defined over  $\{0, 1\}^3$ , such that any algorithm that can learn any distribution  $P_i$  in up to  $\varepsilon$ -KL-distance, that is, can output a distribution  $\hat{P}_i$  such that  $D_{\text{KL}}(P_i||\hat{P}_i) \leq \varepsilon$ , requires  $\Omega(\frac{1}{\varepsilon})$  samples from  $P_i$ , for any  $i \in [3]$ .*

$P_1$  has three nodes  $X_1, Y_1, Z_1$ ,  $P_2$  has three nodes  $X_2, Y_2, Z_2$  and  $P_3$  has three nodes  $X_3, Y_3, Z_3$ . In  $P_1$ ,  $(Y_1, Z_1)$  takes values uniformly at random between  $(0, 0)$  and  $(1, 1)$  and  $X_1$  copies  $Y_1$  and  $Z_1$  with probability  $(1 - \varepsilon)$ , and with the remaining probability, takes a value from  $\text{Ber}(\frac{1}{2})$ .  $P_2$  and  $P_3$  are defined in a similar fashion where we set  $X_2 = Z_2$  and  $X_3 = Y_3$ , respectively.

We will use the following results from [2].

**Claim E.6** (See Fact 7.4 (i) and Lemma 7.5 (i) of [2]). (i)  $H(P_i, P_j) \geq 10\sqrt{\varepsilon}$  for any  $i \neq j \in [3]$ .

(ii) Every tree  $\mathcal{T}$  on 3 vertices is not  $\Theta(\varepsilon \log \frac{1}{\varepsilon})$ -approximate for any of  $P_1, P_2$  and  $P_3$ .

*Proof of Lemma E.5.* We will prove this by contradiction. Suppose there exists an algorithm  $\mathcal{A}$  that can learn any distribution up to  $\varepsilon$ -KL-distance using  $o(\frac{1}{\varepsilon})$  samples, by outputting the pmf of  $\hat{P}$ . In that case, we will argue that using  $\mathcal{A}$ , we can distinguish between  $P_1, P_2$  and  $P_3$ .

Let us denote the unknown distribution be  $P$ , and suppose given sample access to  $P$ ,  $\mathcal{A}$  takes  $o(1/\varepsilon)$  samples from  $P$  and outputs a distribution  $\hat{P}$  such that  $H(P, \hat{P}) \leq \sqrt{\varepsilon}$ . This implies that given sample access to either of  $P_i$  with  $i \in [3]$ ,  $\mathcal{A}$  can output such a  $\hat{P}$  such that  $H(P_i, \hat{P}) \leq \sqrt{\varepsilon}$ , which contradicts Fact E.4. Using Fact E.3, we are done with the proof of the lemma.  $\square$

Now we are ready to prove Theorem E.1.

*Proof of Theorem E.1.* Let us assume that  $n = 3\ell$  for some positive integer  $\ell$ . We will divide the  $n$  variables into  $\ell$  blocks, each of size 3 nodes. We will prove this by contradiction. We will define a distribution  $P$  over  $\{0, 1\}^n$ , where the  $i$ -th block is chosen uniformly to be either  $P_1$  or  $P_2$ .

Suppose there exists an algorithm that can take  $o(\frac{n}{\varepsilon})$  samples and outputs an  $\varepsilon$ -approximate tree  $T$  of  $P$ , with probability at least  $9/10$ . Since we have chosen each block independently,  $T$  is a disjoint union of  $T_1, \dots, T_\ell$ . Using Claim E.6, and setting  $\varepsilon$  as  $\varepsilon/\ell$ , we can say that each  $T_i$  is not  $\Theta(\varepsilon/\ell)$ -approximate with probability at least  $2/3$ . Thus, using Chernoff bound, we can say that at least  $\frac{\ell}{100}$  trees are not  $\frac{C\varepsilon}{\ell}$  approximate. Thus,  $D_{\text{KL}}(P||\hat{P}) \geq \varepsilon$  for a suitable constant  $C$  for any tree-structured distribution  $\hat{P}$ . This completes the proof of the theorem.  $\square$

## F Learning Bayesian networks with bounded vertex cover

In this section, we show that bayesian networks can be learned efficiently if the size of the vertex cover of the associated moralized graph is bounded. Before proceeding to present our result, we need some notations.

Let  $G = (V, E)$  be DAG with the vertex set  $V$  and the edge set  $E$ . Here each vertex  $v \in V$  corresponds to a variable and the edges in  $E$  encode the conditional independence relations between the nodes of  $V$ . Let  $\mathcal{D}$  denote the set of all possible DAG on the vertex set  $V$ . Moreover, for every DAG  $G$ , let us associate it with a non-negative weight function  $f : G \rightarrow \mathbb{R}^+ \cup \{0\}$  which encodes how well  $G$  fits a given dataset. Now we are ready to define the notion of modular weight function, which will be crucially used in our proofs.

**Definition F.1** (Modular weight function). Let  $G(V, E)$  be a DAG with vertex set  $V$  and edge set  $E$ . Moreover, let  $f : G \rightarrow \mathbb{R}^+ \cup \{0\}$  denotes the weight function associated with  $G$ . The weight function  $f$  is said to be a *modular* weight function if it has the following form:

$$f(G) = \prod_{v \in V} f_v(G_v) \quad (18)$$

Now we define the notion of the vertex cover number of a DAG below.

**Definition F.2** (Vertex cover number of a DAG). Given a DAG  $G$ , let  $G_M$  denote the undirected *moralized graph* corresponding to  $G$ . The *vertex cover number*  $\tau(G)$  of  $G$  is the size of the smallest vertex cover of  $G_M$ . Moreover, for some integer  $\ell$ , the set of DAGs  $G$  such that  $\tau(G) \leq \ell$  is denoted as  $\mathcal{DAG}^{(\ell)}$ .

Now we are ready to state the main result that we will be proving in this section.

**Theorem F.3.** Let  $P^*$  be an unknown discrete distribution on  $[k]^n$  defined over a DAG  $G$  such that the size of the vertex cover of the moralized graph  $G_M$  corresponding to  $G$  is bounded by an integer  $\ell$ . Moreover, let  $\mathcal{DAG}^{(\ell)}$  be the family of distributions over  $[k]^n$  that can be defined as Bayes nets over DAGs whose vertex cover size of the associated moralized graph is bounded by  $\ell$ . Given sample access from  $P^*$ , and a parameter  $\varepsilon > 0$ , there exist the following:

- (i) An *agnostic PAC-learner* for  $\mathcal{DAG}^{(\ell)}$  which uses  $O\left(\frac{\ell^4 n^4}{\varepsilon^4} \log^4\left(\frac{nk\ell}{\varepsilon}\right) \log\left(\frac{1}{\delta}\right) + \frac{\ell^3 nk^{\ell+1}}{\varepsilon} \log^2\left(\frac{nk}{\varepsilon\delta}\right)\right)$  samples and runs in time  $O(\exp(\ell)\text{poly}(n))$  that is *improper* and returns an *efficiently samplable mixture* of distributions from  $\mathcal{DAG}^{(\ell)}$ .
- (ii) An *agnostic PAC-learner* for  $\mathcal{DAG}^{(\ell)}$  which uses  $O\left(\frac{\ell^3 n^3}{\delta^2 \varepsilon^2} \log^2\left(\frac{nk}{\varepsilon\delta}\right) + \frac{\ell^3 nk^{\ell+1}}{\varepsilon} \log^2\left(\frac{nk}{\varepsilon\delta}\right)\right)$  samples and runs in time  $O(\exp(\ell)\text{poly}(n))$  that is *proper* and returns a distribution from  $\mathcal{DAG}^{(\ell)}$ .

In order to prove the above theorem, we will be using the following result from [11] which states that  $\mathcal{DAG}^{(\ell)}$  can be sampled efficiently.

**Theorem F.4** (Theorem 13 of [11] restated). *There exists a randomized algorithm that can sample from  $\mathcal{DAG}^{(\ell)}$  with weight proportional to  $f(G)$  efficiently in expected sampling time  $4^\ell n^{O(1)}$ .*

Note that the above result gives a polynomial time sampling algorithm for  $\mathcal{DAG}^{(\ell)}$  when  $\ell$  is bounded by a constant.

Interestingly, our loss function is also a modular weight function.

**Claim F.5.** *Our loss function  $\exp_{\beta} \log \left( \prod_{s=1}^t P(x^{(s)}) \right)$  is a modular weight function.*

This follows from the fact that the distribution  $P^*$  defined over the DAG  $G$  factorizes: see Equation (1) in Definition A.3. Now we are ready to present the proof of our main result.

*Proof of Theorem F.3.* First note that a vertex cover of the moralized graph  $G_M$  having size  $\leq \ell$  implies that the indegree of  $G$  is  $\leq \ell$ ; if the indegree of  $G$  is  $> d$ , there exists a *clique* in the moralized graph of size  $> d + 1$ , which implies that any vertex cover of the moralized graph must have size  $> d$ .

- (i) We will be using the EWA algorithm (Algorithm 1) for this purpose. In each round of the algorithm, we will be using the algorithm from Theorem F.4. The sample complexity (aka. number of rounds of EWA algorithm) follows from the guarantee of EWA algorithm for learning Bayesian networks with bounded indegree (see Theorem B.11). Since in each round, we will be calling the algorithm corresponding to Theorem F.4, the running time of our algorithm will be  $O(\exp(\ell)\text{poly}(n))$ .
- (ii) We will be calling RWM algorithm (Algorithm 2) here. Similar to the above, in each round, we will be using the algorithm from Theorem F.4. The sample complexity (aka. number of rounds of RWM algorithm) follows from the guarantee of RWM algorithm for learning Bayesian networks (see Theorem B.12). Since in each round, we will be calling the algorithm corresponding to Theorem F.4, the running time of our algorithm will be  $O(\exp(\ell)\text{poly}(n))$ .

□

## G Efficient Maximum Likelihood Estimation

It is well-known that maximizing the likelihood  $\mathbb{E}_{x \sim P^*} P(x)$  is equivalent to minimizing the KL divergence  $D_{\text{KL}}(P^* \| P)$ . This equivalence, in expectation, follows from the definition of KL divergence as  $D_{\text{KL}}(P^* \| P) \triangleq \mathbb{E}_{x \sim P^*} \log \left( \frac{P^*(x)}{P(x)} \right)$ . From a finite-sample PAC distribution learning perspective, we can still say that if  $\hat{P} \in \mathcal{C}$  is a distribution that maximizes the empirical log-likelihood  $\frac{1}{T} \sum_{t=1}^T \log P(x^{(t)})$  — computed using a sufficiently large set of samples  $\{x^{(1)}, \dots, x^{(T)}\} \sim (P^*)^{\otimes T}$ ,  $T \geq \text{Sample-Complexity}(\mathcal{C}, \varepsilon, \delta)$  — over all  $P \in \mathcal{C}$ , then with probability  $\geq 1 - \delta$ , we will have  $D_{\text{KL}}(P^* \| P) \leq \min_{Q \in \mathcal{C}} D_{\text{KL}}(P^* \| Q) + \varepsilon$ . For instance, see Theorem 17 (Appendix G) of [10]. However, if  $\mathcal{N}$  is a class of discretized Bayes nets  $\{(G_1, P_1), \dots, (G_N, P_N)\}$ , even with each DAG structure  $G_i$  endowed with add-on conditional probabilities  $P_i$  as we do in our EWA/RWM based learning-via-sampling approach, *efficiently* finding a  $(G_{i^*}, P_{i^*})$  that maximizes the empirical log-likelihood over such a class is not a trivial task.

By the Bayes net factorization property, the empirical log-likelihood of each  $G_i$  does decompose nicely; as  $\sum_{t=1}^T \log P_i(x^{(t)}) = \sum_{v=1}^n \sum_{t=1}^T \log P_{i,v}(x_v^{(t)} | x_{\text{pa}_{G_i}(v)}^{(t)})$ . However, maximizing  $\sum_t \log P_{i,v}(x_v^{(t)} | x_S^{(t)})$  over all  $S \subseteq N(v)$  with  $|S| \leq d$ , for each node  $v$  (independently), does not suffice since this may result in the formation of cycles in the final digraph. Note that this is not an issue with tree-skeletons (tree-structured or polytree-structured distributions).

Another issue with independent maximization is that the final structure  $G$  needs to be consistent with the skeleton and with the choice of parents for each node (each edge of the skeleton can only be oriented in one direction), which would need to be kept track of in the maximization algorithm.

We claim that the dynamic programming approaches developed for weighted-counting/sampling of exponentially-large classes of polytree and chordal-structured DAGs can be adapted to efficiently compute the maximum likelihood as well. For polytree distributions, we can root the skeleton

$G$  arbitrarily and recursively compute  $T[v, P, t]$  for each “subtree”  $T_v$  rooted at vertex  $v$ , where  $T[v, P, t]$  gives the maximum empirical likelihood ( $t$ -sample) over all possible DAG structures (with add-one probabilities) of  $T_v$  such that vertex  $v$  has fixed parents  $P$ , for all  $P \subseteq N(v)$  with  $|P| \leq d$ . In the base case (only single edge), we can brute-force. We can then use the recurrence  $T[v, P, t] = \sum_{v_1, \dots, v_k \text{ "children" of } v} \sum_{P_1, \dots, P_k \text{ consistent with } P \text{ and } G} T[v_i, P_i, t]$  to compute a maximum-likelihood acyclic orientation of  $G$ . This is similar to our proposed weighted-counting DP, replacing the weighted sum with maximum empirical likelihood. For chordal-structured Bayes nets, we can do a similar adaptation of the CountChordalDist algorithm (Algorithm 5); still recursively computing the DP table bottom-up from the clique-tree decomposition, orienting the “link” of each clique to prevent cycles while getting “independence” of orientations of child-subtrees, etc.

Finally, we remark on the sample complexity of this approach. From e.g., [10, Theorem 17], which uses a Hoeffding bound on the difference between the empirical and the true log-likelihood, we obtain a sample complexity of  $\tilde{O}\left(\frac{\log^2(1/\tau) \log |C|}{\varepsilon^2}\right)$  for proper realizable learning in KL divergence (with constant error probability) via maximum likelihood, and the analysis can be adapted to agnostic learning as well. This matches our sample complexity bounds via RWM-regret up to log factors. For improper learning in the realizable case, our EWA-regret approach gives a better, near-optimal, sample complexity, as has been discussed before.

## References

- [1] Ivona Bezáková and Wenbo Sun. Counting and sampling orientations on chordal graphs. In *International Conference and Workshops on Algorithms and Computation*, pages 352–364. Springer, 2022.
- [2] Arnab Bhattacharyya, Sutanu Gayen, Eric Price, Vincent YF Tan, and NV Vinodchandran. Near-optimal learning of tree-structured distributions by chow and liu. *SIAM Journal on Computing*, 52(3):761–793, 2023.
- [3] Carl Wilhelm Borchardt. Ueber eine der interpolation entsprechende darstellung der eliminations-resultante. *Journal für die reine und angewandte Mathematik*, 57:111–121, 1860.
- [4] Johannes Brustle, Yang Cai, and Constantinos Daskalakis. Multi-item mechanisms without item-independence: Learnability via robustness. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 715–761, 2020.
- [5] Clément L Canonne, Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Testing bayesian networks. In *Conference on Learning Theory*, pages 370–448. PMLR, 2017.
- [6] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.
- [7] Seth Chaiken. A combinatorial proof of the all minors matrix tree theorem. *SIAM Journal on Algebraic Discrete Methods*, 3(3):319–329, 1982.
- [8] Patrick De Leenheer. An elementary proof of a matrix tree theorem for directed graphs. *SIAM Review*, 62(3):716–726, 2020.
- [9] Luc Devroye and Gábor Lugosi. *Combinatorial Methods in Density Estimation*. Springer New York, NY, 2001.
- [10] Jon Feldman, Ryan O’Donnell, and Rocco A. Servedio. Learning mixtures of product distributions over discrete domains. *SIAM Journal on Computing*, 37(5):1536–1564, 2008.
- [11] Juha Harviainen and Mikko Koivisto. Revisiting bayesian network learning with small vertex cover. In *Uncertainty in Artificial Intelligence*, pages 819–828. PMLR, 2023.
- [12] Donald J Rose, R Endre Tarjan, and George S Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.
- [13] Wenbo Sun. *Efficient Sampling and Counting of Graph Structures Related to Chordal Graphs*. Rochester Institute of Technology, 2022.

- [14] William Thomas Tutte. *Graph theory*, volume 21. Cambridge university press, 2001.
- [15] Yannis G. Yatracos. Rates of Convergence of Minimum Distance Estimators and Kolmogorov's Entropy. *The Annals of Statistics*, 13(2):768 – 774, 1985.