Parameter estimation for integer-valued Gibbs distributions

David G. Harris
University of Maryland, Department of Computer Science
davidgarris29@gmail.com

Vladimir Kolmogorov
Institute of Science and Technology Austria
vnk@ist.ac.at

Abstract

We consider Gibbs distributions, which are families of probability distributions over a discrete space $\Omega$ with probability mass function given by $\mu_\beta^\Omega(x) = \frac{e^{\beta H(x)}}{Z(\beta)}$. Here $H : \Omega \to \{0, 1, \ldots, n\}$ is a fixed function (called a Hamiltonian), $\beta$ is the parameter of the distribution, and the normalization factor $Z(\beta) = \sum_{x \in \Omega} e^{\beta H(x)}$ is called the partition function. We study how function $Z(\cdot)$ can be estimated using an oracle that produces samples $x \sim \mu_\beta^\Omega(\cdot)$ for a value $\beta$ in a given interval $[\beta_{\min}, \beta_{\max}]$.

We consider the problem of estimating the normalized coefficients $c_k$ for indices $k \in K$ satisfying

$$\max_{\beta} \mu_\beta^\Omega(\{x \mid H(x) = k\}) \geq \mu_*, \quad \text{where } \mu_* \in (0, 1)$$

We solve this using $	ilde{O}(\frac{\min\{q, n^2\}}{\epsilon^2})$ samples, where $q = \log \frac{Z(\beta_{\max})}{Z(\beta_{\min})}$, and we show this is optimal up to logarithmic terms.

As a key subroutine, we show how to estimate quantities $q$ using $	ilde{O}(\frac{1}{\epsilon^2})$ samples. This improves over a prior algorithm of Kolmogorov (2018) which uses $O(\frac{q^2}{\epsilon^2})$ samples. We also show a “batched” version of this algorithm which simultaneously estimates $\frac{Z(\beta)}{Z(\beta_{\min})}$ for many values of $\beta$, at essentially the same cost as for estimating just $\frac{Z(\beta_{\max})}{Z(\beta_{\min})}$ alone. We show matching lower bounds, demonstrating that this complexity is optimal as a function of $n$ and $q$ up to logarithmic terms.

1 Introduction

Given a real-valued function $H(\cdot)$ over some finite set $\Omega$, the Gibbs distribution is defined as the family of distributions $\{\mu_\beta^\Omega\}$ over $\Omega$ parameterized by $\beta$, where

$$\mu_\beta^\Omega(x) = \frac{e^{\beta H(x)}}{Z(\beta)}$$

These distributions frequently occur in physics, where the parameter $-\beta$ corresponds to the inverse temperature, the function $H(x)$ is called the Hamiltonian of the system, and the normalizing constant $Z(\beta) = \sum_{x \in \Omega} e^{\beta H(x)}$ is called the partition function. They also occur in a number of applications of computer science, particularly sampling and counting algorithms.

In this paper we consider a restricted form of the Gibbs distributions, where $H(\Omega)$ takes on integer values in the range $H \overset{\text{def}}{=} \{0, 1, \ldots, n\}$ for some known integer $n$. If we set $c_k = |\{x \in \Omega : H(x) = k\}|$, then the partition function can be written as

$$Z(\beta) = \sum_{k \in H} c_k e^{\beta k}$$

and we can also define the associated probability density $\mu_\beta$ over $k \in H$ by:

$$\mu_\beta(k) = \frac{c_k e^{\beta k}}{Z(\beta)}$$
The basic problem we consider is how to estimate various parameters of the Gibbs distribution, given access to an oracle which can return a sample from the distribution for any chosen parameter \( \beta \in [\beta_{\min}, \beta_{\max}] \). Here the Gibbs distribution may be available as some physical process, in which case the oracle is an experimental run, or it may be available as some computational subroutine. Specifically, we seek to estimate the following parameters:

1. The coefficients \( c_k \) (suitably normalized)

2. The ratio \( Q_\beta = \frac{Z(\beta)}{Z(\beta_{\min})} \) for given values of \( \beta \)

These parameters \( q = \log Q_{\beta_{\max}} \) and \( c_k \) are correlated with underlying system parameters in a number of applications. For instance, Komura and Okabe \[16\] propose a method essentially gives full information about the system, and allows computing various physically relevant quantities such as entropy, free energy, etc. For example, \( c_0 \) is proportional to \( \log Z(\beta) \), \( c_k \) is of fundamental importance in statistical physics. The knowledge of these coefficients, usually called (discrete) density of states (DOS), essentially gives full information about the system, and allows computing various physically relevant quantities such as entropy, free energy, etc. For example, Komura and Okabe \[16\] propose a method for studying first-order phase transitions based on analyzing the curve \( \log c_k \) as a function of \( k \): such transition exists if the curve has a non-convex region, and the behavior of the curve in such region determines the temperature of the phase transition.\[1\]

One special case of the Gibbs distribution appears in a number of important combinatorial applications and is worth further mention: the situation where the coefficients \( c_0, \ldots, c_k \) are log-concave, that is, they satisfy the bound \( c_k^2 \geq c_{k-1}c_{k+1} \) for \( k = 1, \ldots, n - 1 \), as well as that bound that the non-zero coefficients form an interval, i.e. there is no triple of indices \( i_0 < i_1 < i_2 \) with \( c_{i_0} > 0, c_{i_1} = 0, c_{i_2} > 0 \). (Such sequences are also referred to as Pólya frequency sequences of order 2.) We refer to this as the log-concave setting, and a number of results will be specialized for this case. We refer to the situation where coefficients \( c_k \) are not restricted to be log-concave as the general setting.

Before we state our results, let us state some basic definitions and background assumptions. “Sample complexity” refers to the number of calls to the sampling oracle. We always assume for brevity that \( \epsilon < \epsilon_{\max}, n \geq 2, q \geq q_{\min} \) for some constants \( q_{\min} > 1, \epsilon_{\max} > 0 \). The algorithms also apply when \( n = 2 \) or \( q \in (0, q_{\min}) \), but the upper bound on sample complexity will be at most that of the case \( q = q_{\min}, n = 2 \). We define \( \mu_\beta(X) \) for an arbitrary set \( X \subseteq \mathbb{R} \) by \( \mu_\beta(X) = \mu_\beta(X \cap \mathcal{H}) \).

Applications and related work  Estimating coefficients \( c_k \) is of fundamental importance in statistical physics. The knowledge of these coefficients, usually called (discrete) density of states (DOS), essentially gives full information about the system, and allows computing various physically relevant quantities such as entropy, free energy, etc. For example, Komura and Okabe \[16\] propose a method for studying first-order phase transitions based on analyzing the curve \( \log c_k \) as a function of \( k \): such transition exists if the curve has a non-convex region, and the behavior of the curve in such region determines the temperature of the phase transition.\[1\]

One of the most popular methods in computational physics for estimating \( \mathbf{c} = (c_0, \ldots, c_n) \) is the Wang-Landau (WL) algorithm \[20\] and its variants, such as the \( 1/t \)-WL algorithm \[3\]. As discussed in \[17\], there are more than 1500 papers on the application of the algorithm and its improvements. The method performs a random walk on \( \Omega \), and maintains current estimates \( \hat{\mathbf{c}} \) of \( \mathbf{c} \). At each step it makes a random move according to a Metropolis-Hastings Markov Chain with the stationary distribution proportional to \( \frac{1}{\hat{\mathcal{H}}(x)} \). Note, if \( \hat{\mathbf{c}} = \mathbf{c} \) then sampling \( x \sim \pi(\cdot|\hat{\mathbf{c}}) \) and taking \( k = H(x) \) will produce a uniform measure over \( \mathcal{H} \). It then updates estimates \( \hat{\mathbf{c}} \) based on the past execution history.

Despite successful applications for many problems of practical interest, many open question remain about the efficiency and dynamics of WL algorithms. Some variants have guaranteed convergence properties \[7\], but the rates of convergence and accuracy of approximation are not known, to our knowledge. Note that distributions \( \pi(x|\hat{\mathbf{c}}) \) used in WL in general are not Gibbs distributions, so fast mixing times proved for some Gibbs distributions (e.g. ferromagnetic Ising model) may not generalize to WL algorithms.

---

\[1\] We remark that the method for the log-concave setting developed in this paper is potentially applicable to this problem: one could apply this method assuming that coefficients \( c_k \) are log-concave, and after obtaining the estimates verify the assumption using additional samples. However, we leave such extension outside the scope of this paper.
The problem of estimating coefficients also has relevance in computer science. For example, it can be used to approximate the number of combinatorial objects such as the connected subgraphs and matchings of different sizes in a given graph. Details are discussed in the next section.

1.1 Our contribution

The main problem we consider is to estimate the coefficients of the Gibbs function. For this we need some preliminary definitions. For any \( k \in \mathcal{H} \), define

\[
\Delta(k) = \max_{\beta \in [\beta_{\min}, \beta_{\max}]} \mu_\beta(k)
\]

Given a parameter \( \mu_* \) and a set \( \mathcal{K} \subseteq \mathcal{H} \), let us define

\[
\mathcal{K}^* = \{ k \in \mathcal{K} | \Delta(k) \geq \mu_* \}
\]

For a set \( \mathcal{K} \subseteq \mathcal{H} \), we say that non-negative vector \( \hat{c} \) is an \((\varepsilon, \mathcal{K})\)-estimate of vector \( c \) if two conditions hold: (i) \( \hat{c}_k > 0 \) for all \( k \in \mathcal{K} \); (ii) for all pairs \( k, \ell \in \mathcal{H} \) with \( c_k > 0, c_\ell > 0 \), we have \( \frac{\hat{c}_k}{c_k} \in [e^{-\varepsilon}, e^\varepsilon] \).

We can now state the problem \( P_{\mu_*}^{\mathcal{K}} \): Given \( \varepsilon, \mu_* > 0 \), as well as set \( \mathcal{K} \subseteq \mathcal{H} \), compute values \( \{\hat{c}_k\}_{k \in \mathcal{H}} \) such that \( \hat{c} \) is an \((\varepsilon, \mathcal{K}^*)\)-estimate of \( c \). Note here that we may not know the precise set \( \mathcal{K}^* \), and in solving the problem \( P_{\mu_*}^{\mathcal{K}} \) we may be provided some estimated coefficients \( \hat{c}_k \) for \( k \in \mathcal{K} - \mathcal{K}^* \).

For a set \( \mathcal{K} \subseteq \mathcal{H} \), we define \( \text{span}(\mathcal{K}) = 1 + \max \mathcal{K} - \min \mathcal{K} \). We prove the following main result.

**Theorem 1.** Let \( h = \text{span}(\mathcal{K}) \) and let \( r = \min\{\sqrt{q \log h}, q, |\mathcal{K}|\} \). There exists an algorithm to solve \( P_{\mu_*}^{\mathcal{K}} \) with success probability \( 1 - \gamma \) and expected sample complexity

\[
O\left(\log n + \frac{r \log |\mathcal{K}|}{\varepsilon^2 \mu_*} + \min\left\{\frac{q \log h \log \frac{r}{\varepsilon^2}}{\mu_*}, \frac{h^2 \Gamma_h \log \frac{r}{\varepsilon^2}}{\varepsilon^2} + h \log q\right\}\right)
\]

where \( \Gamma_h = \Theta(\log h) \) in the general setting and \( \Gamma_h = \Theta(1) \) in the log-concave setting.

In addition to this algorithm, we provide a specialized algorithm for the log-concave setting, which has substantially improved complexity in most cases. We summarize it as follows:

**Theorem 2.** Let \( h = \text{span}(\mathcal{K}) \). In the log-concave setting, there exists an algorithm to solve \( P_{\mu_*}^{\mathcal{K}} \) with success probability \( 1 - \gamma \) and expected sample complexity

\[
O\left(\log n + \min\left\{\frac{(h^2 + \frac{1}{\mu_*}) \log h}{\varepsilon^2}, h \log q, \frac{(q \log h + h + \frac{1}{\mu_*}) \log h}{\varepsilon^2} + h \log^2 h\right\}\right)
\]

Note that these sample complexities are nearly independent of \( n \) (aside from a single \( \log n \) term), depending instead on properties of the estimated set \( \mathcal{K} \).

The bounds of Theorem 1 and Theorem 2 are complex and have many different cases. The following formulas have slightly worse logarithmic terms but are significantly easier to read;

**Corollary 3.** Let \( h = \text{span}(\mathcal{K}) \). In the general setting, there exists an algorithm to solve \( P_{\mu_*}^{\mathcal{K}} \) with success probability \( 1 - \gamma \) and expected sample complexity

\[
O\left(\frac{\log q}{\varepsilon^2} \cdot \left(\min\{q, h^2\} \log h + \min\{\sqrt{q \log h}, |\mathcal{K}|\}\right)\right)
\]

In particular, for \( \mathcal{K} = \mathcal{H} \), there is an algorithm to solve \( P_{\mu_*}^{\mathcal{H}} \) with expected sample complexity

\[
O\left(\frac{\log q}{\varepsilon^2} \cdot \min\left\{\frac{q \log n}{\mu_*}, \frac{n^2 \log n + \frac{n}{\mu_*}}{\varepsilon^2}\right\}\right)
\]
If \( h = O(1) \), then there exists an algorithm to solve \( P^{\mu_* K}_\text{coef} \) with expected sample complexity
\[
O\left( \log(nq) + \frac{\log \frac{1}{\varepsilon}}{\varepsilon^2 \mu_*} \right)
\]

In the log-concave setting, there exists an algorithm to solve \( P^{\mu_* K}_\text{coef} \) with expected sample complexity
\[
O\left( \frac{\log nq}{\varepsilon^2} \cdot \left( 1/\mu_* + \min\{h^2, q \log h + h\} \right) \right)
\]

We also show a lower bound for \( P^{\mu_* \gamma}_\text{coef} \) of \( \Omega\left( (\min\{q,n^2\} + \min\{q^2,n\}) \log \frac{1}{\varepsilon} \right) \) for the general setting and \( \Omega\left( (\min\{q,n^2\} + \min\{q^2,n\}) \log \frac{1}{\varepsilon^2} \right) \) for the log-concave setting. In the general case, this matches our algorithm up to logarithmic factors in \( n \) and \( q \). In the log-concave case, there is an additional additive discrepancy between the upper and lower bounds of order \( \tilde{O}(n/\varepsilon^2) \) in the regime when \( 1/\mu_* + q = o(n) \).

To our knowledge, problem \( P_\text{coef} \) has not been studied yet in its general form, even though coefficient estimation is quite important e.g. in physics (as indicated by the widespread use of the Wang-Landau algorithm discussed earlier). As two concrete applications, we obtain faster algorithms to approximate the number of connected subgraphs and number of matchings in a given graph.

**Theorem 4.** Let \( G = (V,E) \) be a connected graph and for \( i = |V| - 1, \ldots, |E| \) let \( N_i \) denote the number of connected subgraphs of \( G \) with \( i \) edges. There is an FPRAS for the sequence \( N_i \) with time complexity \( O\left( |E|^3 |V| \log^3 |E| \right) \).

**Theorem 5.** Let \( G = (V,E) \) be a graph with \( |V| = 2v \) and for \( i = 0, \ldots, v \) let \( M_i \) denote the number of matchings in \( G \) with \( i \) edges. Suppose \( M_0 > 0 \) and \( M_{v-1}/M_v \leq f \) for a known parameter \( f \). There is an FPRAS for the sequence \( M_i \) running in time \( \tilde{O}(|E||V|^3 f/\varepsilon^2) \).

In particular, if \( G \) has minimum degree at least \( |V|/2 \), then there is an FPRAS for the sequence \( M_i \) with time complexity \( \tilde{O}(|V|^7/\varepsilon^2) \).

**Theorem 4** improves by a factor of \( |V| \) compared to the FPRAS for counting matchings given in [13]. While other FPRAS algorithms for counting connected subgraphs have been proposed by [21, 2], the runtime appears to be very large (and not specifically stated in those works); thus **Theorem 4** appears to be the first potentially practical algorithm for this problem.

As a key subroutine for \( P_\text{coef} \), we also develop new algorithms to estimate the partition function \( Z(\beta) \). It will be critical to compute \( Z(\beta) \) for many values of \( \beta \) simultaneously, which allows us to amortize many precomputation steps. We consider a problem we refer to as \( P^{\beta}_\text{ratio} \): Given \( \varepsilon > 0 \) and a set of values \( \mathcal{B} \subseteq [\beta_{\min}, \beta_{\max}] \), compute values \( \bar{Q}_\beta > 0 \) such that \( \bar{Q}_\beta/Q_\beta \) are \( \varepsilon^{-1} \) for all \( B \in \mathcal{B} \).

Previous algorithms for \( P^{\beta}_\text{ratio} \) had focused on the case where \( \mathcal{B} = \{\beta_{\max}\} \); we denote this special case by simply \( P^{\beta}_\text{ratio} \). Algorithms for \( P^{\beta}_\text{ratio} \) with steadily improving expected sample complexities have been proposed by several authors [4, 18, 15]. The best prior algorithm, due to Kolmogorov [15], had expected sample complexity \( O\left( \frac{\alpha^2 \log n}{\varepsilon^2} \right) \). We improve on these results as follows.

**Theorem 6.** There is an algorithm to solve \( P^{\beta}_\text{ratio} \) with success probability \( 1 - \gamma \) and expected sample complexity
\[
O\left( \min\left\{ \frac{|\mathcal{B}| + q \log n \log |\mathcal{B}|}{\varepsilon^2}, \frac{|\mathcal{B}| |n + n^2| \Gamma \log |\mathcal{B}|}{\varepsilon^2} + n \log q \right\} \right)
\]
where \( \Gamma = \Theta(\log n) \) in the general setting and \( \Gamma = \Theta(1) \) in the log-concave setting.

When \( \beta = \{\beta_{\max}\} \), this gives us an improved algorithm for the classical \( P^{\beta}_\text{ratio} \) problem:

**Corollary 7.** There is an algorithm to solve \( P^{\beta}_\text{ratio} \) with expected sample complexity
\[
O\left( \min\left\{ \frac{q \log n \log 1}{\varepsilon^2}, \frac{n^2 \Gamma \log 1}{\varepsilon^2} + n \log q \right\} \right)
\]
where \( \Gamma = \Theta(\log n) \) in the general setting and \( \Gamma = \Theta(1) \) in the log-concave setting.
In most known applications \( q \log n \ll n^2 \) and so Corollary 7 has the same complexity as the prior algorithm of Kolmogorov [15]. However, we still find Theorem 8 significant due to the following reasons:

- We will show a lower bound of \( \Omega\left( \frac{\min(q,n^2)}{\varepsilon^2} \right) \) for \( P_{\text{ratio}} \), even in the log-concave setting. Thus, our algorithm is optimal up to logarithmic factors, and this result essentially settles the complexity for \( P_{\text{ratio}} \) as functions of \( n \) and \( q \).
- The new algorithm, and in particular its dependence upon \( n \), is critical in order to get sample complexities for Theorems 1 and 2 which depend primarily on \( h = \text{span}(\mathcal{K}) \).
- Theorem 4 shows that the batched problem \( P_{\text{ratio}}^B \) can be solved essentially with no increase in complexity as long as \( B \) is not too large (specifically, as long as \( |B| \leq \min\{q \log n, n\} \)).

1.2 Algorithm overview

Section 2 summarizes the subroutines and data structures that we will need for our algorithms, and Section 3 summarizes the main algorithm to solve \( P_{\text{coef}} \). Before we describe the technical details, let us provide a high-level roadmap. For simplicity, we ignore edge cases, e.g. when the optimum of \( \max_{\beta \in [\beta_{\min},\beta_{\max}]} \mu_\beta(k) \) for a given \( k \) is attained at \( \beta_{\min} \) or \( \beta_{\max} \). We also assume that the tasks need to be solved with some constant probability of success (it can then be boosted to the desired probability with some standard techniques that add an extra logarithmic factor to the complexity).

Our plan to solve \( P_{\text{coef}}^\mathcal{K} \) is to estimate the values \( \frac{\Theta}{Z(\beta_{\min})} \) for \( k \in \mathcal{K} \), instead of directly estimating ratios \( \frac{\Theta}{e_k} \) for \( k, \ell \in \mathcal{K} \). Clearly, solving the former problem with accuracy \( \varepsilon/2 \) will solve the latter one with accuracy \( \varepsilon \) (where accuracy means relative error). We have \( \frac{\Theta_k}{Z(\beta_{\min})} = \frac{\Theta_k}{Z(\beta)} \cdot \frac{Z(\beta)}{Z(\beta_{\min})} = e^{-\beta k} \cdot \mu_\beta(k) \cdot Q_\beta \) for any \( \beta \). So this can be reduced to two subproblems, which we discuss next:

(A) estimate \( \mu_\beta(k) \) for some \( \beta \in [\beta_{\min},\beta_{\max}] \);
(B) estimate \( Q_\beta \) for the value of \( \beta \) chosen in step (A).

**Solving (A) for single value of \( k \):** To estimate \( \mu_\beta(k) \) with accuracy \( \varepsilon \) for given \( k \) and \( \beta \), we can draw \( \Theta(\frac{1}{\mu_{\beta}(k)}) \) samples of the Gibbs distribution with parameter \( \beta \) and compute the empirical frequency of the value of \( k \). To get good complexity, we thus need to find value \( \beta \) for which \( \mu_\beta(k) \) is sufficiently large. We make the following observation: if \( \mu_\beta([0,k]) \approx \mu_\beta([k+1,n]) \), or more precisely \( \min\{\mu_\beta([0,k]),\mu_\beta([k+1,n])\} = \Theta(1) \), then \( \mu_\beta(k) \approx \Omega(\max_\beta \mu_\beta(k)) \). The latter value is at least \( \Omega(\mu^* ) \) for \( k \in \mathcal{K}^* \). Therefore, we can do the following: (i) Use a binary search to find value \( \beta \in [\beta_{\min},\beta_{\max}] \) with \( \mu_\beta([0,k]) \approx \mu_\beta([k+1,n]) \); and (ii) Estimate \( \mu_\beta(k) \) using \( O(\frac{1}{\mu_{\beta}\varepsilon^2}) \) samples.

We remark that this binary search is somewhat delicate, since the interval \([\beta_{\min},\beta_{\max}]\) may be unbounded, and since we can only approximate the values \( \mu_\beta([0,k]) \) and \( \mu_\beta([k+1,n]) \) (by sampling).

The next question is how to solve (A) for multiple values of \( k \). To get good complexity, we need to reuse the same value of \( \beta \) for multiple \( k \)'s. Below we discuss two methods for doing that.

**Method A1 to solve (A) for multiple \( k \):** First, we find value \( \beta \) with \( \mu_\beta([0,k]) \approx \mu_\beta([k+1,n]) \) for the largest index \( k \) in \( \mathcal{K} \), as described above. By inspecting empirical frequencies of distribution \( \mu_\beta \), we then find smallest \( k' \) such that \( \mu_\beta([0,k']) \approx \mu_\beta([k'+1,n]) \) still holds. We remove \([k',k] \) from \( \mathcal{K} \), and repeat the procedure until \( \mathcal{K} \) becomes empty. The formal algorithm is called \textit{FindRepresentatives} and is described in Section 5. By construction, this outputs set \( R \subseteq [\beta_{\min},\beta_{\max}] \) such that for every \( k \in \mathcal{K}^* \) there exists \( \beta \in R \) such that \( \mu_\beta(k) \geq \Omega(\mu^* ) \). We call such \( R \) a \textit{representative set}. Crucially, we show that \( R \) has size at most \( O(\sqrt{q \log n}) \), regardless of \( \mathcal{K} \). A more precise estimate is \( |R| = O(\min\{\sqrt{q \log n}, q, |\mathcal{K}|\}) \).
To summarize, this approach solves problem (A) for all $k \in \mathbb{K}$ using $O\left(\frac{|R|}{\mu \cdot \varepsilon^2}\right)$ samples, plus the samples needed for $|R|$ binary searches.

**Method A2 to solve (A) for multiple $k$:** In the log-concave setting we develop another technique which is more efficient in most scenarios. The log-concavity implies that for a fixed $\beta$ distribution $\mu_\beta(k)$ is log-concave as a function of $k$, and in particular $\mu_\beta(k) \geq \min\{\mu_\beta(k^-), \mu_\beta(k^+\right)$ if $k \in [k^-, k^+] \subseteq \mathbb{H}$. Thus, a single value of $\beta$ will "cover" some interval $[k^-, k^+]$. Our goal will be to find sequence $\beta_0, \beta_1, \ldots, \beta_t$ with $(\beta_0, \beta_t) = (\beta_{\min}, \beta_{\max})$ such that value $\beta_i$ covers interval $[k_i^-, k_i^+]$ and there are no "gaps", i.e. $k_i^+ = k_{i+1}^-$ for all $i$. Such sequence will be called a schedule. Let us denote $w_i = \min\{\mu_\beta(k_i^-), \mu_\beta(k_i^+\right)$ ("weight" of $i$). We no longer require that $\mu_\beta([0, k]) \approx \mu_\beta([k+1, n])$ for $k \in [k_i^-, k_i^+]$; instead, we aim to find a schedule so that the sum $\sum_i \frac{1}{w_i}$ (called inverse weight) is small, since this quantity will determine the final complexity. We show that there exists a schedule with inverse weight $O(n)$. (This is optimal since we may have $\mu_\beta(k) = \frac{1}{n\cdot 1}$ for all $\beta, k$.) Given such schedule, problem (A) can be solved using $O\left(\frac{n}{\varepsilon^2} + \frac{1}{\mu \cdot \varepsilon^2}\right)$ samples, by drawing $\Theta\left(\frac{1}{w_i \varepsilon^2}\right)$ samples at $\beta_i$ and $\Theta\left(\frac{1}{\mu \varepsilon^2}\right)$ samples at $\beta_{\min}$ and $\beta_{\max}$.

The algorithm for computing a schedule is described in Section 7 technically, this is the most involved part of the paper. Here we just describe some key ideas. First, we relax constraint $k_i^+ = k_{i+1}^-$ to $k_i^+ \geq k_{i+1}$; this can be easily fixed in a postprocessing step. Also, we only use intervals satisfying $\frac{1}{w_i} \leq \Theta\left(k_i^+ - k_i^-\right)$. By throwing away redundant intervals we can make sure that each $k$ is covered by at most two intervals; this will imply the bound of $O(n)$ on the inverse weight of the schedule.

The algorithm tries to "fill gaps", i.e. make sure that each pair of consecutive integers $(k, k+1)$ is covered by an interval. In each iteration we pick such gap and use binary search to find $\beta_i$'s in a given set $B \subseteq [\beta_{\min}, \beta_{\max}]$. This allows to adapt the method for constructing a schedule also works in the general setting (with a slightly worse bound on the inverse weight, namely $O(n \log n)$ instead of $O(n)$).

**Solving (B):** Next, we discuss how to solve problem (B) for multiple values of $\beta$, i.e. estimate ratios $Q_\beta = \frac{Z(\beta)}{Z(\beta_{\min})}$ for all $\beta$’s in a given set $B \subseteq [\beta_{\min}, \beta_{\max}]$. Note that this is precisely problem $P_{\text{ratio}}^B$.

**Method B1:** One possibility is based on the method in [15] that estimates $Q_\beta$ for a single value $\beta$ using $O\left(\frac{n^2 \log n}{\varepsilon^2}\right)$ samples. This method works by constructing a cooling schedule $\alpha = (\alpha_0, \ldots, \alpha_t)$ with $(\alpha_0, \alpha_t) = (\beta_{\min}, \beta)$ that has a small “curvature” $\kappa(\alpha)$ (see Appendix D for the definition of $\kappa$). We observe that $\kappa(\alpha_0, \ldots, \alpha_t, \beta) \leq \kappa(\alpha_0, \ldots, \alpha_t, \alpha_{t+1}, \ldots, \alpha_t)$ if $\beta \in (\alpha_t, \alpha_{t+1}]$. This allows to adapt the method to the case when $|B| > 1$ as follows: (i) construct a cooling schedule $\alpha$ for value $\beta = \beta_{\max}$; (ii) for each $\beta \in B$ find index $i$ with $\beta \in (\alpha_i, \alpha_{i+1}]$, and estimate $Q_\beta$ with the cooling schedule $(\alpha_0, \ldots, \alpha_i, \beta)$. The structure of the algorithm means that most of the samples can be reused. We get complexity $O\left(\frac{|B| + n \log n}{\varepsilon^2}\right)$ (times an extra logarithmic factor to boost the probability of success).

**Method B2:** We can use the schedule, as constructed in method (A2), to get a new algorithm which is more efficient than (B1) when $q \gg n^2$. Consider consecutive values $\beta_i, \beta_{i+1}$ of the schedule and integer $k = k_i^+ = k_{i+1}^-$. It can be checked that $Q_\beta = \frac{\mu_\beta(k)}{\mu_{\beta+1}(k)} e^{\beta_i - \beta_{i+1}} k$. By construction, values $\mu_\beta(k)$ and $\mu_{\beta+1}(k)$ are sufficiently large, and so $Q_\beta = \frac{\mu_\beta(k)}{\mu_{\beta+1}(k)} e^{\beta_i - \beta_{i+1}} k$ can be estimated efficiently by sampling at $\beta_i$ and $\beta_{i+1}$ and observing the empirical frequency of $k$. We then choose the total number of samples of
our algorithm so that telescoping products $\prod_i Q_{\beta_i}^{k_i}$ are estimated to within overall accuracy $\varepsilon$. This shows how to estimate $Q_{\beta_i}$ for values $\beta_i$ in the schedule. To estimate $Q_{\beta}$ for values $\beta \in \mathcal{B}$, we insert $\mathcal{B}$ into the sequence $(\beta_0, \ldots, \beta_t)$ and then use an approach from the previous paragraph.

Wrap-up: In summary, we have described two techniques for solving problem (A) and two techniques for solving problem (B). Taking different combinations yields $2 \times 2 = 4$ methods for solving $P^{\mu_{\text{coef}}}_{\text{coef}}$ in the log-concave setting and $1 \times 2 = 2$ methods for solving $P^{\mu_{\text{coef}}}_{\text{coef}}$ in the general setting. This explains rather complicated expressions for the complexities given in Theorems 1 and 2. The most practical method is probably combination (A2,B1) for solving $P^{\mu_{\text{coef}}}_{\text{coef}}$ in the log-concave setting. Indeed, this combination will be used for the algorithms developed in Section 8 for approximate counting of matching and connected subgraphs.

We remark that methods (A2) and (B2) outlined above have at least linear dependence on $n$, while Theorems 1 and 2 claim complexities that depend mainly on $h = \text{span}(\mathcal{K})$ (which can be much smaller than $n$). We achieve this via the following “black-box” reduction. Suppose for simplicity that $\mathcal{K} = \{0, \ldots, h\}$. First, we use binary search to find value $\alpha$ with $\mu_{\alpha}(\mathcal{K}) \approx \mu_{\alpha}(\mathcal{H} - \mathcal{K})$. Then we consider a new problem in which the input range is $[\beta_{\text{min}}, \alpha]$ and coefficients $c_k$ for $k \in \mathcal{H} - \mathcal{K}$ are set to zero (so parameter $n$ for the new problem effectively becomes $h$). We use rejection sampling to simulate sampling oracle $\mu_{\beta}(\cdot)$ for the new problem. For $\beta \in [\beta_{\text{min}}, \alpha]$, our choice of $\alpha$ ensures that we will throw away only a constant fraction of samples. Further details are given in Section 3.2.

1.3 Computational extensions

For the most part, we focus on the sample complexity, i.e. the number of calls to the Gibbs distribution oracle. There are two mild extensions of this framework worth further discussion.

Computational complexity. The oracle may actually be provided as a randomized sampling algorithm. This is the situation, for example, in our applications to counting connected subgraphs and matchings. In this case we also need to bound our algorithm’s computational complexity. In all the algorithms we develop, the time complexity will be a small logarithmic factor times the query complexity. The cost of the oracle will be typically be much larger than this overhead. Thus, all our sampling procedures translate directly into efficient sampling algorithms, whose runtime is the expected sample complexity multiplied by the computational cost of the oracle. We will not comment explicitly on time complexity henceforth.

Approximate sampling oracles. Many applications have only approximate sampling oracles $\tilde{\mu}_{\beta}$, that are close to $\mu_{\beta}$ in terms of the variation distance $|| \cdot ||_{TV}$ defined via

$$
\delta = ||\tilde{\mu}_{\beta} - \mu_{\beta}||_{TV} = \max_{K \subseteq \mathcal{H}} |\tilde{\mu}_{\beta}(K) - \mu_{\beta}(K)| = \frac{1}{2} \sum_{k \in \mathcal{H}} |\tilde{\mu}_{\beta}(k) - \mu_{\beta}(k)|
$$

By a standard coupling trick (see e.g. [18, Remark 5.9]), Theorems 6 and 2 remain valid if exact oracles are replaced with approximate oracles satisfying $||\tilde{\mu}_{\beta} - \mu_{\beta}||_{TV} \leq O(\gamma/T)$ where $T$ is the sample complexity of the corresponding algorithm. In particular, we have the following result; for completeness, we give a proof in Appendix A.

Theorem 8. Suppose that algorithm $\mathcal{A}$ has expected sample complexity $T$ and, suppose for some condition $C$ and value $\gamma > 0$ we have $\mathbb{P}[\text{output of } \mathcal{A} \text{ satisfies } C] \geq 1 - \gamma$. Let $\mathcal{A}'$ be the algorithm obtained from $\mathcal{A}$ by replacing calls $k \sim \mu_{\beta}$ with calls $k \sim \tilde{\mu}_{\beta}$ where $\tilde{\mu}_{\beta}$ is a distribution over $\mathcal{H}$ satisfying $||\tilde{\mu}_{\beta} - \mu_{\beta}||_{TV} \leq \gamma/T$. Then $\mathbb{P}[\text{output of } \mathcal{A}' \text{ satisfies } C] \geq 1 - 2\gamma$.

For a number of applications, the cost of the approximate sampling oracle is polylogarithmic in the value $\delta$. In such cases, we can use the following crude estimate:
Corollary 9. There is an absolute constant \(c > 0\) for which the following holds. Suppose that the sampling oracle \(\tilde{\mu}_\beta\) has \(\|\tilde{\mu}_\beta - \mu_\beta\|_{TV} \leq (\frac{1}{n^*} + \min\{n, q\} + \frac{1}{\epsilon} + \log \frac{m^2}{\gamma})^{-c}\). Then all our algorithmic results remain valid for oracle \(\tilde{\mu}_\beta\) as they do for an exact sampling oracle \(\mu_\beta\).

1.4 Miscellaneous formulas and definitions

We collect a few assorted results and notations we will use in our algorithm.

- For values \(\alpha, \beta\) and \(k, \ell \in \mathcal{H}\) we have
  \[\mu_\alpha(k)\mu_\beta(\ell) = e^{(\alpha - \beta)(k - \ell)} \cdot \mu_\alpha(\ell)\mu_\beta(k)\]  
  In particular, if \(\alpha \leq \beta\) and \(k \leq \ell\) then \(\mu_\alpha(k)\mu_\beta(\ell) \geq \mu_\alpha(\ell)\mu_\beta(k)\).

- For two real numbers \(x, y\) we say that \(x\) is an \(\epsilon\)-estimate of \(y\) if \(|\log x - \log y| \leq \epsilon\).

- We show the following lemma in Appendix C:

  Lemma 10. Let \(a_1, \ldots, a_m\) be a non-negative log-concave sequence satisfying \(a_k \leq \frac{1}{k}\) for each \(k \in [m]\). Then \(a_1 + \ldots + a_m < e\).

Note that without the log-concavity assumption we would have \(a_1 + \ldots + a_m \leq \sum_{k=1}^m \frac{1}{k} \leq 1 + \log m\) (by a well-known inequality for the harmonic series). Motivated by these facts, we define the following parameter for any \(h \leq n\) which we use throughout the paper:

\[\Gamma_h = \begin{cases} 1 + \log(h + 1) & \text{in the general setting} \\ e & \text{in the log-concave setting} \end{cases}\]

We also define \(\Gamma = \Gamma_n\).

- We define \(\overline{\mathcal{H}} = \mathcal{H} \cup \{-\infty, +\infty\}\). For a set \(\mathcal{K} \subseteq \mathcal{H}\), we define \(\text{span}(\mathcal{K}) = 1 + \max \mathcal{K} - \min \mathcal{K}\). For a set \(\mathcal{K} \subseteq H\), we define \(\text{span}(\mathcal{K}) = \text{span}(\mathcal{K} \cap \mathcal{H})\).

2 Main data structures and subroutines

Before we describe our algorithms for \(P_{\text{coef}}\), let us formally define the main data structures and subroutines that we will need. We only state the summary results for these procedures here; the formal proofs will be described in later sections. There are five main subroutines we will use:

1. Sample
2. BinarySearch
3. FindRepresentatives
4. FindSchedule, which generates an object that we refer to as a schedule.
5. SolvePratio, to solve problem \(P_{\text{ratio}}\) using a schedule.

We will now provide formal specifications of these routines and associated data structures.
Theorem 13. We can obtain an unbiased estimator $\hat{\mu}_\beta$ of vector $\mu_\beta \in [0,1]^K$ for any given value $\beta \in [\beta_{\text{min}}, \beta_{\text{max}}]$ by taking $N \geq 1$ independent samples from $\mu_\beta(\cdot)$ and computing the empirical frequencies. We write $\hat{\mu}_\beta \leftarrow \text{Sample}(\beta; N)$ for this process. We analyze it using some standard concentration bounds for the binomial distribution which we derive in Appendix [B].

Lemma 11. For parameters $\varepsilon > 0, \gamma \in (0,1], p_0 \in (0,1]$ define the value

$$ R(\varepsilon, \gamma, p_0) = \left\lceil \frac{2\varepsilon^2 \log \frac{2}{\gamma}}{(1 - e^{-\varepsilon})^2 p_0} \right\rceil = \Theta \left( \frac{\log \frac{1}{\varepsilon^2 p_0}}{\varepsilon^2 p_0} \right) $$

Let $\hat{p} \sim \frac{1}{N} \text{Binom}(N, p)$ for $N \geq R(\varepsilon, \gamma, p_0)$. Then with probability at least $1 - \gamma$ we have

$$ \hat{p} \in \begin{cases} [e^{-\varepsilon}p, e^\varepsilon p] & \text{if } p \geq e^{-\varepsilon}p_0 \\ [0, p_0] & \text{if } p < e^{-\varepsilon}p_0 \end{cases} \quad (4) $$

We write $\hat{\mu}_\beta \leftarrow \text{Sample}(\beta; \varepsilon, \gamma, p_0)$ as shorthand for $\hat{\mu}_\beta \leftarrow \text{Sample}(\beta; R(\varepsilon, \gamma, p_0))$. Most of our algorithms are based on executing $\hat{\mu}_\beta \leftarrow \text{Sample}(\beta; \varepsilon, \gamma, p_0)$ for various choices of $\beta, \varepsilon, \gamma, p_0$, and making certain decisions or estimates based on values $\hat{\mu}_\beta(k)$. The algorithms succeed as long as $\hat{\mu}_\beta(k)$ does not deviate much from the true value $\mu_\beta(k)$, in line with the conditions given above. When we execute $\hat{\mu}_\beta \leftarrow \text{Sample}(\beta; \varepsilon, \gamma, p_0)$, we say that this well-estimates some value $k$ if condition (4) holds for $p = \mu_\beta(k)$ and $\hat{p} = \hat{\mu}_\beta(k)$; otherwise it mis-estimates $k$. This condition depends on the parameters $\varepsilon, p_0$ and not solely on the values of $\hat{\mu}_\beta(k), \mu_\beta(k)$. Regardless of the value $\mu_\beta(k)$, Lemma [11] ensures that any given index $k$ is mis-estimated with probability at most $\gamma$. Crucially, if Eq. (4) holds, and either $p \geq p_0$ or $\hat{p} \geq p_0$, then $\hat{p}$ is an $\varepsilon$-estimate of $p$.

BinarySearch. Given value $\theta \in \mathbb{R}$, this subroutine attempts to find a value $\beta$ such that $\mu_\beta([0, \theta]) \approx 1/2 \approx \mu_\beta([\theta, n])$, if any such value exists. Since the value $\mu_\beta([0, \theta])$ is a monotonic function of $\beta$, it does this by binary search. We describe this in Section [4]

Formally, the algorithm $\text{BinarySearch}(\beta_{\text{left}}, \beta_{\text{right}}, \theta, \gamma, \tau)$ takes as inputs values $\beta_{\text{left}}, \beta_{\text{right}}$ with $\beta_{\text{min}} \leq \beta_{\text{left}} < \beta_{\text{right}} \leq \beta_{\text{max}}$ and value $\theta \in \mathbb{R}$. It must return a value $\beta \in [\beta_{\text{left}}, \beta_{\text{right}}]$. Here, the parameter $\gamma$ is the requested failure probability. Ideally, $\beta$ should satisfy $\mu_\beta([0, \theta]) \approx 1/2 \pm \tau \approx \mu_\beta([\theta, n])$. The parameter $\tau$ is the required accuracy in the approximation; in all the algorithms we consider, this will always be regarded as a constant.

We say that the call $\beta \leftarrow \text{BinarySearch}(\beta_{\text{left}}, \beta_{\text{right}}, \theta, \gamma, \tau)$ is good if $\beta \in \Lambda_{\tau}(\beta_{\text{left}}, \beta_{\text{right}}, \theta)$ where we define

$$ \Lambda_{\tau}(\beta_{\text{left}}, \beta_{\text{right}}, \theta) = \left\{ \beta \in [\beta_{\text{left}}, \beta_{\text{right}}] : \begin{array}{ll} \beta > \beta_{\text{left}} & \Rightarrow \mu_\beta([0, \theta]) \geq \tau \\ \beta < \beta_{\text{right}} & \Rightarrow \mu_\beta([\theta, n]) \geq \tau \end{array} \right\} $$

Our main result for this procedure, which may be of independent interest, will be the following:

Theorem 12. Suppose that $\tau$ is any fixed constant. Then $\text{BinarySearch}(\beta_{\text{left}}, \beta_{\text{right}}, \theta, \gamma, \tau)$ has expected sample complexity $O(\log \frac{2\tau}{\gamma})$, and the call is good with probability at least $1 - \gamma$.

FindRepresentatives. A representative set $R$ is a subset of the interval $[\beta_{\text{min}}, \beta_{\text{max}}]$. It is called proper for $\mathcal{K}$ with respect to parameter $\zeta \leq 1$ if every $k \in \mathcal{K}$ satisfies $\max_{\alpha \in R} \mu_\alpha(k) \geq \zeta \Delta(k)$.

Given a set $\mathcal{K}$, the algorithm $\text{FindRepresentatives}$ attempts to find a representative set $R$ which is proper for $\mathcal{K}$, with respect to some constant parameter $\zeta$. We describe this in Section [5] Formally, we show the following:

Theorem 13. The procedure $R \leftarrow \text{FindRepresentatives}(\mathcal{K}, \gamma)$ has the following properties:
(a) The expected sample complexity is $O(\min\{\sqrt{q \log n}, q, |\mathcal{K}|\} \log \frac{2\tau}{\gamma})$.
(b) The representative set $R$ is proper for $\mathcal{K}$ and parameter $\zeta = 1/256$ with probability at least $1 - \gamma/n$.
(c) $|R| \leq |\mathcal{K}|$ with probability one, and $|R| \leq O(\min\{q, \sqrt{q \log n}\})$ with probability at least $1 - \gamma/n$. 

9
FindSchedule. Let us first introduce some basic terminology to define a schedule.

- A weighted interval is a tuple $\sigma = ([\sigma^-, \sigma^+], \sigma^{\text{weight}})$ where $\sigma^-, \sigma^+ \in \mathcal{H}$, and $\sigma^\leq \sigma^+$. We denote $\text{Ends}(\sigma) = \{\sigma^-, \sigma^+\} \cap \mathcal{H}$. We define $\text{span}(\sigma) = \text{span}(\sigma \cap \mathcal{H})$.

- An extended weighted interval is a tuple $(\beta, \sigma)$ where $\beta \in [\beta_{\min}, \beta_{\max}]$ and $\sigma$ is a weighted interval. The tuple is called proper if $\mu_\beta(k) \geq \sigma^{\text{weight}}$ for all $k \in \text{Ends}(\sigma)$.

- A sequence $\mathcal{I} = ((\beta_0, \sigma_0), \ldots, (\beta_t, \sigma_t))$ of distinct extended weighted intervals will sometimes be viewed as a set, and so we write $(\beta, \sigma) \in \mathcal{I}$. We also denote $\text{InvWeight}(\mathcal{I}) = \sum_{(\beta, \sigma) \in \mathcal{I}} \frac{1}{\sigma^{\text{weight}}}$.

Our main algorithm, which we show in Section 7, is summarized as follows:

**Theorem 14.** There is an algorithm $\text{FindSchedule}(\gamma)$ that produces a schedule $\mathcal{I}$ with $\text{InvWeight}(\mathcal{I}) \leq a(n+1)\Gamma$ and $\mathbb{P}[\mathcal{I} \text{ is proper}] \geq 1 - \gamma$, where $a > 4$ is an arbitrary constant. This algorithm has expected sample complexity $O(n\Gamma (\log^2 n + \log \frac{1}{\gamma}) + n \log q)$.

Solving $P^B_{\text{ratio}}$. We have already summarized this algorithm $\text{SolvePratio}$ in the introduction. To reiterate, in Section 6, we show the following:

**Theorem 15.** The procedure $\hat{Q} \leftarrow \text{SolvePratio}(\mathcal{B}, \gamma, \varepsilon)$ has the following guarantees:
(a) The expected sample complexity is
$$O\left(\min\left\{\frac{(|\mathcal{B}| + q \log n) \log |\mathcal{B}|}{\varepsilon^2}, \frac{(|\mathcal{B}|n + n^2)\Gamma \log |\mathcal{B}|}{\varepsilon^2} + n \log q\right\}\right)$$
(b) With probability at least $1 - \gamma$, every $\beta \in \mathcal{B}$ satisfies $\hat{Q}_\beta/Q_\beta \in [e^{-\varepsilon}, e^\varepsilon]$.

3 Solving $P^\mu_{\text{coef}}$.

At this point, we are ready to describe our main algorithms to solve $P^\mu_{\text{coef}}$. This will have two parts. First, we will develop algorithms with sample complexity depending on $n$ instead of $\text{span}(\mathcal{K})$ directly. Later, in Section 6.2, we use rejection sampling to transfer these to the bounds shown in Theorem 11.

Although our definition of an $(\varepsilon, \mathcal{K})$-estimate does not require any condition on individual entries of $\hat{c}_k$, the algorithm here will yield a stronger guarantee which we refer to as lower-normalized $(\varepsilon, \mathcal{K})$-estimate of $c$. Specifically, the estimate $\hat{c}$ satisfies (i) $\hat{c}_k > 0$ for all $k \in \mathcal{K}$; (ii) for all $k \in \mathcal{H}$ with $\hat{c}_k > 0$ the value $\hat{c}_k$ is an $\varepsilon/2$-estimate of the value $c_k = \frac{c_k}{Z(\beta_{\min})}$. It is immediate that this is also an $(\varepsilon, \mathcal{K})$-estimate of $c$. Note that this can be useful even when $|\mathcal{K}| = 1$.

Our generic algorithm to solve $P^\mu_{\text{coef}}$ is described as follows:

**Algorithm 1**: Solving $P^\mu_{\text{coef}}$. **Input:** parameters $\mathcal{K}, \gamma, \mu_* > 0$.

1. call $R \leftarrow \text{FindRepresentatives}(\mathcal{K}, \gamma/3)$
2. call $\hat{Q} \leftarrow \text{SolvePratio}(R, \gamma/3, \varepsilon/4)$
3. set $\hat{c}_k \leftarrow 0$ for all $k \in \mathcal{H}$
4. foreach $\alpha \in R$ do
   5.      set $\hat{\mu}_\alpha \leftarrow \text{Sample}(\alpha; \varepsilon/4, \frac{\gamma}{|\mathcal{K}|}, e^{-\varepsilon/4} \mu_* \zeta)$
   6.      estimate $\hat{c}_k \leftarrow \hat{Q}_\alpha e^{-\alpha_k} \hat{\mu}_\alpha(k)$ for each $k \in \mathcal{K}$ with $\hat{\mu}_\alpha(k) \geq e^{-\varepsilon/4} \mu_* \zeta$ and $\hat{c}_k = 0$.

The following two results summarize Algorithm 1.
Theorem 16. With probability at least $1 - \gamma$, the estimated vector $\hat{c}$ produced by Alg. 1 is a lower-normalized $(\varepsilon, K^*)$-estimate of $c$.

Proof. Let us first show that the $\hat{c}_k$ is a lower-normalized estimate of $c_k$ with good probability. Suppose that $R$ is proper for $K$, the call at line 2 estimates every $Q_\alpha$ correctly for $\alpha \in R$, and line 4 well-estimates every $k \in K$. Since $|R| \leq |K|$ with probability one, this overall has probability at least $1 - \gamma$. We then show that the resulting values $\hat{c}_k$ are lower-normalized $(\varepsilon, K^*)$-estimates of $c$.

First, suppose $\hat{c}_k$ is set to a non-zero value at line 5. So $\hat{\mu}_\alpha(k) \geq e^{-\varepsilon/4} \mu_\alpha \zeta$, and so $\hat{\mu}_\alpha(k)$ is an $\varepsilon/4$-estimate of $\mu_\alpha(k)$. Also, $\hat{Q}_\alpha$ is an $\varepsilon/4$-estimate of $Q_\alpha$. Since $\hat{c}_k = Q_\alpha e^{-\alpha k} \mu_\alpha(k)$, this implies that $\hat{c}_k$ is an $\varepsilon/2$-estimate of $c_k$.

Next, consider $k \in K^*$. Since $R$ is proper for $K$, there is $\alpha \in R$ with $\mu_\alpha(k) \geq \mu_\star \zeta$. This value of $\alpha$ has $\hat{\mu}_\alpha(k) \geq e^{-\varepsilon/4} \mu_\star \zeta$, and so line 5 sets $\hat{c}_k$ to a non-zero value (if it was not already set in an earlier iteration).

Proposition 17. For $r = \min\{\sqrt{q \log n}, q, |K|\}$, Alg. 1 has expected sample complexity

$$O\left(\frac{r \log |K|}{\varepsilon^2 \mu_\star} + \min\left\{\frac{q \log n \log r}{\varepsilon^2}, \frac{n^2 \Gamma \log \frac{r}{\varepsilon^2}}{\varepsilon^2} + n \log q\right\}\right)$$

Proof. Let us write $m = |K|$ and $S = |R|$. Since $S \leq m + 1 \leq n + 2$ with probability one, while $S \leq O(r)$ with probability $1 - \gamma/n$, we see that $E[S] \leq O(r)$. Now let us examine the expected sample complexity line by line.

Line 1: $O(r \log \frac{nq}{\gamma})$.

Line 2: If we condition on fixed $R$, then the expected sample complexity is

$$O\left(\min\left\{\frac{(S + q \log n) \log \frac{S}{\varepsilon^2}}{\varepsilon^2}, \frac{(Sn + n^2) \Gamma \log \frac{S}{\varepsilon^2}}{\varepsilon^2} + n \log q\right\}\right).$$

We can now integrate out the random variable $S$. Note that since $S \leq m \leq n$, we have $E[S \log S] \leq E[S \log n] \leq O(q \log n)$. Also, by Jensen’s inequality we have $E[\log S] \leq O(\log r)$. Overall, the expected sample complexity of line 2 is given by

$$O\left(\min\left\{\frac{q \log n \log r}{\varepsilon^2}, \frac{n^2 \Gamma \log \frac{r}{\varepsilon^2}}{\varepsilon^2} + n \log q\right\}\right).$$

Line 5: The expected sample complexity is

$$O\left(\frac{E[S] \log \frac{m}{\varepsilon^2 \mu_\star}}{\varepsilon^2}\right) = O\left(\frac{r \log \frac{m}{\varepsilon^2 \mu_\star}}{\varepsilon^2}\right).$$

Summing these terms, we get a total expected sample complexity of

$$O\left(r \log \frac{nq}{\gamma} + \frac{r \log \frac{m}{\varepsilon^2 \mu_\star}}{\varepsilon^2} + \min\left\{\frac{q \log n \log r}{\varepsilon^2}, \frac{n^2 \Gamma \log \frac{r}{\varepsilon^2}}{\varepsilon^2} + n \log q\right\}\right).$$

We claim that the term $r \log \frac{nq}{\gamma}$ is always dominated by one of the other terms. This is clear in the second term of $\min\{\}$ is the minimizer, since $r \leq n$. Otherwise, if the first term $\frac{q \log n \log r}{\varepsilon^2}$ is the minimizer, then we must have $q \leq \text{poly}(n)$; in this case $r \log \frac{m}{\varepsilon^2} \leq O(r \log \frac{1}{\gamma}) \leq O(q \log \frac{1}{\gamma}) \leq O\left(\frac{q \log n \log r}{\varepsilon^2}\right)$. Dropping the term $r \log \frac{nq}{\gamma}$ gives the stated expected sample complexity. \qed
3.1 Alternative algorithm for the log-concave setting

In the log-concave setting, the following alternative algorithm for $P_{\text{coef}}^{\mu,K}$ is more efficient than Algorithm 1 in most cases:

**Algorithm 2**: Solving $P_{\text{coef}}^{\mu,H}$ in the log-concave setting. **Input**: parameters $\varepsilon, \gamma, \mu > 0$

1. Let $\mathcal{I} = ((\beta_0, \sigma_0), \ldots, (\beta_t, \sigma_t)) \leftarrow \text{FindSchedule}(\gamma/3)$
2. call $\text{SolvePratio}(\mathcal{B}, \varepsilon/4, \gamma/3)$ where $\mathcal{B} = \{\beta_0, \ldots, \beta_t\}$
3. update $\sigma_0^{\text{weight}} \leftarrow \min\{\sigma_0^{\text{weight}}, \frac{\mu_+}{1+\mu_+\sigma_0}\}$ and $\sigma_t^{\text{weight}} \leftarrow \min\{\sigma_t^{\text{weight}}, \frac{\mu_+}{1+\mu_+(n-\sigma_t)}\}$
4. foreach $(\beta, \sigma) \in \mathcal{I}$ let $\hat{\mu}_\beta \leftarrow \text{Sample}(\beta; \varepsilon/4, \frac{\gamma}{3(n+1)}; e^{-\varepsilon/4}\sigma^{\text{weight}})$
5. foreach $k \in \mathcal{H}$ do
6. pick tuple $(\beta, \sigma) \in \mathcal{I}$ with $k \in [\sigma^-, \sigma^+]$
7. set $\hat{c}_k = \begin{cases} \hat{Q}_\beta e^{-\beta k} \cdot \hat{\mu}_\beta(k) & \text{if } \hat{\mu}_\beta(k) \geq e^{-\varepsilon/4}\sigma^{\text{weight}} \\ 0 & \text{otherwise} \end{cases}$

Note that lines 1 and 2 have expected sample complexity of respectively $O(n(\log^2 n + \log q + \log \frac{\mu}{\gamma}))$ and $O(\frac{\min(n+q \log n, n^2) \log \frac{\mu}{\varepsilon}}{\varepsilon^2})$. The update in line 3 increases $\text{InvWeight}(\mathcal{I})$ by at most $2/\mu_+ + 2n$, therefore line 4 has sample complexity $O((n+1/\mu_+) \cdot \frac{\log \frac{\mu}{\varepsilon}}{\varepsilon^2})$. Overall, we see that Algorithm 2 has expected sample complexity

$$O\left(\min\left\{\frac{(n^2 + \frac{1}{\mu_+}) \log \frac{\mu}{\varepsilon}}{\varepsilon^2} + n \log q, \frac{\log n + n + \frac{1}{\mu_+}}{\varepsilon^2} + n \log^2 n\right\}\right)$$

The following main result shows that Algorithm 2 indeed solves $P_{\text{coef}}$:

**Theorem 18.** In the log-concave setting, the estimated vector $\hat{c}$ produced by Alg. 2 is a lower-normalized $(\varepsilon, \mathcal{H}^*)$-estimate of $c$ with probability at least $1 - \gamma$.

**Proof.** By construction, the schedule $\mathcal{I}$ in line 1 is proper with probability at least $1 - \gamma/3$. By the specification of $\text{SolvePratio}$ the value $\hat{Q}_\beta$ is an $\varepsilon/4$-estimate of $Q_\beta$ with probability at least $1 - \gamma/3$. With probability at least $1 - \gamma/3$, every iteration of line 4 well-estimates every value $\ell \in \mathcal{H}$. Let us assume that all these events occur, which has overall probability at least $1 - \gamma$, and then show that the resulting values $\hat{c}$ satisfy the required property.

First, if $\hat{c}_k > 0$ for index $k$, then $\hat{\mu}_\beta(k) \geq e^{-\varepsilon/4}\sigma^{\text{weight}}$ for some tuple $(\beta, \sigma)$, so $\hat{\mu}_\beta(k)$ is an $\varepsilon/4$-estimate of $\mu_\beta(k)$. Since $\hat{Q}_\beta$ is an $\varepsilon/4$-estimate of $Q_\beta$, this implies that $\hat{c}_k$ is an $\varepsilon/2$-estimate of $c_k$.

Next, consider $k \in \mathcal{H}^*$ and corresponding tuple $(\beta, \sigma)$ chosen at line 6 with $k \in [\sigma^-, \sigma^+]$. We need to show that $\hat{c}_k > 0$. We claim that $\mu_\beta(k) \geq \sigma^{\text{weight}}$; for this, three cases are possible.

- $0 < i < t$. Then $\mu_\beta(k) \geq \min\{\mu_\beta(\sigma^-), \mu_\beta(\sigma^+)\} \geq \sigma^{\text{weight}}$ where the first inequality follows from log-concavity of the coefficients and the second inequality holds since $\mathcal{I}$ is proper.
- $i = 0$, and so $\beta = \beta_{\min}$. Suppose for contradiction that $\mu_\beta(k) < \sigma^{\text{weight}}$. By log-concavity $\mu_\beta(\ell) \leq \mu_\beta(k) < \sigma^{\text{weight}}$ for all $\ell \leq k$. Therefore, $\mu_\beta([0, k-1]) < k\mu_\beta(k) \leq \sigma^+ \cdot \sigma^{\text{weight}}$.

Since $k \in \mathcal{H}^*$, we have $\mu_\alpha(k) \geq \mu_+$ for some $\alpha \in [\beta_{\min}, \beta_{\max}]$. By Eq. (3), for each $\ell \geq k$ we have $\mu_\beta(\ell) \leq \mu_\alpha(\ell) \cdot \frac{\mu_\beta(k)}{\mu_\alpha(\ell)} < \mu_\alpha(\ell) \cdot \frac{\sigma^{\text{weight}}}{\mu_+}$, and therefore $\mu_\beta([k, n]) \leq \frac{\sigma^{\text{weight}}}{\mu_+} \cdot \mu_\alpha([k, n]) \leq \frac{\sigma^{\text{weight}}}{\mu_+}$. We can now obtain a contradiction as follows:

$$1 = \mu_\beta([0, k-1]) + \mu_\beta([k, n]) < \sigma^+ \cdot \sigma^{\text{weight}} + \frac{\sigma^{\text{weight}}}{\mu_+} \leq (\sigma^+ + 1/\mu_+) \cdot \frac{\mu_+}{1+\mu_+\sigma^+} = 1.$$

- $i = t$. This case is completely analogous to the previous one.

We have thus shown that $\mu_\beta(k) \geq \sigma^{\text{weight}}$. Since $\hat{\mu}_\beta$ well-estimates $k$ with respect to parameters $\varepsilon/4, \sigma^{\text{weight}}$, we get that $\hat{\mu}_\beta(k) \geq e^{-\varepsilon/4}\mu_\beta(k) \geq e^{-\varepsilon/4}\sigma^{\text{weight}}$, and so $\hat{c}_k > 0$. \[\square\]
3.2 Reducing the parameter range

We now describe how to get sample complexities for $P_{\mu, K}^{\mu_0, K}$ that depend upon $h = \text{span}(K)$ instead of $n$. The key idea is to use rejection sampling in order to simulate a Gibbs distribution that takes on values only in the range $[\min K, \max K]$. We begin with some general results about sampling from restricted intervals.

Proposition 19. Let $k \in \mathcal{H}$, and suppose that $\beta \in \Lambda_\tau(\beta_{\min}, \beta_{\max}, k)$ for some $\tau$ in $(0, 1/2)$.

(a) For $\ell \geq k$ we have $\max_{\ell' \in [\beta, \beta_{\max}]} \mu_{\ell'}(\ell) \geq \tau \Delta(\ell)$.

(b) For $\ell \leq k$ we have $\max_{\ell' \in [\beta_{\min}, \beta]} \mu_{\ell'}(\ell) \geq \tau \Delta(\ell)$.

Proof. We only show (a); part (b) is completely analogous.

Let $\alpha \in [\beta_{\min}, \beta_{\max}]$ be chosen so that $\mu_\alpha(\ell) = \Delta(\ell)$. If $\alpha \geq \beta$, then the result holds immediately by setting $\alpha' = \alpha$. Otherwise, if $\alpha < \beta$, note that $\beta > \beta_{\min}$. By definition of $\Lambda_\tau$, we then have $\mu_\beta([0, k]) \geq \tau$. Since $\ell \geq k$ and $\alpha \leq \beta$, this implies

$$\mu_\alpha(\ell) = \frac{c_\ell e^{\alpha \ell}}{\sum_{i=0}^\ell c_i e^{\alpha i}} \leq \frac{c_\ell}{\sum_{i=0}^\ell c_i e^{-\alpha (\ell-i)}} \leq \frac{c_\ell}{\sum_{i=0}^\ell c_i e^{-\beta (\ell-i)}} = \frac{\mu_\beta(\ell)}{\mu_\beta([0, k])} \leq \frac{\mu_\beta(\ell)}{\tau}$$

Thus, setting $\alpha' = \beta$ works. □

Proposition 20. Let $k \in \mathcal{H}$, and suppose that $\beta \in \Lambda_\tau(\beta_{\min}, \beta_{\max}, k)$ for some constant $\tau$ in $(0, 1/2)$.

(a) Suppose that $\beta < \beta_{\max}$. Then in $O(1)$ expected samples from $\mu$, we can sample from the Gibbs distribution $\mu'$ with coefficients $c_0, \ldots, c_n$ for any $\alpha \in [\beta, \beta_{\max}]$. Furthermore, the distribution $\mu'$ on this range has diversity parameter $Z'(\beta_{\max})/Z'(\beta) = q' \leq q + O(1)$, and for any $\ell \geq k$ we have

$$\Delta'(\ell - k) = \max_{\alpha' \in [\beta, \beta_{\max}]} \mu_{\alpha'}(\ell - k) \geq \tau \Delta(\ell)$$

(b) Suppose that $\beta > \beta_{\min}$. Then in $O(1)$ expected samples from $\mu$, we can sample from the Gibbs distribution with coefficients $c_0, \ldots, c_k$ for any $\alpha \in [\beta_{\min}, \beta]$. Furthermore, the distribution $\mu'$ on this range has diversity parameter $q' \leq q + O(1)$, and for any $\ell \leq k$ we have

$$\Delta'(\ell) = \max_{\alpha' \in [\beta_{\min}, \beta]} \mu_{\alpha'}(\ell) \geq \tau \Delta(\ell)$$

Proof. We prove only part (a); part (b) is very similar, but slightly simpler.

Since $\beta < \beta_{\max}$, by definition of $\Lambda_\tau$ we have $\mu_\beta([k, n]) \geq \tau$. For $\ell \in \{0, \ldots, n-k\}$ and $\alpha \geq \beta$ we have

$$\mu_\alpha(\ell) = \frac{c_\ell e^{\alpha \ell}}{\sum_{i=0}^\ell c_i e^{\alpha i}} = \frac{c_\ell e^{\alpha (\ell-k)}}{\sum_{i=0}^{n-k} c_i e^{\alpha (i-k)}} = \frac{\mu_\alpha(\ell)}{\mu_\alpha([k, n])}$$

By Lemma 23, we have $\mu_\alpha([k, n]) = \mu_\beta([k, n]) \geq \tau$ for every $\alpha \geq \beta$. Thus, in order to sample $\mu'$ for $\alpha \geq \beta$, we can repeatedly sample from $\mu_\alpha$ until we draw an item $i \geq k$; we then output $i-k$. The expected number of samples we draw is $O(1)$, and we select value $i$ with probability $\mu_\alpha(i)/\mu_\alpha([k, n]) = \mu'_\alpha(i-k)$.

Next, we compute ratio $Z'_{\beta_{\max}}/Z'_{\beta}$ as

$$\frac{Z'_{\beta_{\max}}}{Z'_{\beta}} = \frac{\sum_{i=0}^{n-k} c_i e^{\beta_{\max} i}}{\sum_{i=0}^{n-k} c_i e^{\beta i}} = \frac{e^{-\beta_{\max} k}}{e^{-\beta k}} \sum_{i=0}^{n-k} c_i e^{\beta_{\max} i} = e^{(1-\beta_{\max}) k} \frac{Z(\beta_{\max}) \mu_{\beta_{\max}}([k, n])}{Z(\beta) \mu_{\beta}([k, n])}$$

Clearly $e^{(1-\beta_{\max}) k} \leq 1$. Furthermore, we have $\mu_{\beta_{\max}}([k, n]) \leq 1$ and $\mu_\beta([k, n]) \geq \tau$, so $\frac{Z'_{\beta_{\max}}}{Z'_{\beta}} \leq \tau \times \frac{Z(\beta_{\max})}{Z(\beta_{\min})} = O(Q)$. Taking logarithms, we thus have $q' \leq q + O(1)$.

Finally, by Proposition 20 there is $\alpha' \in [\beta, \beta_{\max}]$ with $\mu_{\alpha'}(\ell) \geq \tau \Delta(\ell)$, and $\mu'_{\alpha'}(\ell-k) \geq \mu_{\alpha'}(\ell)$. □
As a first step, we will discuss how to get sample complexity which is a function of \( k^+ \) instead of \( n \). As a rough outline, we will first use binary search to find a value such that \( \mu, \beta([0, k^+]) \geq \Omega(1) \). We will then convert our oracle for \( \mu \) into an oracle for the distribution \( \mu' \) on coefficients \( c_0, \ldots, c_{k^+} \), and solve \( P_{\mu', \mathcal{K}} \) for \( \mu' \).

Unfortunately, there are a few edge cases that can be problematic: the binary search may not find a valid \( \beta \), either because no such \( \beta \) exists, or because there was a random failure in the execution of BinarySearch. The following result shows how to handle these cases.

**Theorem 21.** Let \( k = \max \mathcal{K} \) and let \( r = \min\{\sqrt{q \log k}, q, |\mathcal{K}|\} \).

There exists an algorithm to obtain a lower-normalized \((\varepsilon, \mathcal{K}^*)\)-estimate of \( c \) with success probability \( 1 - \gamma \) and expected sample complexity

\[
O\left( \log n + \frac{r \log |\mathcal{K}|}{\varepsilon^2 \mu_s} + \min\left\{ \frac{q \log k \log \frac{\varepsilon}{\gamma}}{\varepsilon^2}, \frac{k^2 \Gamma_k \log \frac{\varepsilon}{\gamma}}{\varepsilon^2} + k \log q \right\} \right)
\]

In the log-concave setting, there exists an algorithm to obtain a lower-normalized \((\varepsilon, \mathcal{K}^*)\)-estimate of \( c \) with success probability \( 1 - \gamma \) and expected sample complexity

\[
O\left( \log n + \min\left\{ \frac{(k^2 + \frac{1}{\mu}) \log k}{\varepsilon^2} + k \log q, \frac{(q \log k + k + \frac{1}{\mu}) \log k}{\varepsilon^2} + k \log^2 k \right\} \right)
\]

**Proof.** We prove only the first result; the second is completely analogous. Let us define

\[
L = \frac{r \log |\mathcal{K}|}{\varepsilon^2 \mu_s} + \min\left\{ \frac{q \log k \log \frac{\varepsilon}{\gamma}}{\varepsilon^2}, \frac{k^2 \Gamma_k \log \frac{\varepsilon}{\gamma}}{\varepsilon^2} + k \log q \right\}
\]

which is the target sample complexity (except for the initial \( \log n \) term).

**The algorithm.** The first step in the process is to call \( \beta \leftarrow \text{BinarySearch}(\beta_{\min}, \beta_{\max}, k, \gamma/4, 1/4) \).

Now if \( \beta = \beta_{\min} \), then we set \( \hat{\mu}_{\beta_{\min}} \leftarrow \text{Sample}(\beta_{\min}; \varepsilon/4, \frac{\gamma}{4|\mathcal{K}|}, e^{-\varepsilon/4} \mu_s / 4) \). For each \( k \in \mathcal{K} \), we estimate \( \hat{c}_k \leftarrow \hat{c}^{-\beta_{\min} k} \hat{\mu}_{\beta_{\min}}(k) \) if \( \hat{\mu}_{\beta_{\min}}(k) \geq e^{-\varepsilon/4} \mu_s \), and \( \hat{c}_k \leftarrow 0 \) otherwise.

Otherwise, if \( \beta > \beta_{\min} \), then let us observe that, as long as the call to BinarySearch was good, we have \( \beta \in \Lambda(\beta_{\min}, \beta_{\max}, k) \) for \( \tau = 1/4 \). So the preconditions of Proposition 20(b) are satisfied, and we can simulate the distribution \( \mu' \) with coefficients \( c_0, \ldots, c_k \). (If the call to BinarySearch was not good, then attempting to simulate \( \mu' \) may require arbitrarily large sample complexity.)

At this point, we use Algorithm 1 on the simulated distribution \( \mu' \) in the range \([\beta_{\min}, \beta]\) with respect to error parameter \( \gamma/4 \) and with \( \mu'_s = \mu_s / 4 \), and we return this estimate as our answer. If, during this process, we make more than \( T \) calls to the distribution \( \mu \), for some parameter \( T \) to be specified, then we immediately abort the process and return any arbitrary answer.

The proof of the second result is the same, except that we use Algorithm 2 instead of Algorithm 1.

Now that we have described the algorithm, let us show that it has the desired success probability of \( 1 - \gamma \) and has the expected sample complexity.

**Sample complexity.** The initial call to BinarySearch has expected sample complexity \( O(\log n^2) \). If \( \beta = \beta_{\min} \), then the sampling from \( \mu_{\beta_{\min}} \) has sample complexity \( O\left( \log(|\mathcal{K}|/\gamma) \right) \). We can see that these contribute overall \( O(\log n + L) \) expected sample complexity.

Suppose that we condition on the event that the call BinarySearch was good. In this case, by Proposition 17, the expected sample complexity Algorithm 1 measured in terms of draws from \( \mu' \), is \( O(L) \). By Proposition 20(b), each call to \( \mu' \) can be simulated using \( O(1) \) calls to \( \mu \) in expectation. Also, the rule for aborting termination can only reduce the expected sample complexity. So, the overall expected sample complexity of this process is at most \( CL \), for some constant \( C > 0 \).
Now let us set $T = 4CL/\gamma$. If the call to $\text{BinarySearch}$ is not good, then our termination rule still ensures that the sample complexity is at most $T$. Since $\text{BinarySearch}$ is good with probability at least $1 - \gamma/4$, the overall expected sample of solving $P_{\text{coeff}}$ for $\mu'$ is at most $(1 - \gamma/4) \times CL + \gamma/4 \times T \leq O(L)$. This concludes the analysis of the sample complexity of this process.

**Correctness.** Let us consider the following four events:

1. The call to $\text{BinarySearch}$ is good
2. If $\beta = \beta_{\min}$, then $\hat{\mu}_{\beta}(k)$ is well-estimated for all $k \in \mathcal{K}$.
3. If $\beta > \beta_{\min}$, then Algorithm 1 on distribution $\mu'$ succeeds.
4. If $\beta > \beta_{\min}$, then Algorithm 1 does not attempt to make more than $T$ samples to $\mu$.

We claim that if these four events all occur, then the algorithm produces an $(\varepsilon, \mathcal{K}^*)$-lower-normalized estimate of $c$. We also claim that these four events in total have probability at least $1 - \gamma$.

The first event has probability at least $1 - \gamma^2/4$, and the second has probability at least $1 - \gamma/4$. Let us condition on these two events occurring, and so we have $\beta = \beta_{\min}$, $\{\beta_{\min}, \beta_{\max}\}^*$, $\beta_{\min}$, $\beta_{\max}$ for $\tau = 1/4$.

Suppose first that $\beta = \beta_{\min}$. By Proposition 19(b), we have $\mu_{\beta_{\min}}^*(\ell) \geq \mu_*/4$ for all $\ell \in \mathcal{K}^*$. Thus, since $\hat{\mu}_{\beta_{\min}}$ is well-estimated for all $\ell \in \mathcal{K}$, we have $\hat{c}_k > 0$ for all $k \in \mathcal{K}^*$. Likewise, any coefficient $k$ with $\hat{c}_k > 0$ is an $\varepsilon/2$-estimate of $c_k$.

Next suppose that $\beta > \beta_{\min}$. By Proposition 20(b), each call to $\mu'$ can be simulated by $O(1)$ calls to $\mu$. Thus, the expected sample complexity of applying Algorithm 1 measured in terms of calls to $\mu$, is at most $CL$. By Markov’s inequality, the probability of aborting due to more than $T$ is, is at most $\gamma/4$. Also, by Theorem 16, the call to Algorithm 1 succeeds with probability at least $1 - \gamma/4$.

Thus, the conditional probability of the last two events is still $1 - \gamma/2$, and the overall probability that all four events hold is at least $1 - \gamma$.

Now let us suppose that these events all hold. By Proposition 20(b), we have $\Delta'(\ell) \geq \Delta(\ell)/4 \geq \mu_*/4 = \mu_*$ for all $\ell \in \mathcal{K}^*$. Thus the estimate produced from Algorithm 1 also solves $P_{\text{coeff}}^{\mathcal{K}, \mu_*}$ for distribution $\mu$.

We can now prove Theorem 1 and Theorem 2, which we restate for convenience:

**Theorem 22.** Let $h = \text{span}(\mathcal{K})$ and let $r = \min\{\sqrt{q \log k, q, |\mathcal{K}|}\}$.

There exists an algorithm to obtain an $(\varepsilon, \mathcal{K}^*)$-estimate of $c$ with probability at least $1 - \gamma$ using expected sample complexity

$$O\left(\log n + \frac{r \log |\mathcal{K}|}{\varepsilon^2 \mu_*} + \min\left\{q \log h \log \frac{\ell}{\varepsilon^2}, \frac{h^2 \Gamma_h \log \frac{\ell}{\varepsilon^2}}{\varepsilon^2} + h \log q\right\}\right)$$

In the log-concave setting, there exists an algorithm to obtain an $(\varepsilon, \mathcal{K}^*)$-estimate of $c$ with probability at least $1 - \gamma$ using expected sample complexity

$$O\left(\log n + \min\left\{(h^2 + \frac{1}{\mu_*}) \log \frac{h}{\varepsilon^2} + h \log q, \frac{(q \log h + h + \frac{1}{\mu_*}) \log \frac{h}{\varepsilon^2}}{\varepsilon^2} + h \log^2 h\right\}\right)$$

**Proof.** The proof is nearly identical to Theorem 21, so we only provide a sketch of the algorithm.

We first call $\beta \leftarrow \text{BinarySearch}(\beta_{\min}, \beta_{\max}, \text{min } \mathcal{K}, \gamma/4, 1/4)$.

If $\beta = \beta_{\max}$, then we set $\hat{\mu}_{\beta_{\max}} \leftarrow \text{Sample}(\beta_{\max}; \varepsilon/4, 4|\mathcal{K}|, e^{-\varepsilon/4} \mu_*/4)$. For each $k \in \mathcal{K}$, we estimate $\hat{c}_k \leftarrow \hat{c}_{\max}^{\beta_{\max}} \hat{\mu}_{\beta_{\max}}(k)$ if $\hat{\mu}_{\beta_{\max}}(k) \geq e^{-\varepsilon/4} \mu_*$, and $\hat{c}_k \leftarrow 0$ otherwise.

Otherwise, if $\beta < \beta_{\max}$, then the preconditions of Proposition 21(a) are satisfied, and we can simulate the distribution $\mu'$ with coefficients $c_k, \ldots, c_n$ for $k = \text{min } \mathcal{K}$. We use Theorem 21 on the simulated distribution $\mu'$ in the range $[\beta, \beta_{\max}]$ with respect to error parameter $\gamma/4$, with $\mu'_k = \mu_*/4$, $\mu'_k = \mu_*/4$. 

15
and with the set $K' = \{\ell - k \mid \ell \in K\}$. We return this estimate (suitably shifted) as our answer. If, during this process, we make more than $T$ calls to the distribution $\mu$, for appropriate choice of $T$, then we abort and return any arbitrary answer.

We remark that the algorithm of Theorem 22 is based on simulating a distribution $\mu'$ and applying Theorem 21 to $\mu'$, while Theorem 21 in turn, is based on simulating a distribution $\mu''$ from $\mu'$. These two layers of indirect simulation allow us to better handle the edge cases compared to directly simulating $\mu''$ from $\mu$.

4 The BinarySearch subroutine

We will now prove Theorem 12. We assume throughout that $\tau$ is constant. Also, we assume that $\theta$ is not an integer, as in that case note that if BinarySearch is good for parameter $\theta + 1/2$ it is also good for $\theta$. In this section we will use the following notation:

$$
\mathcal{H}^- = [0, \theta] \cap \mathcal{H} \quad \mathcal{H}^+ = [\theta, n] \cap \mathcal{H} \quad \Lambda_\tau = \Lambda_\tau(\beta_{\text{left}}, \beta_{\text{right}}, \theta) \quad p(\beta) = \mu_\beta(\mathcal{H}^+)
$$

$$
Z^-(\beta) = \sum_{k \in \mathcal{H}^-} c_k e^{\beta k} = (1 - p(\beta))Z(\beta) \quad Z^+(\beta) = \sum_{k \in \mathcal{H}^+} c_k e^{\beta k} = p(\beta)Z(\beta)
$$

It will be easy to verify that BinarySearch succeeds with probability one if $\mu_\beta(\mathcal{H}^-) = 0$ or $\mu_\beta(\mathcal{H}^+) = 0$. Hence we assume for the remainder of this section that $p(\beta) \in (0, 1)$ for all values $\beta \in \mathbb{R}$. Before we begin our algorithm analysis, we record a few elementary properties about these parameters.

Lemma 23. $p(\beta)$ is a strictly increasing function of $\beta$.

Proof. For any $\beta \in \mathbb{R}$ and $\delta > 0$ we have $Z^-(\beta + \delta) < Z^-(\beta) \cdot e^{\delta \theta}$ and $Z^+(\beta + \delta) > Z^+(\beta) \cdot e^{\delta \theta}$, and thus $\frac{Z^-(\beta + \delta)}{Z^+(\beta + \delta)} < \frac{Z^-(\beta)}{Z^+(\beta)}$. Therefore, $\frac{1}{p(\beta)} - 1 = \frac{Z^-(\beta)}{Z^+(\beta)}$ is a strictly decreasing function of $\beta$, and accordingly $p(\beta)$ is a strictly increasing function of $\beta$.

Since $p(\beta)$ is an increasing function, it has an inverse $p^{-1}$. We use this to define parameter $\beta_{\text{crit}}$:

$$
\beta_{\text{crit}} = \begin{cases} 
\beta_{\text{left}} & \text{if } p(\beta_{\text{left}}) > 1/2 \\
\beta_{\text{right}} & \text{if } p(\beta_{\text{right}}) < 1/2 \\
p^{-1}(1/2) & \text{if } p(\beta_{\text{left}}) \leq 1/2 \leq p(\beta_{\text{right}})
\end{cases}
$$

Proposition 24. There holds $\beta_{\text{right}} - \beta_{\text{crit}} \leq q + 1$.

Proof. Let $\beta_1 = \beta_{\text{right}} - q - 1$ and $p_1 = p(\beta_1)$. If $\beta_1 \leq \beta_{\text{left}}$, then $\beta_{\text{right}} - \beta_{\text{left}} \leq q + 1$ and we are done. Otherwise, we can write

$$
Z(\beta_{\text{right}}) \geq Z^+(\beta_{\text{right}}) \geq Z^+(\beta_1) \cdot e^{\beta_{\text{right}} - \beta_1} = p_1 Z(\beta_1) \cdot e^{\beta_{\text{right}} - \beta_1}
$$

where the second inequality holds since $\min \mathcal{H}^+ \geq 1$. Now since $\beta_1 \geq \beta_{\text{left}} \geq \beta_{\text{min}}$, there holds

$$
q \geq \log \frac{Z(\beta_{\text{right}})}{Z(\beta_1)} \geq \beta_{\text{right}} - \beta_1 + \log p_1 = q + 1 + \log p_1
$$

This implies that $\log p_1 \leq -1$, which in turn implies that $p_1 \leq 1/2$. So $\beta_1 \leq \beta_{\text{crit}}$.

The starting point for our algorithm is a sampling procedure of Karp & Kleinberg [14] for noisy binary search. We summarize their algorithm as follows:
Theorem 25 ([14]). Suppose we have oracle access to draws from Bernoulli random variables $X_1, \ldots, X_N$, wherein each $X_i$ has mean $x_i$, and we know $0 \leq x_1 \leq x_2 \leq \cdots \leq x_N \leq 1$ but the values $x_1, \ldots, x_N$ are unknown. Let us also write $x_0 = 0, x_{N+1} = 1$.

Then there is a sampling procedure which takes as input two parameters $\alpha, \Delta \in (0, 1)$, and uses $O(\log N/\Delta^2)$ oracle queries to the variables $X_i$ in expectation. With probability at least $3/4$, it returns an index $v \in \{0, \ldots, N\}$ such that $[x_v, x_{v+1}] \cap [\alpha - \Delta, \alpha + \Delta] \neq \emptyset$.

By quantization, we can adapt Theorem 25 to weakly solve BinarySearch; we will afterward describe the limitations of this preliminary algorithm and how to get the full result.

Theorem 26. Let $\tau' \in (0, \frac{1}{2})$ be an arbitrary constant. There is a sampling procedure with the following properties:

(i) It takes as input an interval $[\beta'_{\text{left}}, \beta'_{\text{right}}] \subseteq [\beta_{\text{left}}, \beta_{\text{right}}]$ and returns a value $\hat{\beta} \in [\beta'_{\text{left}}, \beta'_{\text{right}}]$.

(ii) If $\beta'_{\text{left}} \leq \beta_{\text{crit}} \leq \beta'_{\text{right}}$, then with probability at least $3/4$ the output $\hat{\beta}$ satisfies $\hat{\beta} \in \Lambda_{\tau'}$.

(iii) The expected sample complexity is $O(\log(n + \beta'_{\text{right}} - \beta'_{\text{left}}))$.

Proof. Let us define parameters

\[
\delta = \frac{2}{n} \log \frac{(1 - \tau') \cdot (1 - 2\tau')}{\tau' \cdot (3 - 2\tau')} > 0
\]

\[
N = \left\lceil \frac{\beta'_{\text{right}} - \beta'_{\text{left}}}{\delta} \right\rceil + 1 = O(n(\beta'_{\text{right}} - \beta'_{\text{left}}) + 1)
\]

Let us define values $u_1, \ldots, u_N$ by $u_i = \beta'_{\text{left}} + \frac{i - 1}{N - 1} (\beta'_{\text{right}} - \beta'_{\text{left}})$. Note that we simulate access to a Bernoulli variable $X_i$ with rate $x_i = p(u_i)$ by drawing $k \sim \mu_{u_i}$ and checking if $k < \theta$.

Our algorithm is to apply Theorem 25 with respect to the variables $X_1, \ldots, X_N$ and with parameters $\alpha = \frac{1}{2}, \Delta = \frac{1}{2} - \tau'$; let $v \in \{0, \ldots, N\}$ denote the resulting return value. If $1 \leq v \leq N - 1$, then we output $\hat{\beta} = \frac{u_v + u_{v+1}}{2}$. If $v = 0$, then we output $\hat{\beta} = \beta'_{\text{left}}$. If $v = N$, then we output $\hat{\beta} = \beta'_{\text{right}}$. This has expected sample complexity $O(\log N/\Delta^2) = O(\log(n + \beta'_{\text{right}} - \beta'_{\text{left}}))$ (bearing in mind that $\Delta$ is constant). This shows property (iii).

To show property (ii), suppose that $v$ satisfies $[x_v, x_{v+1}] \cap \left[\frac{1}{2} - \Delta, \frac{1}{2} + \Delta\right] \neq \emptyset$, which occurs with probability at least $3/4$; we will show that then $\hat{\beta} \in \Lambda_{\tau'}$ as desired. There are a number of cases.

- Suppose that $1 \leq v \leq N - 1$. Then we need to show that $\tau' \leq p(\hat{\beta}) \leq 1 - \tau'$. We will show only the inequality $p(\hat{\beta}) \geq \tau'$; the complementary inequality is completely analogous.

Choose arbitrary $x \in [x_v, x_{v+1}]$ such that $x \geq \frac{1}{2} - \Delta$ (this exists because of our hypothesis that the algorithm of Theorem 25 returned a good answer). We write $u = p^{-1}(x) \in [u_v, u_{v+1}]$. If $u \leq \hat{\beta}$, then $p(\hat{\beta}) \geq p(u) \geq \frac{1}{2} - \Delta \geq \tau'$.

Otherwise, suppose that $u > \hat{\beta}$. Since $\max H^+ = n$, we can then write

\[
\frac{p(\hat{\beta})}{1 - p(\hat{\beta})} = \frac{Z^+(\hat{\beta})}{Z^-(\hat{\beta})} \geq \frac{Z^+(u)e^{-n(u - \hat{\beta})}}{Z^-(u)} = \frac{e^{-n(u - \hat{\beta})}p(u)}{1 - p(u)} \geq \frac{e^{-n(u - \hat{\beta})}(\frac{1}{2} - \Delta)}{\frac{1}{2} + \Delta}
\]

We know that $u_{v+1} - u_v = \frac{1}{N - 1}(\beta'_{\text{right}} - \beta'_{\text{left}}) \leq \delta$, and since $u \geq \frac{1}{2} = (u_v + u_{v+1})/2$, this implies that $\hat{\beta} \geq u - \delta/2$. So we have shown that

\[
\frac{p(\hat{\beta})}{1 - p(\hat{\beta})} \geq \frac{e^{-n\delta/2}(\frac{1}{2} - \Delta)}{\frac{1}{2} + \Delta} = \frac{\tau'}{1 - \tau'}
\]

This in turn implies that $p(\hat{\beta}) \geq \tau'$ as desired.
• Suppose that \( v = 0 \) and \( p(\beta_{\text{left}}) \leq \frac{1}{2} \). Again, we must show that \( \tau' \leq p(\beta) \leq 1 - \tau' \). Since \( \hat{\beta} = \beta'_{\text{left}} \leq \beta_{\text{crit}} \), we have \( p(\beta) \leq \frac{1}{2} \leq 1 - \tau' \).

To show the lower bound, as in the first case, let \( x \in [x_0, x_1] \) be such that \( x \geq \frac{1}{2} - \Delta \). Since \( x_0 = 0 \) and \( x_1 = p(\beta_{\text{left}}) \), we know that \( p^{-1}(x) \leq u_1 = \beta'_{\text{left}} \), so that \( p(\beta_{\text{left}}) \geq x \geq \frac{1}{2} - \Delta \geq \tau' \).

• Suppose that \( v = 0 \) and \( p(\beta_{\text{left}}) > \frac{1}{2} \). In this case, since \( \beta_{\text{left}} \leq \beta'_{\text{left}} \leq \beta_{\text{crit}} \), we know that \( \beta'_{\text{left}} = \beta_{\text{left}} \). The algorithm returns value \( \beta = \beta'_{\text{left}} = \beta_{\text{left}} \) and so \( \beta \in \Lambda_{\tau'} \).

• Suppose that \( v = N \). This is completely analogous to the cases where \( v = 0 \). □

Theorem 26 does not directly solve BinarySearch on its own, for two reasons. First, the success probability is only a constant 3/4, not the desired value 1 − \( \gamma \). Second, the runtime depends on the difference \( \beta'_{\text{right}} - \beta'_{\text{left}} \), which may be unbounded.

We formulate the following algorithm for BinarySearch to address both issues via an exponential back-off strategy. Note that the loop in line 2 runs indefinitely, starting at index value \( i = i_0 \).

**Procedure BinarySearch(\( \beta_{\text{left}}, \beta_{\text{right}}, \theta, \gamma, \tau \)).**

1. set \( i_0 = \lceil \log_2 \log_2 \frac{n}{\tau} \rceil \) and \( \tau' = \frac{\tau + \tau}{2} \)
2. for \( i = i_0, i_0 + 1, i_0 + 2, \ldots \), do
3. set \( \beta_i' = \max \{ \beta_{\text{left}}, \beta_{\text{right}} - 2^{2i} \} \)
4. let \( \beta \) be the output of the alg. of Theorem 26 with \( \beta_{\text{left}} = \beta_i'; \beta_{\text{right}} = \beta_{\text{right}} \)
5. set \( \hat{\mu}_\beta \leftarrow \text{Sample}(\beta; \frac{\tau}{\tau'}, \theta, \gamma, \beta, \tau) \)
6. if \( (\beta = \beta_{\text{left}} \lor \hat{\mu}_\beta(\mathcal{H}^-) \geq \sqrt{\tau'}) \land (\beta = \beta_{\text{right}} \lor \hat{\mu}_\beta(\mathcal{H}^+) \geq \sqrt{\tau'}) \) then return \( \beta \)

**Proposition 27.** For constant \( \tau \), the expected sample complexity of BinarySearch is \( O(\log \frac{n}{\tau}) \).

**Proof.** We claim that the expected sample complexity of iteration \( i \) (if it is reached) is \( O(2^i) \). Indeed, the complexities at lines 4 and 5 are respectively \( O(\log(n(\beta_{\text{right}} - \beta_i'))) + 1 \leq O(\log(n2^{2i})) = O(2^i + \log n) \) and \( O(\log(2^i - i_0 + 2/\gamma)) \leq O(i + \log \frac{1}{\gamma}) \), which together give \( O(2^i + \log \frac{1}{\gamma}) \). By observing that \( 2^i \geq 2^c \geq \log_2 \frac{n}{\tau} \) we get the desired claim.

Let \( s \) be the minimal integer such that \( \beta_{\text{right}} - 2^s \leq \beta_{\text{crit}} \). By Proposition 24 we have \( s \leq \log_2 \log_2(q + 1) \). Let \( t = \max \{ i_0, s \} \).

We first consider the complexity of iterations \( i = i_0, i_0 + 1, \ldots, t \). Each such iteration \( i \) has expected sample complexity is \( O(2^i) \). Summing over \( i = i_0, \ldots, t \) gives expected sample complexity \( O(2^t) \).

We next claim that in each iteration \( \tau > t \), the algorithm BinarySearch terminates with probability at least \( 1 - 2 \cdot 2^{1+i+2} \geq 3/4 \). Overall, the probability of termination at this iteration is at least \( 3/4 \times 3/4 = 9/16 \).

This in turn implies that the probability that BinarySearch reaches iteration \( i = t + 1 + j \) is at most \( (7/16)^2 \). If it does reach this iteration, the expected sample complexity is \( O(2^t) = O(2^{t+j}) \). Thus, the overall expected sample complexity due to iteration \( i > t \) is \( O((7/16)^2 i^2) \).

So the expected sample complexity due to iterations \( i > t \) is at most \( \sum_{i=0}^{\infty} O((7/8)^2 i^2) = O(2^i) \).

The total expected sample complexity of the algorithm is \( O(2^t) = O(\max \{ 2^s, 2^{i_0} \}) = O(\log \frac{n}{\tau}) \). □

Proposition 27 implies, in particular, that BinarySearch terminates with probability 1.

**Proposition 28.** With probability at least \( 1 - \gamma \), the return value \( \beta \) of BinarySearch satisfies \( \beta \in \Lambda_{\tau} \).

**Proof.** By construction, line 5 at iteration \( i \) well-estimates \( \mathcal{H}^- \) and \( \mathcal{H}^+ \) with probability at least \( 1 - \gamma/2^{i-i_0+1} \). Thus, \( \mathcal{H}^- \) and \( \mathcal{H}^+ \) are well-estimated at all iterations with probability at least \( 1 - \sum_{i \geq i_0} \gamma/2^{i-i_0+1} = 1 - \gamma \). If such event happens and BinarySearch returns value \( \beta \) then \( \beta \in \Lambda_{\tau} \). □
5 Proof of Theorem \[13\]: Procedure \texttt{FindRepresentatives}

The procedure \texttt{FindRepresentatives}(\(\mathcal{K}, \gamma\)) is defined formally as follows:

**Algorithm 3: \texttt{FindRepresentatives}(\(\mathcal{K}, \gamma\))**

| Step | Description |
|------|-------------|
| 1 | \(i \leftarrow \max \mathcal{K} \text{ and } R \leftarrow \emptyset\) |
| 2 while true do | |
| 3 | \(\alpha \leftarrow \text{BinarySearch}(\beta_{\min}, \beta_{\max}, i + \frac{1}{2}, \frac{\gamma}{4(\alpha + 1)^2}, 1/4)\) |
| 4 | \(\hat{\alpha} \leftarrow \text{Sample}(\alpha; \log 2, \frac{\gamma}{2(\alpha + 1)^2}, 1/2)\) |
| 5 | set \(j\) to be the maximum integer in the range \(\{0, \ldots, i\}\) such that \(\hat{\alpha}([0, j - 1]) \leq \frac{1}{128}\) |
| 6 | insert \(\alpha\) into \(\mathcal{R}\) |
| 7 if \(\mathcal{K} \cap \{0, \ldots, j - 1\} = \emptyset\) or \(\alpha = \beta_{\min}\) then return \(\mathcal{R}\) |
| 8 update \(i \leftarrow \max \mathcal{K} \cap \{0, \ldots, j - 1\}\) |

For the algorithm analysis, we let \(\alpha_t, i_t, j_t\) denote the values of those variables at iteration \(t\) and \(T\) be the final iteration count. We also write \(Z_t = Z(\alpha_t)\). We write \(R = \{\alpha_1, \ldots, \alpha_T\}\) for the final set returned by this procedure, and we also set \(r = \min\{q, \sqrt{\log n}, |\mathcal{K}|\}\) throughout. We note that, at each iteration, \(i_t\) is an element of \(\mathcal{K}\), and also that \(i_t + 1 < j_t \leq i_t\). Thus \(T \leq |\mathcal{K}|\) with probability one.

Let us say that the execution is *good* if call to \texttt{BinarySearch} is good and each iteration of line 5 well-estimates every interval \(\{0, \ldots, k\}\). By the specification of these subroutines, this has probability at least \(1 - \frac{2}{n}\). We then argue that in the case the set \(R\) satisfies the required conditions, namely that \(R\) is proper with respect to constant \(\zeta = 1/256\) and that \(|R| \leq O(r)\).

We first show Theorem \[13\](b).

**Proposition 29.** Suppose the execution is good. Then for any \(k \in \mathcal{K}\) there exists \(\alpha \in R\) with \(\mu_\alpha(k) \geq \Delta(k)/256\). In particular, \(R\) is proper for \(\mathcal{K}\) with respect to parameter \(\zeta = 1/256\).

**Proof.** Consider \(\beta \in [\beta_{\min}, \beta_{\max}]\) with \(\mu_\beta(k) = \Delta(k)\). We can observe that, due to the termination condition of \texttt{FindRepresentatives} and due to the choice of \(i\) in line 8, there are two possibilities: either there is some iteration \(t\) with \(j_t \leq k \leq i_t\), or \(k < j_T\) and \(\alpha_T = \beta_{\min}\).

For the former case, let us write \(\alpha = \alpha_t, i = i_t, j = j_t\). First, suppose that \(\beta > \alpha\). In this case, \(\alpha \neq \beta_{\max}\), and so since the call to \texttt{BinarySearch} at line 3 is good, we have \(\mu_\alpha([i + 1, n]) \geq \tau = 1/4\). Since \(k \leq i\), we have:

\[
\mu_\beta(k) \leq \frac{c_ke^{\beta k}}{\sum_{\ell=0}^{j_t} c_{e^{\beta \ell}}} \leq \frac{c_ke^{\alpha k}}{\sum_{\ell=0}^{j_t} c_{e^{\alpha \ell}}} = \frac{\mu_\alpha(k)}{\mu_\alpha([i + 1, n])} \leq 4\mu_\alpha(k)
\]

If \(\beta = \alpha\) then the condition obviously holds. Otherwise, suppose that \(\beta < \alpha\). In this case, \(\alpha \neq \beta_{\min}\), and so since the call to \texttt{BinarySearch} at line 3 is good, we have \(\mu_\alpha([0, i]) \geq 1/4\).

By the definition of \(j\), it holds that either \(\hat{\alpha}([0, j]) > 1/128\) or \(j = i\). In the former case, since line 4 well-estimates interval \([0, j]\), it must be that \(\mu_\alpha([0, j]) \geq 1/256\). In latter case, we also have \(\mu_\alpha([0, j]) = \mu_\alpha([0, i]) \geq 1/4 \geq 1/256\). Thus, in either case, since \(k \geq j\), we have:

\[
\mu_\beta(k) \leq \frac{c_ke^{\beta k}}{\sum_{\ell=0}^{j_t} c_{e^{\beta \ell}}} \leq \frac{c_ke^{\alpha k}}{\sum_{\ell=0}^{j_t} c_{e^{\alpha \ell}}} = \frac{\mu_\alpha(k)}{\mu_\alpha([0, j])} \leq 256\mu_\alpha(k)
\]

Finally, suppose that \(k < j_T\) and \(\alpha_T = \beta_{\min} < \beta\). Let us set \(i = i_T\). Since \(\alpha_T \neq \beta_{\max}\) and the call to \texttt{BinarySearch} at line 3 is good, it holds that \(\mu_{\alpha_T}([i + 1, n]) \geq 1/4\). We then compute:

\[
\mu_\beta(k) \leq \frac{c_ke^{\beta k}}{\sum_{\ell=i+1}^{n} c_{e^{\beta \ell}}} \leq \frac{c_ke^{\alpha k}}{\sum_{\ell=i+1}^{n} c_{e^{\alpha \ell}}} = \frac{\mu_{\alpha_T}(k)}{\mu_{\alpha_T}([i + 1, n])} \leq 4\mu_{\alpha_T}(k)
\]

We will next bound \(T\), allowing us to show Theorem \[13\](a), (c).
Proposition 30. Suppose the execution is good. Then for \( t = 1, \ldots, T - 2 \) we have the bound:

\[
\frac{Z_t}{Z_{t+1}} \geq 8e^{\frac{jt_{t+1}}{3(t-j_{t+1})}} > 1
\]

Proof. Because the algorithm terminates whenever \( \alpha_t = \beta_{\min} \), we know that \( \alpha_t > \beta_{\min} \) for \( t < T \). Since BinarySearch is good it holds that \( \mu_{\alpha_t}([0,i_t]) \geq 1/4 \). Also, since line 4 well-estimates every interval and \( i_{t+1} < j_t \), we have \( \mu_{\alpha_t}([0,i_{t+1}]) \leq \mu_{\alpha_t}([0,j_t-1]) \leq 1/64 \). For \( t < T - 1 \), we thus have

\[
\mu_{\alpha_{t+1}}([0,i_{t+1}]) \geq 1/4 > 1/64 > \mu_{\alpha_t}([0,i_{t+1}]).
\]

By Lemma 23, this implies that \( \alpha_t > \alpha_{t+1} \).

If we define the interval \( V_t = [j_t,i_t] \), we have \( \mu_{\alpha_t}(V_t) = \mu_{\alpha_t}([0,i_t]) - \mu_{\alpha_t}([0,j_t-1]) \geq 1/4 - 1/64 \geq 1/8 \) and \( \mu_{\alpha_t}(V_{t+1}) \leq \mu_{\alpha_t}([0,i_{t+1}]) \leq 1/64 \). We can estimate:

\[
\frac{Z_t}{Z_{t+1}} = \frac{\mu_{\alpha_{t+1}}(V_{t+1})}{\mu_{\alpha_t}(V_t)} \times \frac{\sum_{k \in V_{t+1}} c_k e^{\alpha_{t+1}k}}{\sum_{k \in V_t} c_k e^{\alpha_t k}} \geq \frac{1/8}{1/64} \times \frac{\sum_{k \in V_{t+1}} c_k e^{\alpha_{t+1}k}}{\sum_{k \in V_t} c_k e^{\alpha_t k}} \geq 8e^{(\alpha_t - \alpha_{t+1})j_{t+1}}
\]

where the last inequality here comes from the fact that \( j_{t+1} \) is the least element of \( V_{t+1} \) and that \( \alpha_t > \alpha_{t+1} \).

Alternatively, we can estimate:

\[
\frac{Z_t}{Z_{t+1}} = \frac{\mu_{\alpha_{t+1}}(V_t)}{\mu_{\alpha_t}(V_t)} \times \frac{\sum_{k \in V_t} c_k e^{\alpha_{t+1}k}}{\sum_{k \in V_t} c_k e^{\alpha_t k}} \leq \frac{1}{1/4} \times \frac{\sum_{k \in V_t} c_k e^{\alpha_{t+1}k}}{\sum_{k \in V_t} c_k e^{\alpha_t k}} \leq 4e^{(\alpha_t - \alpha_{t+1})i_t}
\]

where again the last inequality comes from the fact that \( i_t \) is the largest element of \( V_t \) and that \( \alpha_t > \alpha_{t+1} \). Putting these two inequalities together, we conclude that
\[
8e^{(\alpha_t - \alpha_{t+1})j_{t+1}} \leq 4e^{(\alpha_t - \alpha_{t+1})j_{t+1}},
\]

which implies that \( (\alpha_t - \alpha_{t+1})(i_t - j_{t+1}) \geq \log 2 \geq 1/2 \). Substituting this bound into Eq. (5) gives the claimed result.

We can use this to estimate \( q \) in terms of \( T \).

Proposition 31. If the execution is good, we have \( T \leq O(\min\{q, \sqrt{q \log n}\}) \).

Proof. If \( T \leq 3 \), this is clear. Otherwise, since \( \beta_{\max} \geq \alpha_1 > \alpha_2 > \cdots > \alpha_{T-2} \geq \beta_{\min} \), we can compute:

\[
q = \log \frac{Z(\beta_{\max})}{Z(\beta_{\min})} \geq \sum_{t=1}^{T-2} \log \frac{Z(\alpha_t)}{Z(\alpha_{t+1})} \geq (T - 2) \log 8 + \frac{1}{2} \sum_{t=1}^{T-2} \frac{j_{t+1}}{i_t - j_{t+1}}
\]

This immediately shows that \( T \leq O(q) \). If \( q \leq \log n \), then we are done. So, let us suppose that \( q > \log n \). For notational convenience, let us suppose that \( T = 3 \) (the odd case is nearly the same), and define \( \mathcal{L} = \{2, 4, 6, \ldots, T - 2\} \).

For \( \ell \in \mathcal{L} \), let us set \( a_\ell = \log(\frac{j_{\ell+1}}{j_{\ell+1}}) \). The sum \( \sum a_\ell \) telescopes as:

\[
\sum_{\ell \in \mathcal{L}} a_\ell = \sum_{\ell \in \mathcal{L}} \log j_{\ell-1} - j_{\ell+1} = \log j_1 - \log j_{T-1} \leq \log n - \log 1 = \log n
\]

Noting that \( i_t \leq j_{t-1} \), we can also lower bound the sum in Eq. (7) as

\[
\sum_{t=1}^{T-2} \frac{j_{t+1}}{i_t - j_{t+1}} \geq \sum_{t=2}^{T-2} \frac{j_{t+1}}{j_{t-1} - j_{t+1}} \geq \sum_{\ell \in \mathcal{L}} \frac{j_{\ell+1}}{j_{\ell-1} - j_{\ell+1}} = \sum_{\ell \in \mathcal{L}} \frac{1}{e^{a_\ell} - 1}
\]

The function \( f(x) = \frac{1}{e^x - 1} \) is decreasing concave-up, and so by Jensen’s inequality we have:

\[
\sum_{\ell \in \mathcal{L}} \frac{1}{e^{a_\ell} - 1} = \sum_{\ell \in \mathcal{L}} f(a_\ell) \geq |\mathcal{L}| \times f\left(\frac{\sum_{\ell \in \mathcal{L}} a_\ell}{|\mathcal{L}|}\right) \geq |\mathcal{L}| \times f\left(\frac{\log n}{|\mathcal{L}|}\right) = \frac{(T - 2)/2}{e^{(\log n)/2} - 1}
\]
Now recall that we have assumed that \( q > \log n \). So if \( T \leq 6 + 2 \log n \), then \( T \leq O(\sqrt{q \log n}) \) and we are done. Otherwise, for \( T \geq 6 + 2 \log n \), we have \( e^{\log q \log n} - 1 \leq \frac{e^{\log q \log n}}{2} \), and therefore

\[
\sum_{t \in L} \frac{j_{t+1} - j_t}{j_{t-1} - j_{t+1}} \geq \frac{(T - 2)/2 \times (T - 2)/2}{e \log n} \geq \Omega(T^2 / \log n)
\]

which further implies that \( q \geq \Omega(T^2 / \log n) \), i.e. that \( T \leq O(\sqrt{q \log n}) \) as desired.

Since \( |R| \leq T \), this immediately shows Theorem 13(c). Also, the expected sample complexity of each iteration \( t \) of FindRepresentatives is \( O(\log \frac{nq}{\Upsilon}) \). This expectation holds even conditioned on the entire past history, i.e., on all the randomness at iterations \( 1, \ldots, t - 1 \). Because of this, the overall expected sample complexity is \( \mathbb{E}[T] \times O(\log \frac{nq}{\Upsilon}) \). Since \( T \leq n + 1 \) with probability one, and \( T \leq O(r) \) when the execution is good, we have \( \mathbb{E}[T] \leq O(r) \) which yields Theorem 13(a).

### 6 Proof of Theorem 15: solving \( P_{\text{ratio}}^B \)

By telescoping products, the following expression holds for each \( i \in \{0, \ldots, t\} \):

\[
Q_{\beta_i} = \frac{Z(\beta_i)}{Z(\beta_{\min})} = C_i(\mathcal{I}) \cdot \frac{\mu_{\beta_0}(\sigma_0^+)}{\mu_{\beta_1}(\sigma_1^+)} \cdots \frac{\mu_{\beta_{i-1}}(\sigma_{i-1}^+)}{\mu_{\beta_i}(\sigma_i^+)} \cdots \frac{\mu_{\beta_{\max}}(\sigma_{\max}^+)}{\mu_{\beta_{\min}}(\sigma_{\min}^+)}
\]

where \( C_i(\mathcal{I}) \) is defined as \( \Theta(\frac{1}{\varepsilon^2}) \).

Let us suppose first that we have run algorithm FindSchedule, obtaining a schedule \( \mathcal{I} \) (hopefully proper) such that \( \text{InvWeight}(\mathcal{I}) \leq O(n \Gamma) \). Given set \( B \subseteq [\beta_{\min}, \beta_{\max}] \), consider the following algorithm to “weakly” solve \( P_{\text{ratio}}^B \) (we will explain more explicitly the guarantees it provides):

**Algorithm 4: Weakly solving \( P_{\text{ratio}}^B \).** **Input:** schedule \( \mathcal{I} = ((\beta_0, \sigma_0), \ldots, (\beta_t, \sigma_t)) \), parameter \( \varepsilon > 0 \), set \( B \subseteq [\beta_{\min}, \beta_{\max}] \). **Output:** estimates \( \hat{Q} = \{\hat{Q}_{\alpha}\}_{\alpha \in B} \).

1. **foreach** \( i \in \{0, \ldots, t\} \) **do**
   2. set \( \hat{\mu}_{\beta_i} \leftarrow \text{Sample}(\beta_i; \max\{\frac{\mathcal{I}\mathbb{E}[N]{\sigma_{i}^\text{weight}^2}}{\sigma_{i}^\text{weight}}, R(\frac{\varepsilon}{T}, \frac{1}{10}, \sigma_i^\text{weight})\}) \) where \( N = 1 + \frac{16}{(1 - e^{\varepsilon/2})^2} = \Theta(\frac{1}{\varepsilon^2}) \)
   3. set \( \hat{Q}_{\beta_i} = C_i(\mathcal{I}) \cdot \frac{\hat{\mu}_{\beta_0}(\sigma_0^+)}{\mu_{\beta_1}(\sigma_1^+)} \cdots \frac{\hat{\mu}_{\beta_i-1}(\sigma_{i-1}^+)}{\mu_{\beta_i}(\sigma_i^+)} \cdots \frac{\hat{\mu}_{\beta_{\max}}(\sigma_{\max}^+)}{\mu_{\beta_{\min}}(\sigma_{\min}^+)} \) \( \star \) treat division by 0 arbitrarily \( \star / \)
4. **foreach** \( \alpha \in B \setminus \{\beta_0, \ldots, \beta_t\} \) **do**
   5. select index \( i \) such that \( \alpha \in (\beta_i, \beta_{i+1}) \)
   6. let \( \hat{\mu}_{\alpha} \leftarrow \text{Sample}(\alpha; \frac{\varepsilon}{T}, \frac{1}{10}, \min\{\sigma_i^\text{weight}, \sigma_{i+1}^\text{weight}\}) \)
   7. set \( \hat{Q}_{\alpha} = \frac{\hat{\mu}_{\beta_i}(\sigma_i^+)}{\hat{\mu}_{\alpha}(\sigma_i^+)} e^{(\alpha - \beta_i)k} \hat{Q}_{\beta_i} \) where \( k = \sigma_i^+ = \sigma_{i+1}^- \)

We now need to show that Algorithm 4 succeeds with constant probability. The proof here uses standard techniques (e.g. cf. [11]). We use the following notation: for a random variable \( X \) we define

\[
S[X] = \mathbb{E}[X^2] / (\mathbb{E}[X])^2 = \frac{\text{var}(X)}{\mathbb{E}[X]^2} + 1
\]

The following fact is well-known (and easy to derive).

**Lemma 32** ([6, page 136]). For \( P = \prod_i P_i \) where the \( P_i \) are independent,

\[
\mathbb{E}[P] = \prod_i \mathbb{E}[P_i], \quad S[P] = \prod_i S[P_i]
\]

**Proposition 33.** For any \( i \in \{0, \ldots, t\} \), we have \( \mathbb{P}[\hat{Q}_{\beta_i} \text{ is an } \varepsilon/2 \text{-estimate of } Q_{\beta_i}] \geq \frac{7}{8} \).

**Proof.** Denote \( \varepsilon = 1 - e^{-\varepsilon/4} \) and \( \lambda = \frac{\varepsilon^2}{16} \), and consider \( (\beta, \sigma) \in \mathcal{I} \). For each \( k \in \text{Ends}(\sigma) \) the value \( \hat{\mu}_{\beta}(k) \) is a scaled binomial random variable with number of trials \( N_{\beta} \geq \left\lceil \frac{\varepsilon T}{\sigma_{\text{weight}}^2} \right\rceil = \left\lceil \frac{(1 + \lambda)T}{\lambda \sigma_{\text{weight}}} \right\rceil \) and success probability \( \mu_{\beta}(k) \geq \sigma_{\text{weight}} \).

Thus,

\[
\mathbb{E}[\hat{\mu}_{\beta}(k)] = \mu_{\beta}(k) \quad S[\hat{\mu}_{\beta}(k)] = 1 + \frac{1 - \mu_{\beta}(k)}{N_{\beta} \cdot \mu_{\beta}(k)} \leq 1 + \frac{1 - 0}{N_{\beta} \cdot \mu_{\beta}(k)} \cdot \sigma_{\text{weight}} = 1 + \frac{\lambda}{(1 + \lambda)T} \quad (8)
\]
Denote $W = \prod_{j=0}^{i-1} \mu_\beta_j(\sigma_j^+)$, $\hat{W} = \prod_{j=0}^{i-1} \hat{\mu}_\beta_j(\sigma_j^+)$, $V = \prod_{j=1}^{\hat{i}} \mu_\beta_j(\sigma_j^-)$, $\hat{V} = \prod_{j=1}^{\hat{i}} \hat{\mu}_\beta_j(\sigma_j^-)$. Here, $Q_{\beta_i} = C_i(\mathcal{I}) \cdot \frac{W}{V}$ and $\hat{W}, \hat{V}$ are random variables satisfying $\hat{Q}_{\beta_i} = C_i(\mathcal{I}) \cdot \frac{\hat{W}}{\hat{V}}$. From Lemma 32 and Eq. (8) we get

$$E[\hat{W}] = W \quad S[\hat{W}] \leq \left(1 + \frac{\lambda}{(1 + \lambda)t}\right)^t < e^{\frac{\lambda}{1 + \lambda}} < 1 + \lambda = 1 + \frac{\varepsilon^2}{16}$$

By Chebyshev’s inequality, $\mathbb{P}(|\hat{W}/W - 1| \geq \varepsilon) \leq (S[\hat{W}] - 1)/\varepsilon^2 < \frac{1}{16}$. Similarly, $\mathbb{P}(|\hat{V}/V - 1| \geq \varepsilon) < \frac{1}{16}$.

This in turn implies that $\hat{W}/W \in [e^{-\varepsilon/4}, e^{\varepsilon/4}]$ and $\hat{V}/V \in [e^{-\varepsilon/4}, e^{\varepsilon/4}]$ and thus $\hat{Q}_{\beta_i}/Q_{\beta_i} \in [e^{-\varepsilon/2}, e^{\varepsilon/2}]$. 

**Theorem 34.** If $\mathcal{I}$ is proper, then for every $\alpha \in B$, the estimate $\hat{Q}_\alpha$ provided by Algorithm 4 satisfies

$$\mathbb{P}[Q_\alpha/Q_\alpha \in [e^{-\varepsilon}, e^{\varepsilon}]] \geq \frac{3}{4}$$

**Proof.** If $\alpha = \beta_i$ for some $i$, then we have already shown this in Proposition 33. Otherwise, suppose $\alpha \in (\beta_i, \beta_{i+1})$. Let $k = \sigma_{i+1}^-$ and let $\omega = \min\{\sigma_i^{\text{weight}}, \sigma_{i+1}^{\text{weight}}\}$.

By Lemma 11, properness of $\mathcal{I}$, and Proposition 33, the following events hold with probability at least $1 - 1/16 - 1/8 - 1/16 = 3/4$: (1) $\hat{\mu}_\beta_i(k)$ is an $\varepsilon/4$-estimate of $\mu_\beta_i(k)$; (2) $\hat{Q}_{\beta_i}$ is an $\varepsilon/2$-estimate of $Q_{\beta_i}$; (3) line 6 well-estimates $k$.

Suppose these events all occur. Properness of $\mathcal{I}$ implies that $\mu_\beta_i(k) \geq \omega$ and $\mu_{\beta_{i+1}}(k) \geq \omega$. It is known [19, Proposition 3.1] that $\log Z(\beta)$ is a convex function of $\beta$. Therefore, function $\log \mu_\beta(k) = \log c_k + \beta k - \log Z(\beta)$ is concave, which implies that $\mu_\alpha(k) \geq \omega$ as well. Since line 6 well-estimates $k$, this implies that $\hat{\mu}_\alpha(k)$ is an $\varepsilon/4$-estimate of $\mu_\alpha(k)$. Since $Q_\alpha = \frac{\hat{\mu}_\alpha(k)}{\mu_\alpha(k)} e^{(\alpha - \beta_i)k} Q_{\beta_i}$, this shows that $\hat{Q}_\alpha$ is an $\varepsilon$-estimate of $Q_\alpha$.

We may use a technique known as median amplification to increase the success probability to $1 - \gamma$, for any desired parameter $\gamma$. We summarize this as follows:

**Proposition 35.** $P_{\text{ratio}}^B$ can be solved using expected sample complexity $O\left(\frac{|B|n^2 + n^2}{\varepsilon^2} \Gamma \log \frac{|B|}{\varepsilon^2} \log n \log q\right) + n \log q$.

**Proof.** We first call $\mathcal{I} \leftarrow \text{FindSchedule}(\gamma/2)$, and we then execute $s = \Theta(\log \frac{|B|}{\varepsilon})$ independent repetitions of Algorithm 4 with schedule $\mathcal{I}$. This yields estimates $\hat{Q}_\alpha^{(i)}$ for $i = 1, \ldots, s$. For each $\alpha \in B$, our final estimate $\hat{Q}_\alpha$ is the median of the values $\hat{Q}_\alpha^{(1)}, \ldots, \hat{Q}_\alpha^{(s)}$. A standard analysis shows that for each $\alpha$ it then holds that $\mathbb{P}[\hat{Q}_\alpha/Q_\alpha \in [e^{-\varepsilon}, e^{\varepsilon}] > 1 - \frac{\gamma}{16}$.

Now let us examine the complexity of this process. First, we claim that Algorithm 4 has sample complexity $O\left(\frac{|B|n + n^2}{\varepsilon^2} \Gamma \right)$. For, as $N = O\left(\frac{1}{\varepsilon^2}\Gamma \right)$ and $t \leq n + 1$, iteration $i$ of line 2 has sample complexity $O\left(\frac{n}{\sigma_i^{\text{weight}} \varepsilon^2}\right)$. The overall sampling complexity of line 2 is thus $\sum_{i=0}^{t} O\left(\frac{n}{\sigma_i^{\text{weight}} \varepsilon^2}\right) = O\left(\frac{n}{\varepsilon^2} \cdot \text{InvWeight}(\mathcal{I})\right)$. Likewise, each iteration of line 6 has sample complexity $O\left(\frac{n}{\varepsilon^2} \cdot \min\{\sigma_i^{\text{weight}}, \sigma_{i+1}^{\text{weight}}\}\right)$; as $\frac{1}{\sigma_i^{\text{weight}}} \leq \text{InvWeight}(\mathcal{I})$ for each $j$, the overall sample complexity of line 6 is $O\left(\frac{|B|}{\varepsilon^2} \cdot \text{InvWeight}(\mathcal{I})\right)$.

Since we need $s$ executions of Algorithm 4, these give a total expected complexity it contributes is $O\left(\frac{|B|}{\varepsilon^2} |B| \log n \log q\right)$. By Theorem 14, the call to FindSchedule contributes expected sample complexity $O(n \Gamma (\log n + \log \frac{1}{\gamma}) + n \log q)$.

Recall that an alternative algorithm for $P_{\text{ratio}}$ with expected sample complexity $O\left(\frac{\log n \log \frac{1}{\gamma}}{\varepsilon^2}\right)$ was proposed in [11, 15]. As we discuss in Appendix D, this algorithm can be adapted to solve $P_{\text{ratio}}^B$ with expected sample complexity $O\left(\frac{|B|n + n^2}{\varepsilon^2} \Gamma \log \frac{|B|}{\varepsilon^2} \log n \log q\right)$. 

22
The parameter \( q \) is not known, but we can still combine these algorithms using a technique known as dovetailing. Let us consider running the two algorithm simultaneously in parallel with error parameter \( \gamma/2 \); as soon as either algorithm terminates, we output its answer. This solves the problem with probability at least \( 1 - \gamma \), for, by the union bound, with probability at least \( 1 - \gamma/2 - \gamma/2 \), both of the two algorithms will (eventually) return a correct answer. The expected runtime of this procedure is at most twice the expected runtime of either algorithm individually. This gives Theorem 6.

7 Proof of Theorem 14: computing a schedule

We now describe the algorithm \texttt{FindSchedule}. Let us fix constants \( \tau \in (0, \frac{1}{2}) \), \( \lambda \in (0, 1) \), and denote \( \phi = \tau \lambda^3 / \Gamma \). Thus, \( \phi = \Theta(\frac{1}{\log n}) \) in the general setting and \( \phi = \Theta(1) \) in the log-concave setting.

The algorithm will maintain a sequence \( J = ((\beta_0, \sigma_0), \ldots, (\beta_t, \sigma_t)) \) of distinct extended weighted intervals satisfying the following invariants:

(I1) \( \beta_{\min} = \beta_0 \leq \ldots \leq \beta_t = \beta_{\max} \).

(I2) \( -\infty = \sigma^-_0 \leq \ldots \leq \sigma^-_t \leq n \) and \( 0 \leq \sigma^+_0 \leq \ldots \leq \sigma^+_t = +\infty \).

(I3) If \( \beta_{i-1} = \beta_i \) then either \( \sigma^-_{i-1} = \sigma^-_i \) or \( \sigma^+_{i-1} = \sigma^+_i \).

(I4) If \( \sigma^-_i = -\infty \) then \( \beta_i = \beta_{\min} \), and if \( \sigma^+_t = +\infty \) then \( \beta_t = \beta_{\max} \).

(I5) \( \sigma^{\text{weight}} \geq \frac{\phi}{\text{span}(\sigma)} \) for each \( (\beta, \sigma) \in J \).

For a sequence \( J \) we define \( J^U = \bigcup_{(\beta, \sigma) \in J} [\sigma^-, \sigma^+] \subseteq [\infty, +\infty] \). If \( J^U = [\infty, +\infty] \) and \( J \) satisfies (I1) - (I5), the sequence \( J \) is called a pre-schedule. In Section 7.1, we describe how to generate a pre-schedule satisfying two additional invariants; in Section 7.2 we show how to convert this into a proper schedule.

In order to define these two invariants, we say that interval \( (\beta, \sigma) \) is extremal if it satisfies the following conditions:

\[
\mu_{\beta}(k) \leq \frac{1}{\lambda} \cdot \frac{\text{span}(\sigma)}{\text{span}(\sigma) + (\sigma^- - k)} \cdot \mu_{\beta}(\sigma^-) \quad \forall k \in \{0, \ldots, \sigma^- - 1\} \tag{9a}
\]

\[
\mu_{\beta}(k) \leq \frac{1}{\lambda} \cdot \frac{\text{span}(\sigma)}{\text{span}(\sigma) + (k - \sigma^+)} \cdot \mu_{\beta}(\sigma^+) \quad \forall k \in \{\sigma^+ + 1, \ldots, n\} \tag{9b}
\]

We say that \( (\beta, \sigma) \) is left-extremal if it satisfies (9a) and right-extremal if it satisfies (9b). With this notation, we can state the additional invariants (I6), (I7) we hope to maintain.

(I6) Each interval \( (\beta, \sigma) \in J \) is proper.

(I7) Each interval \( (\beta, \sigma) \in J \) is extremal.

Note that conditions (I6), (I7) are defined in terms of the distribution \( \mu \), so they cannot be checked directly. We say that interval \( (\beta, \sigma) \) is conformant if it obeys all the conditions (I5) - (I7). We will later show how to convert a pre-schedule satisfying conditions (I1)-(I7) into a proper schedule.

It is convenient to denote \( L = \{\frac{1}{2}, \frac{3}{2}, \ldots, n - \frac{1}{2}\} \) and \( \text{Gaps}(J) = L - J^U \). Let \( \text{GapIntervals}(J) = \{(k + \frac{1}{2}, k + \frac{3}{2}, \ldots, \ell - \frac{1}{2}, \ell - \frac{3}{2}) \subseteq \text{Gaps}(J) : k, \ell \in J^U\} \) be the set of maximal discrete intervals in \( \text{Gaps}(J) \). Note that \( \text{Gaps}(J) = \bigcup_{\Theta \in \text{GapIntervals}(J)} \Theta \). From (I2), we see that for any \( \theta \in \text{Gaps}(J) \) there exists unique index \( i \in [t] \) with \( \sigma^-_i < \theta < \sigma^+_{i+1} \).
7.1 Generating a pre-schedule

This algorithm will use a key subroutine \( \text{FindInterval}(\beta, \mathcal{H}^-, \mathcal{H}^+) \) (complete details will be provided later). Given \( \beta \in [\beta_{\min}, \beta_{\max}] \) and subsets \( \mathcal{H}^-, \mathcal{H}^+ \subseteq \mathcal{H} \) with \( \max \mathcal{H}^- < \min \mathcal{H}^+ \), this must return weighted interval \( \sigma \) such that \( \sigma^- \in \mathcal{H}^- \), \( \sigma^+ \in \mathcal{H}^+ \), and \( \sigma_{\text{weight}} = \frac{\phi}{\text{span}(\sigma)} \). Ideally, the interval \( \sigma \) should also be proper and extremal.

With this subroutine, we can now formulate the algorithm to generate a proper pre-schedule with high probability.

**Algorithm 5:** Computing pre-schedule.

1. call \( \sigma_{\min} \leftarrow \text{FindInterval}(\beta_{\min}, \{-\infty\}, \mathcal{H}) \)
2. call \( \sigma_{\max} \leftarrow \text{FindInterval}(\beta_{\max}, \mathcal{H}, \{+\infty\}) \)
3. set \( \mathcal{J} = (\sigma_{\min}, \sigma_{\max}) \)
4. while \( \text{Gaps}(\mathcal{J}) \neq \emptyset \) do
   5.   pick arbitrary \( \Theta \in \text{GapIntervals}(\mathcal{J}) \), let \( \theta \in \Theta \) be a median value in \( \Theta \)
   6.   let \( (\beta_{\text{left}}, \sigma_{\text{left}}), (\beta_{\text{right}}, \sigma_{\text{right}}) \) be the unique consecutive pair in \( \mathcal{J} \) with \( \sigma_{\text{left}} < \theta < \sigma_{\text{right}} \)
   7.   call \( \beta \leftarrow \text{BinarySearch}(\beta_{\text{left}}, \beta_{\text{right}}, \theta, \frac{\Gamma}{\sqrt{n}}, \tau) \)
   8.   call \( \sigma \leftarrow \begin{cases} & \text{FindInterval}(\beta, [\sigma_{\text{left}}, \theta] \cap \mathcal{H}, [\theta, \sigma_{\text{right}}] \cap \mathcal{H}) \text{ if } \beta_{\text{left}} < \beta < \beta_{\text{right}} \\ & \text{FindInterval}(\beta, [\sigma_{\text{left}}, \theta] \cap \mathcal{H}, [\sigma_{\text{right}}^+, \sigma_{\text{right}}^-] \cap \mathcal{H}) \text{ if } \beta = \beta_{\text{left}} \\ & \text{FindInterval}(\beta, [\sigma_{\text{left}}, \theta] \cap \mathcal{H}, [\sigma_{\text{left}}, \sigma_{\text{right}}] \cap \mathcal{H}) \text{ if } \beta = \beta_{\text{right}} \\ & \text{FindInterval}(\beta, [\sigma_{\text{left}}, \theta] \cap \mathcal{H}, [\sigma_{\text{right}}, \sigma_{\text{left}}] \cap \mathcal{H}) \text{ if } \beta = \beta_{\text{right}} \end{cases} \)
   9.   insert \((\beta, \sigma)\) into \( \mathcal{J} \) between \((\beta_{\text{left}}, \sigma_{\text{left}})\) and \((\beta_{\text{right}}, \sigma_{\text{right}})\)
10. return \( \mathcal{J} \)

By specification of the subroutines, the algorithm preserves invariants (11) – (15) and produces a pre-schedule upon termination. Note that property (13) ensures that \( \beta_{\text{left}} < \beta_{\text{right}} \) in each iteration \( i \), as is required by \text{BinarySearch}. Furthermore, the loop in lines 4 – 9 is executed at most \( |\mathcal{J}| = n \) times, so the algorithm makes at most \( n + 2 \) calls to \text{FindInterval} and at most \( n \) calls to \text{BinarySearch}.

We say the call \( \sigma \leftarrow \text{FindInterval}(\beta, \mathcal{H}^-, \mathcal{H}^+) \) is good if interval \((\beta, \sigma)\) is proper and extremal. If we execute \text{FindInterval} with arbitrary inputs it is very likely that no good output exists. The overall structure of Algorithm 5 has been carefully designed so that, as long as invariants (11)–(17) have been satisfied so far and calls to \text{BinarySearch} have been good, then the output of \text{FindInterval} will be good with high probability. More specifically, let us say that a call to \text{FindInterval} at line 8 is valid if \( \beta \in \Lambda_{\tau}(\beta_{\text{left}}, \beta_{\text{right}}, \theta) \), interval \((\beta_{\text{left}}, \sigma_{\text{left}})\) is conformant, and interval \((\beta_{\text{right}}, \sigma_{\text{right}})\) is conformant. We also say that the calls to \text{FindInterval} at line 1 and 2 are valid.

Let us fix some constant \( \kappa \in (0, 1) \). The following result summarizes \text{FindInterval}.

**Theorem 36.** \text{FindInterval}(\beta, \mathcal{H}^-, \mathcal{H}^+) has sample complexity \( O(\Gamma \log n \times \text{span}(\mathcal{H}^- \cup \mathcal{H}^+)) \). If the call is valid, then the call is good with probability at least \( 1 - \frac{\kappa}{2(n+2)} \).

We will prove this later in Section 7.3. Putting this aside for the moment, we show the following result for Algorithm 5.

**Proposition 37.** Algorithm 5 has expected sample complexity \( O(n \log q + n \Gamma \log^2 n) \), and the resulting pre-schedule \( \mathcal{J} \) is proper with probability at least \( 1 - \kappa \).

**Proof.** First, note that if all calls to \text{BinarySearch} and \text{FindInterval} are good, then the resulting pre-schedule \( \mathcal{J} \) maintains properties (16) and (17). In particular, by property (16), it is proper. Since \text{BinarySearch} or \text{FindInterval} fail with probability at most \( \frac{\kappa}{2n} \) and \( \frac{\kappa}{2(n+2)} \) respectively, a simple union bound shows that properties (16) and (17) are maintained with probability at least \( 1 - \kappa \).

We thus need to show the bound on the expected sample complexity. By Theorem 12 and bearing mind that \( \kappa = O(1) \), the subroutines \text{BinarySearch} have expected sample complexity \( O(n \log(nq)) \). Let us show that subroutines \text{FindInterval} have sample complexity \( O(n \Gamma \log^2 n) \). Let \( \Theta_i, \theta_i, \sigma_{\text{left},i}, \sigma_{\text{right},i} \) be the variables at the \( i \)th iteration and \( \mathcal{J}_i \) be the sequence at the beginning of
this iteration. Define \( \mathcal{H}_i = [\sigma_{left,i}, \sigma_{right,i}] \cap \mathcal{H} \). By Theorem 36(a), the \( i \)th iteration of FindInterval has sample complexity \( O(\Gamma |\mathcal{H}_i| \log n) \). We will show next that \( \sum_i |\mathcal{H}_i| = O(n \log n) \), which will yield the claim about the complexity.

At each iteration \( \ell \) we add a new interval \( \sigma_i \) intersecting \( \Theta_\ell \). This means that interval \( \Theta_\ell \) is removed from \( \text{GapIntervals}(J_\ell) \) and is replaced by two new intervals \( \Theta', \Theta'' \), which are subsets of \( \Theta_\ell \), in \( \text{GapIntervals}(J_{\ell+1}) \). Furthermore, \( \theta_\ell \) is the median of \( \Theta_\ell \) and after iteration \( \ell \) the value \( \theta_\ell \pm \frac{1}{2} \) is added to \( J_\ell \), so \( |\Theta'| \leq \frac{1}{2} |\Theta_\ell| \) and \( |\Theta''| \leq \frac{1}{2} |\Theta_\ell| \). As a consequence of this, the intervals \( \Theta_\ell \) have the property that for \( i < j \) we have

\[
\Theta_i \cap \Theta_j \neq \emptyset \Rightarrow |\Theta_j| \leq \frac{1}{2} |\Theta_i|
\]

(10)

For \( k \in \mathcal{H} \) define \( I^{-}(k) = \{ i : k \in \mathcal{H}_i \land \theta_i < k \} \), and consider \( i, j \in I^{-}(k) \) with \( i < j \). We claim that \( \theta_j \in \Theta_i \). Indeed, suppose not. Since the endpoints of \( \Theta_i \) equal \((\sigma_{left,i} + \frac{1}{2}, \sigma_{right,i} - \frac{1}{2}) \) and \( \theta_j \) cannot belong to \( \sigma_{left,i} \) or \( \sigma_{right,i} \), one the following must hold:

- \( \theta_j < \sigma_{left,i} \). Condition \( i, j \in I^{-}(k) \) implies that \( \sigma_{left,i} < k \leq \sigma_{right,i} \) and so by property (12) interval \( \sigma_{left,i} \) comes before \( \sigma_{right,i} \) in sequence \( J_j \). This is a contradiction since the algorithm chooses \( \sigma_{right,i} \) as the leftmost interval in \( J_j \) satisfying \( \theta_j < \sigma_{right,i} \).

- \( \theta_j > \sigma_{right,i} \). Condition \( i, j \in I^{-}(k) \) implies that \( \theta_j < k \leq \sigma_{right,i} \), again a contradiction.

Thus \( \theta_j \in \Theta_i \) and so \( \Theta_i \cap \Theta_j \neq \emptyset \). By Eq. (10) this implies that \( |\Theta_j| \leq \frac{1}{2} |\Theta_i| \). Since this holds for all pairs \( i, j \in I^{-}(k) \), we conclude that \( |I^{-}(k)| \leq \lceil \log_2 |\mathcal{L}| \rceil + 1 = O(\log n) \).

In a similar way we can show that \( |I^{+}(k)| = O(\log n) \) where \( I^{+}(k) = \{ i : k \in \mathcal{H}_i \land \theta_i > k \} \). It remains to observe that \( \sum_i |\mathcal{H}_i| = \sum_{k \in \mathcal{H}} |I^{-}(k) \cup I^{+}(k)| \).

### 7.2 Converting a pre-schedule into a schedule

Having formed a pre-schedule, we next need to convert it to a proper schedule, using a subroutine \( \text{FinalizeSchedule}(J, \gamma) \). This procedure \( \text{FinalizeSchedule} \) also validates its input: it is allowed to output an error code \( \perp \), and in particular even if \( J \) is not proper, it should still be unlikely that \( \text{FinalizeSchedule} \) returns a non-proper schedule — it should output \( \perp \) in this case.

**Algorithm 6:** Algorithm FindSchedule\((\gamma)\)

1. **while** true do
2. 1. call Algorithm 5 with parameter \( \kappa = 1/2 \) to compute pre-schedule \( J \)
3. 2. call \( I \leftarrow \text{FinalizeSchedule}(J, \gamma/4) \)
4. 3. if \( I \neq \perp \) then return \( I \)

We will show that \( \text{FinalizeSchedule}(J, \gamma) \) can be implemented to have the following behavior:

**Theorem 38.** (a) The output \( I \) is either a schedule with \( \text{InvWeight}(I) \leq 2e^\nu(n+1)/\phi \), for an arbitrary constant \( \nu > 0 \), or is the error code \( \perp \).
(b) For any input \( J \), with probability at least \( 1 - \gamma \), the output \( I \) is either \( \perp \) or a proper schedule \( I \).
(c) If the input \( J \) is proper, then with probability at least \( 1 - \gamma \), the output \( I \) is a proper schedule.
(d) The sample complexity is \( O(n \Gamma \log \frac{1}{2}\gamma) \).

To complete the proof, we prove Theorem 36 (the implementation of FindInterval) in Section 7.3 and Theorem 38 (the implementation of FinalizeSchedule) in Section 7.4. Assuming these results for the moment, we can combine all our algorithmic results to show Theorem 14.

**Proof of Theorem 14.** Proposition 37 and Theorem 38 show that each iteration of Algorithm 6 terminates with probability at least \((1 - \kappa)(1 - \gamma/4) \geq \Omega(1) \). Therefore, the expected number of runs is \( O(1) \). Each call to \( \text{FinalizeSchedule} \) has sample complexity \( O(n \Gamma \log \frac{1}{2}\gamma) \). Each iteration of Algorithm 6 has sample complexity \( O(n \log q + n \Gamma \log n) \). Thus, the overall expected sample complexity of Algorithm 6 is \( O(n \Gamma \log \frac{1}{2}\gamma + n \log q + n \Gamma \log^2 n) \).

25
By Theorem 38(a), we have \( \text{InvWeight}(I) \leq 2e^\nu(n+1)/\phi = 2\Gamma(n+1) \times e^\nu/\tau^\lambda \). The term \( e^\nu/\tau^\lambda \) gets arbitrarily close to 2 for constants \( \nu, \lambda, \tau \) sufficiently close to 0, 1, \( 1/2 \) respectively.

Finally, let us show that the output value \( I \) of \text{FindSchedule}(\gamma) \) is proper with probability at least \( 1 - \gamma \). Let \( \tilde{I} \) denote the value obtained at line 5 of any given iteration. Since the iterations of Algorithm 5 are independent, the distribution of \( I \) is the same as the distribution of \( \tilde{I} \), conditioned on \( \tilde{I} \neq \bot \). Thus \( P[I \text{ is an improper schedule}] = P[\tilde{I} \text{ is an improper schedule} | \tilde{I} \neq \bot] \).

By Theorem 38(b), the probability that \( \tilde{I} \) is an improper schedule is at most \( \gamma/4 \), even conditioned on any fixed value for the pre-schedule \( J \). By Proposition 37(b), in any given iteration the pre-schedule \( J \) is proper with probability at least \( \kappa = 1/2 \); in such case, by Theorem 38(b), we have \( \tilde{I} \neq \bot \) with probability at least \( 1 - \gamma \geq 1/2 \). Overall, we have \( P[\tilde{I} \neq \bot] \geq 1/4 \). We therefore have

\[
P[\tilde{I} \text{ is an improper schedule} | \tilde{I} \neq \bot] \leq \frac{P[\tilde{I} \text{ is an improper schedule}]}{P[\tilde{I} \neq \bot]} \leq \frac{\gamma/4}{1/4} = \gamma
\]

### 7.3 Proof of Theorem 36(a): Procedure \text{FindInterval}(\beta, \mathcal{H}^-, \mathcal{H}^+)

To describe the algorithm, let us define \( h^- = \min \mathcal{H}^-, a^- = \max \mathcal{H}^- + 1, a^+ = \min \mathcal{H}^+ - 1, \) and \( h^+ = \max \mathcal{H}^+ \). We also set \( \gamma = \frac{\kappa}{2(\alpha+2)} \).

**Algorithm 7: FindInterval(\beta, \mathcal{H}^-, \mathcal{H}^+).**

1. let \( \mu_\beta \leftarrow \text{Sample}(\beta; \frac{1}{2} \log \frac{1}{A}, \frac{\gamma}{\mu+1}, p_o) \) where \( p_o = \frac{\phi}{\text{span}[h^-, h^+]} \)
2. foreach \( i \in \mathcal{H} \) set \( \alpha(i) = \begin{cases} 
1 & \text{if } i \in \{-\infty, +\infty\} \\
\lambda^{3/2} \cdot \mu_\beta(i) & \text{if } i \in \mathcal{H} - \{h^-, h^+\} \\
\lambda^{1/2} \cdot \mu_\beta(i) & \text{if } i \in \mathcal{H} \cap \{h^-, h^+\} 
\end{cases} \)
3. set \( k^- = \text{arg max}_{i \in \mathcal{H}^-} (a^- - i) \alpha(i) \) and \( k^+ = \text{arg max}_{i \in \mathcal{H}^+} (i - a^+) \alpha(i) \)
4. return \( \sigma = \{(k^-, k^+); \phi \cdot \frac{\lambda^{1/2} \cdot \mu_\beta(i)}{\text{span}[h^-, h^+]} \}

The cases when \text{FindInterval} is called from Algorithm 5 in line 1 or line 2, or in line 8 when \( \beta \in \{\beta_{\text{left}}, \beta_{\text{right}}\} \), are handled very differently from the main case, which is line 8 with \( \beta \in (\beta_{\text{left}}, \beta_{\text{right}}) \) strictly. In these special cases, there is no “free choice” for the left margin \( k^- = \sigma^- \) or right-margin \( k^+ = \sigma^+ \) respectively. We say that the call to \text{FindInterval} at line 1, or the call at line 8 with \( \beta = \beta_{\text{left}}, \) is left-forced; the call at line 2, or at line 8 with \( \beta = \beta_{\text{right}} \) is right-forced. Otherwise the call is left-free and right-free respectively.

In the unforced case, we give a slight bias to the endpoints \( h^- \) or \( h^+ \); this helps preserve the slack factor \( \frac{1}{\lambda} \) in the definition of extremality (11a), (9b). (Without this bias, the factor would grow uncontrollably as the algorithm progresses.) In the forced cases, desired properties of \( \sigma \) (namely, extremality and properness) instead follow from the corresponding properties of \( \sigma_{\text{left}} \) or \( \sigma_{\text{right}} \).

The sample complexity is \( O(\text{span}(h^-, h^+)) \log n) \) (bearing in mind that \( \lambda = O(1) \) and \( \gamma = 1/\text{poly}(n) \)). The interval \( \sigma \) clearly satisfies property (15). The non-trivial thing to check is that if the call is valid, then \( \sigma \) is extremal and proper with probability at least \( 1 - \gamma \).

For the remainder of this section, let us therefore suppose that the call is valid. So either we are executing \text{FindInterval} at line 1 or 2 in Algorithm 5 or \( \beta \in \Lambda_{\tau}(\beta_{\text{left}}, \beta_{\text{right}}, \theta) \), and intervals \((\beta_{\text{left}}, \sigma_{\text{left}})\) and \((\beta_{\text{right}}, \sigma_{\text{right}})\) are both conformant.

Let us first state a useful formula.

**Lemma 39.** There holds

\[
\mu_{\beta}(i) \leq \frac{1}{\lambda} \cdot \frac{j - h^-}{j - i} \cdot \mu_{\beta}(h^-) \quad \forall i \in \{0, \ldots, h^- - 1\}, \forall j \in \{a^-, a^- + 1, \ldots, n\} \quad (11a)
\]

\[
\mu_{\beta}(i) \leq \frac{1}{\lambda} \cdot \frac{h^+ - j}{i - j} \cdot \mu_{\beta}(h^+) \quad \forall i \in \{h^+, \ldots, n\}, \forall j \in \{0, \ldots, a^+ - 1, a^+\} \quad (11b)
\]
Proof. We only show (11a); the proof of (11b) is analogous. If we are calling FindIntervals at line 1 of Algorithm [5] then $h^* = -\infty$ and the claim is vacuous. Likewise, if we are calling FindInterval at line 2 of Algorithm [5] then $a^- = n + 1$ and the claim is vacuous.

So assume that we are calling FindIntervals at line 8 of Algorithm [5] and interval $\sigma_{\text{left}}$ is well-defined. Now consider $i < h^-$ and $j \geq a^-$. Since $(\beta_{\text{left}}, \sigma_{\text{left}})$ is left-extremal and $h^- = \sigma_{\text{left}}^-$, we have

$$
\mu_{\beta_{\text{left}}}(i) \leq \frac{1}{\lambda} \cdot \frac{\text{span}(\sigma_{\text{left}})}{\text{span}(\sigma_{\text{left}}) + (h^- - i)} \cdot \mu_{\beta_{\text{left}}}(h^-)
$$

(12)

Since $i < h^-$ and $\beta \geq \beta_{\text{left}}$, Eq. (3) gives $\mu_{\beta_{\text{left}}}(i) \mu_{\beta}(h^-) \geq \mu_{\beta_{\text{left}}}(h^-) \mu_{\beta}(i)$. Combined with Eq. (12), this yields

$$
\mu_{\beta}(i) \leq \frac{1}{\lambda} \cdot \frac{\text{span}(\sigma_{\text{left}})}{\text{span}(\sigma_{\text{left}}) + (h^- - i)} \cdot \mu_{\beta}(h^-)
$$

Finally, since $j \geq a^- \geq \sigma_{\text{left}}^+ + 1$ we have $\text{span}(\sigma_{\text{left}}) \leq j - h^-$ and therefore

$$
\frac{\text{span}(\sigma_{\text{left}})}{\text{span}(\sigma_{\text{left}}) + (h^- - i)} \leq \frac{j - h^-}{(j - h^-) + (h^- - i)} = \frac{j - h^-}{j - i}
$$

We need another existential result on some values of $\mu_{\beta}$. Note that this is the only place in the analysis that we need to distinguish between the general setting (where $\phi = \Theta(1/\log n)$) and the log-concave setting (where $\phi = \Theta(1)$).

**Lemma 40.** In both the general or log-concave settings, the following holds:

(a) If the call is left-free, then there exists $k \in \mathcal{H}^-$ with $(a^- - k) \cdot \mu_{\beta}(k) \geq \tau \lambda / \Gamma = \phi / \lambda^2$.

(b) If the call is right-free, then there exists $k \in \mathcal{H}^+$ with $(k - a^+) \cdot \mu_{\beta}(k) \geq \tau \lambda / \Gamma = \phi / \lambda^2$

**Proof.** The two claims are completely analogous, so we only prove (a). Denote $\mathcal{A} = \{0, \ldots, a^- - 1\}$ and $\delta = \max_{k \in \mathcal{A}} (a^- - k) \cdot \mu_{\beta}(k)$. We make the following claim:

$$
\mu_{\beta}(\mathcal{A}) \leq \Gamma \delta
$$

(13)

Indeed, if we denote $b_i = \frac{\mu_{\beta}(a^- - i)}{\delta}$ for $i = 1, \ldots, a^-$, then the definition of $\delta$ implies that $b_i \leq \frac{1}{\Gamma}$ for all $i = 1, \ldots, a^-$. Also, we have $\mu_{\beta}(\mathcal{A}) = \delta \sum_{i=1}^{a^-} b_i$. Now consider two possible cases.

- **Log-concave setting (with $\Gamma = \gamma$).** If coefficients $c_k$ are log-concave then so is the sequence $b_1, \ldots, b_{a^-}$ (since $\mu_{\beta}(k) \propto c_k e^{\beta k}$). Lemma [10] then gives $\sum_{i=1}^{a^-} b_i \leq \gamma = \Gamma$.

- **General setting (with $\Gamma = 1 + \log(n + 1)$).** We have $\sum_{i=1}^{a^-} b_i \leq 1 + \log a^- \leq 1 + \log(n + 1) = \Gamma$ by the well-known inequality for the harmonic series.

From now on we assume that (a) is false, i.e. $(a^- - k) \cdot \mu_{\beta}(k) < \frac{\tau \lambda}{\Gamma}$ for all $k \in \mathcal{H}^-$. If we are calling FindInterval at line 2 of Algorithm [5] then $\mathcal{H}^- = \mathcal{A} = \mathcal{H}$. Thus $\delta < \frac{\tau \lambda}{\Gamma}$. From Eq. (13) we have $\mu_{\beta}(\mathcal{H}) \leq \Gamma \delta < \tau \lambda < \frac{1}{\Gamma} \cdot 1$, which is a contradiction since $\mu_{\beta}(\mathcal{H}) = 1$.

Now suppose that we are calling FindIntervals at line 8 of Algorithm [5]. We claim that the following holds:

$$
\mu_{\beta}(k) < \frac{\tau}{\Gamma} \cdot \frac{1}{a^- - k} \quad \text{for all } k \in \mathcal{A}
$$

(14)

Indeed, we already have the stronger inequality $\mu_{\beta}(k) < \frac{\tau \lambda}{\Gamma} \cdot \frac{1}{a^- - k}$ for $k \in \mathcal{H}^-$. In particular, we know $\mu_{\beta}(h^-) < \frac{\tau \lambda}{\Gamma} \cdot \frac{1}{a^- - h^-}$. It remains to show Eq. (14) for some $k < h^-$. Eq. (11a) with $(i,j) = (k,a^-)$ gives

$$
\mu_{\beta}(k) \leq \frac{1}{\lambda} \cdot \frac{a^- - h^-}{a^- - k} \mu_{\beta}(h^-)
$$
Using our bound on $\mu_\beta(h^-)$, we now get the desired claim:

$$
\mu_\beta(k) < \frac{1}{\lambda} \cdot \frac{a^- - h^-}{a^- - k} \times \frac{\tau \lambda}{\Gamma} \cdot \frac{1}{a^- - h^-} = \frac{\tau}{\Gamma} \cdot \frac{1}{a^- - k}
$$

Eq. ([14]) implies that $\delta < \frac{1}{\Gamma}$. So from Eq. ([13]) we get $\mu_\beta(A) < \tau$. On the other hand, since the call is left-free, we have $\beta > \beta_{\text{left}}$. We assumed that $\beta \in \Lambda\gamma(\beta_{\text{left}}, \beta_{\text{right}}, \theta)$, and therefore $\mu_\beta([0, \theta]) \geq \tau$. This is a contradiction, since $[0, \theta] \cap \mathcal{H} = A$. 

We are now ready to show that $\text{FindInterval}$ is good with probability at least $1 - \gamma$. We have already assumed that the call is valid; let us also suppose that line 1 well-estimates every construction, this holds with probability at least $1 - \gamma$. We will show that under this condition, the output interval $\sigma$ is extremal and proper.

**Proposition 41.** (a) If the call is left-free, we have $(a^- - k^-) \cdot \alpha(k^-) \geq \phi$ and $\mu_\beta(k^-) \geq \sqrt{\lambda} \cdot \hat{\mu}_\beta(k^-)$. (b) If the call is right-free, we have $(k^+ - a^+) \cdot \alpha(k^+) \geq \phi$ and $\mu_\beta(k^+) \geq \sqrt{\lambda} \cdot \hat{\mu}_\beta(k^+)$. 

**Proof.** We only prove (a); the case (b) is completely analogous.

By Lemma [10] there exists $k \in \mathcal{H}^-$ with $(a^- - k^-)\mu_\beta(k^-) \geq \phi/\lambda^2$. Note that $\mu_\beta(k^-) \geq \frac{\phi}{\lambda^2(a^- - k^-)} \geq \frac{\phi}{\lambda^2} > p_\phi$; since line 1 well-estimates $k$, this implies that $\hat{\mu}_\beta(k^-) \geq \frac{\sqrt{\lambda} \cdot \mu_\beta(k^-)}{\sqrt{\lambda}}$. Therefore $\alpha(k^-) \geq \frac{\phi}{\lambda^2}$. Since $k^-$ is chosen as the argmax, this means that $(a^- - k^-)\alpha(k^-) \geq (a^- - k^-)\alpha(k) \geq \phi$.

This further implies that $\hat{\mu}_\beta(k^-) \geq \frac{\alpha(k^-)}{\sqrt{\lambda}} \geq \frac{\phi}{\sqrt{\lambda(a^- - k^-)}} \geq p_\phi$. Since $k^-$ is well-estimated, this implies that $\mu_\beta(k^-) \geq \sqrt{\lambda} \hat{\mu}_\beta(k^-)$. 

**Proposition 42.** Interval $\sigma$ is proper.

**Proof.** We need to show that if $k^- \neq -\infty$ then $\mu_\beta(k^-) \geq \frac{\phi}{\text{span}(\sigma)}$ and likewise if $k^+ \neq +\infty$ then $\mu_\beta(k^+) \geq \frac{\phi}{\text{span}(\sigma)}$. We will show only the former; the latter is completely analogous. Two cases are possible.

- The call is left-free. We have $\text{span}(\sigma) = \min\{k^+ + 1, n + 1\} - k^- \geq a^- - k^-$. By Proposition [11] we have $(a^- - k^-)\alpha(k^-) \geq \phi$ and $\mu_\beta(k^-) \geq \sqrt{\lambda} \hat{\mu}_\beta(k^-)$. Since $\hat{\mu}_\beta(k^-) \geq \alpha(k^-)/\sqrt{\lambda}$, this implies that $(a^- - k^-)\mu_\beta(k^-) \geq \phi$.

- The call is left-forced. In this case, as $k^- \neq -\infty$, necessarily $\mathcal{H}^- = \{\sigma_{\text{left}}^-, \beta = \beta_{\text{left}}\}$ and $k^- = \sigma_{\text{left}}^-$. Since interval $\sigma_{\text{left}}^-$ is conformant, we have $\mu_\beta(k^-) \geq \sigma_{\text{left}}^\text{weight} \geq \frac{\phi}{\text{span}(\sigma_{\text{left}}^-)}$. Note now that $\sigma \supseteq \sigma_{\text{left}}^-$, and so $\mu_\beta(k^-) \geq \frac{\phi}{\text{span}(\sigma)}$ as desired. 

**Proposition 43.** Interval $\sigma$ is extremal.

**Proof.** We only verify that the interval is left-extremal; the proof of right-extremality is completely analogous. We can assume that $k^- \geq 1$, otherwise there is nothing to show. Let $\ell = \min\{n + 1, k^+ + 1\}$, so that $\text{span}(\sigma) = \ell - k^-$. Note $\ell \geq a^-$. We thus need to prove that

$$
\mu_\beta(i) \leq \frac{1}{\lambda} \cdot \frac{\ell - k^-}{\ell - i} \cdot \mu_\beta(k^-) \quad \forall i \in \{0, \ldots, k^- - 1\}
$$

(15)

Two cases are possible.

Case 1: $k^- = h^-$. Then Eq. ([11a]) with $j = \ell$ gives Eq. ([15]).

Case 2: $k^- > h^-$. The call must be left-free since $k^-, h^- \in \mathcal{H}^-$. For $i \in \{h^-, \ldots, k^-\}$ define

$$
\rho_i = \begin{cases} 
\lambda^{1/2} & i = h^- \\
\lambda^{3/2} & i > h^- 
\end{cases}
$$
so that \( \alpha(i) = \hat{\mu}_\beta(i) \rho_i \). By definition of \( k^- \), we have \( (a^- - i) \alpha(i) \leq (a^- - k^-) \alpha(k^-) \), i.e.

\[
\hat{\mu}_\beta(i) \leq \frac{(a^- - k^-) \alpha(k)}{\rho_i (a^- - i)} \tag{16}
\]

We can show that the RHS here is at least \( p_0 \). For, by Proposition \[41\] we have \( (a^- - k^-) \alpha(k^-) \geq \phi \) and so \( \frac{(a^- - k^-) \alpha(k^-)}{\rho_i (a^- - i)} \geq \frac{\phi}{\lambda^2 \rho_i S} \geq \frac{\phi}{\lambda S} > p_0 \). Since line 1 well-estimates \( i \), this in turn implies that

\[
\mu_\beta(i) \leq \frac{(a^- - k^-) \alpha(k^-)}{\rho_i (a^- - i)}
\]

Proposition \[41\] shows that \( \hat{\mu}_\beta(k^-) \leq \mu_\beta(k^-)/\sqrt{\lambda} \). Since \( k^- \neq h^- \), we have \( \alpha(k^-) = \lambda^{3/2} K_\beta(k^-) \).

We also have \( \ell \geq a^- \). Combining all these bounds, we have shown that

\[
\mu_\beta(i) \leq \frac{(\ell - k^-) \lambda^{1/2} \mu_\beta(k^-)}{\rho_i (\ell - i)} \tag{17}
\]

For \( i \in \{h^- + 1, \ldots, k^- - 1\} \), we have \( \rho_i = \lambda^{3/2} \), and so Eq. \[17\] shows that \( \mu_\beta(i) \leq \frac{(\ell - k^-) \mu_\beta(k^-)}{\lambda (\ell - i)} \), which establishes Eq. \[15\]. For \( i = h^- \), we have \( \rho_i = \lambda^{1/2} \) and so Eq. \[17\] shows

\[
\mu_\beta(h^-) \leq \frac{(\ell - k^-) \mu_\beta(k^-)}{\ell - h^-} \tag{18}
\]

which again establishes Eq. \[15\]. Finally, for \( i \in \{0, \ldots, h^- - 1\} \), Eq. \[11\a\] with \( j = \ell \) gives

\[
\mu_\beta(i) \leq \frac{1}{\lambda} \cdot \frac{\ell - h^-}{\ell - i} \cdot \mu_\beta(h^-)
\]

Combined with Eq. \[18\], this immediately establishes Eq. \[15\]. \( \square \)

### 7.4 Proof of Theorem 38: Procedure FinalizeSchedule(\( J, \gamma \))

In this section we assume that \( J \) is a given pre-schedule. The algorithm FinalizeSchedule has two parts. First, we transform \( J \) into a minimal pre-schedule \( J' \). We then “uncross” \( J' \) to get a schedule.

The algorithm for minimizing \( J \) is very simple, and has zero sample complexity:

**Algorithm 8: MinimizePreschedule(\( J \))**

1. initialize \( J' = J \)
2. while there exists \((\beta, \sigma) \in J'\) such that \((J' - (\beta, \sigma))^{-1} = [\infty, \infty] \) do
   3. update \( J' \leftarrow J' - (\beta, \sigma) \) for any such \((\beta, \sigma)\)
3. return \( J' \)

The resulting pre-schedule \( J' \) has the following nice properties:

**Proposition 44.** For a pre-schedule \( J \) let \( J' = \text{MinimizePreschedule}(J) = ((\beta_0, \sigma_0), \ldots, (\beta_t, \sigma_t)) \).

Then, \( J' \) is also a pre-schedule and InvWeight(\( J' \)) \( \leq \frac{2(n+1)}{\phi} \). If \( J \) is proper then so is \( J' \). Furthermore, for \( i = 0, \ldots, t - 1 \) we have \( \sigma_i^- < \sigma_{i+1}^- \leq \sigma_i^+ < \sigma_{i+1}^+ \).

**Proof.** Clearly \( J'^{-1} = J'^{-1} = [-\infty, \infty] \). \( J' \) satisfies properties \((13), (14), (15)\), since \( J \) does so and we are only removing intervals from \( J \). This same reasoning also shows that if \( J \) is proper then so is \( J' \).

Since \( J' \) is a subsequence of \( J \) with \( J'^{-1} = [-\infty, \infty] \), property \((12)\) implies that \( -\infty = \sigma_0^- \leq \sigma_1^- \leq \cdots \leq \sigma_i^- \leq \sigma_i^+ \leq \cdots \leq \sigma_t^+ \leq +\infty \). So \( J' \) satisfies \((12)\).

Since \( J \) satisfies property \((11)\), it is clear that \( \beta_0 \leq \beta_1 \leq \cdots \leq \beta_t \). Furthermore, by property \((15)\) only intervals of the form \((\sigma, \beta_{\text{min}})\) in \( J \) can cover \( -\infty \); thus \( \beta_0 = \beta_{\text{min}} \). A similar argument shows that \( \beta_t = \beta_{\text{max}} \). So \( J' \) satisfies property \((11)\). We have now shown that \( J' \) is a pre-schedule.
To show that \( \sigma_i^- < \sigma_{i+1}^- \), suppose for contradiction that \( \sigma_i^- = \sigma_{i+1}^- \). Since \( \sigma_{i+1}^+ \geq \sigma_i^+ \) by property (12), this implies \( \sigma_i \leq \sigma_{i+1} \). In particular, removing interval \( \sigma_i \) would not change the value of \( J^{\nu} \), contradicting minimality of \( J' \). A similar argument shows that \( \sigma_i^+ < \sigma_{i+1}^+ \).

To show that \( \sigma_{i+1} \leq \sigma_i^+ \), suppose for contradiction that \( \sigma_i^+ < \sigma_{i+1}^+ \), and let \( \theta \in [\sigma_i^+, \sigma_{i+1}^-] \). Note that \( \sigma_j^- < \sigma_i^+ \) for \( j < i \) and \( \sigma_j^+ > \sigma_{i+1}^+ \) for \( j > i \). Thus, \( \theta \notin J^{\nu} \), contradicting that \( J^{\nu} = [-\infty, \infty] \).

Finally, since \( J' \) is minimal, each \( k \in H \) is covered in at most two intervals. So \( \sum_{i=0}^{t} \frac{\text{span}(\sigma_i)}{\phi} \leq 2(n+1) \). By (14), we have \( \frac{1}{\sigma_i^{\text{weight}}} \leq \frac{\text{span}(\sigma_i)}{\phi} \) for each \( i \), so \( \text{InvWeight}(J') = \sum_{i} \frac{\text{span}(\sigma_i)}{\phi} \leq \frac{2(n+1)}{\phi} \). 

The minimized pre-schedule \( J' \) satisfies all the constraints of a schedule, except that the intervals may cross each other. The second part of the algorithm, described below in Algorithm 9, fixes this. Here \( \nu > 0 \) is some arbitrary constant.

**Algorithm 9: FinalizeSchedule(\( J, \gamma \)) for pre-schedule \( J \).**

1. set \( J' = \text{MinimizePreschedule}(J) = ( (\beta, \sigma_0), \ldots, (\beta, \sigma_t) ) \)
2. foreach \( i \in \{0, \ldots, t\} \) let \( \hat{\mu}_i \leftarrow \text{Sample}(\beta_i; \nu/2, \frac{\gamma}{2(n+1)}, e^{-\nu/2} \sigma_i^{\text{weight}}) \)
3. set \( s_0 = -\infty \) and \( s_{t+1} = +\infty \)
4. foreach \( i \in \{1, \ldots, t\} \) do
   5. if \( \exists k \in \{ \sigma_i^- \), \( \sigma_i^+ \} \) s.t. \( \hat{\mu}_{i-1}(k) \geq e^{-\nu/2} \sigma_{i-1}^{\text{weight}} \) and \( \hat{\mu}_i(k) \geq e^{-\nu/2} \sigma_i^{\text{weight}} \) then
   6. set \( s_i = k \) for arbitrary such \( k \)
   7. else output \( \perp \)
8. return the schedule \( \mathcal{I} = ( (\beta, (s_i, s_{i+1}], e^{-\nu} \sigma_i^{\text{weight}}) : i = 0, \ldots, t ) \)

**Proof of Theorem 52**. The sample complexity is \( O(\text{InvWeight}(J') \log \frac{2}{\phi}) = O(n \Gamma \log \frac{2}{\phi}) \), and the algorithm returns either \( \perp \) or a sequence \( \mathcal{I} \) with \( \text{InvWeight}(\mathcal{I}) \leq e^\nu \text{InvWeight}(J') \leq 2e^\nu(n+1)/\phi \).

The bound \( \sigma_i^- < \sigma_{i+1}^- \leq \sigma_i^+ < \sigma_{i+1}^+ \) shown in Proposition 44 implies \( -\infty < s_1 < \cdots < s_t < +\infty \). Thus \( \mathcal{I} \) is indeed a schedule. We need to argue that the output is good with high probability. Let us suppose that each iteration \( i \) of line 2 well-estimates \( \sigma_i^+, \sigma_i^- \), \( \sigma_{i+1}^- \), \( \sigma_{i+1}^+ \). Since we use error parameter \( \frac{\gamma}{2(n+1)} \), this has probability at least \( 1 - \gamma \). We show that, under this condition, we output either a proper schedule or \( \perp \); furthermore, if \( J \) is proper, then we output a proper schedule \( \mathcal{I} \).

We first claim that if we return schedule \( \mathcal{I} \), it is proper. We need to show that \( \mu_i(s_i) \geq e^{-\nu} \sigma_i^{\text{weight}} \) for \( i \geq 1 \) and \( \mu_i(s_{i+1}) \geq e^{-\nu} \sigma_i^{\text{weight}} \) for \( i \leq t - 1 \). For the former, note that \( s_i = k \) where \( k \) is some interval that \( \hat{\mu}_i(k) \geq e^{-\nu/2} \sigma_i^{\text{weight}} \). Since line 2 well-estimates \( k \), this implies that \( \mu_i(k) \geq e^{-\nu} \sigma_i^{\text{weight}} \) as required. The case for \( \mu_i(s_{i+1}) \) is completely analogous.

Next, we argue that if \( J \) is proper then we do not output \( \perp \). Suppose we do so at iteration \( i \), and let \( \ell = s_i^- \) and \( k = s_{i+1}^- \) where \( \hat{\mu}_{i-1}(k) < e^{-\nu/2} \sigma_{i-1}^{\text{weight}} \) and \( \hat{\mu}_i(\ell) < e^{-\nu/2} \sigma_i^{\text{weight}} \). Since \( k, \ell \) are well-estimated, this implies \( \mu_{i-1}(k) < \sigma_i^{\text{weight}} \) and \( \mu_i(\ell) < \sigma_i^{\text{weight}} \). On the other hand, \( J' \) is proper if \( J \) is, and so \( \mu_{i-1}(k) \geq \sigma_i^{\text{weight}} \) and \( \mu_i(\ell) \geq \sigma_i^{\text{weight}} \).

Therefore, \( \mu_{i-1}(k) \mu_i(\ell) < \sigma_i^{\text{weight}} \sigma_i^{\text{weight}} \leq \mu_{i-1}(k) \mu_i(\ell) \). By Proposition 44 we have \( k \leq \ell \), so this contradicts Eq. (3). 

8 Applications

There is a pervasive close connection between sampling and counting. Consider the following scenario: we have a collection of objects of various sizes, and we would like to estimate the number \( C_i \) of objects of size \( i \). If we can sample from the Gibbs distribution on these objects (weighted by their size), then our algorithm allows us to convert this sampling procedure into a counting procedure.

In an number of combinatorial applications, we further know that the counts \( C_i \) are log-concave; for example, the matchings in a graph [10], or the number of independent sets in a matroid [11]. One main motivation for our focus on log-concave coefficients is indeed to handle these combinatorial situations.
In the context of log-concave coefficients, there are natural choices for certain parameters for our algorithm which lead to particularly clean bounds:

**Theorem 45.** Suppose coefficients \( \{c_k\}_{k \in \mathcal{H}} \) are log-concave and non-zero. If we select appropriate values \( \beta_{\min} \leq \log \frac{c_k}{c_1}, \mu_*, \frac{1}{n+1} \) and \( \beta_{\max} \geq \log \frac{c_{n+1}}{c_n} \), then when we execute Theorem 4 we get the following results where \( F := \max\{\beta_{\max}, \log \frac{c_n}{c_0}, 1\} \):

(a) We obtain a \((\varepsilon, \mathcal{H})\)-estimate of \( c \) with probability at least \( 1 - \gamma \).

(b) We have \( q \leq O(nF) \).

(c) The expected sample complexity is \( O\left(\min\left\{\frac{nF \log n \log \frac{n}{\varepsilon^2}}{\varepsilon^2}, \frac{n^2 \log n}{\varepsilon^2} + n \log F\right\}\right) \).

**Proof.** Define \( b_i = c_{i-1}/c_i \) for \( i = 1, \ldots, n \). Since \( c_i \) is log-concave, the sequence \( b_1, \ldots, b_n \) is non-decreasing. Let us first show the following fact: for each \( i, k \in \mathcal{H} \), we have the bound

\[
 c_i e^{i \log b_i} \geq c_k e^{k \log b_i} \tag{19}
\]

To show this for \( k > i \), we use the fact the sequence \( b_j \) is non-decreasing to compute:

\[
 \frac{c_i e^{i \log b_i}}{c_k e^{k \log b_i}} = e^{(i-k) \log b_i} \prod_{j=i}^{k-1} \frac{c_j}{c_{j+1}} = \exp \left( \sum_{j=i}^{k-1} \log b_{j+1} - \log b_j \right) \geq 1
\]

A similar calculation applies for \( k < i \). Since \( \mu_\beta(k) \propto c_k e^{\alpha k} \), Eq. (19) shows that \( \mu_{\log b_i}(i) \geq \frac{1}{n+1} \).

Also, since sequence \( b_i \) is non-decreasing, we have \( \log b_i \in [\log b_0, \log b_n] \subseteq [\beta_{\min}, \beta_{\max}] \) for \( i \geq 1 \). By similar reasoning, we have \( \frac{\mu_{\log b_i}(0)}{b_i} \geq \frac{1}{n+1} \). Therefore, with \( \mu^* = \frac{n}{n+1} \), we have \( \mathcal{H}^* = \mathcal{H} \), and so we have shown (a).

We next turn to part (b). To begin, we can lower-bound \( Z(\beta_{\min}) \) as

\[
 Z(\beta_{\min}) = \sum c_i e^{i \beta_{\min}} \geq c_0 e^{0 \cdot \beta_{\min}} = c_0
\]

To upper-bound \( Z(\beta_{\max}) \), we observe that for every \( k \leq n \), we have

\[
 \frac{c_n e^{n \beta_{\max}}}{c_k e^{k \beta_{\max}}} = \frac{c_n e^{n b_n}}{c_k e^{k b_n}} \times e^{(\beta_{\max} - b_n)(n-k)}
\]

By Eq. (19), we have \( \frac{c_n e^{n b_n}}{c_k e^{k b_n}} \geq 1 \) and by hypothesis we have \( \beta_{\max} \geq b_n \). Therefore, \( c_n e^{n \beta_{\max}} \geq c_k e^{k \beta_{\max}} \) for every \( k \leq n \), and so we bound \( Z(\beta_{\max}) \) as

\[
 Z(\beta_{\max}) = \sum c_i e^{i \beta_{\max}} \leq (n+1) c_n e^{n \beta_{\max}}
\]

Thus we estimate

\[
 q = \frac{Z(\beta_{\max})}{Z(\beta_{\min})} \leq \frac{e^{n \beta_{\max}(n+1)c_n}}{c_0}
\]

Here, the ratio \( c_n/c_0 \) telescopes as:

\[
 \frac{c_n}{c_0} = \prod_{i=1}^{n} \frac{c_i}{c_{i-1}} = \prod_{i=1}^{n} \left( \frac{1}{b_i} \right) \leq \prod_{i=1}^{n} \left( \frac{1}{b_1} \right) = \left( \frac{c_1}{c_0} \right)^n
\]

giving

\[
 q \leq e^{n \beta_{\max} \times (n+1) \times (c_1/c_0)^n} \leq e^{nF} \times (n+1) \times e^{nF}
\]

This implies \( q \leq O(nF) \).

With this value of \( q \) and \( \mu_\ast \), we get the stated sample complexity. \( \square \)

### 8.1 Counting connected subgraphs

Consider a connected graph \( G = (V, E) \). For each \( i = |V| - 1, \ldots, |E| \) let \( N_i \) denote the number of connected subgraphs of \( G \) with \( i \) edges.

In [9], Guo & Jerrum described an algorithm to sample a connected subgraph \( G' = (V, E') \) with probability proportional to \( \prod_{f \in E'} (1 - p(f)) \prod_{f \in E - E'} p(f) \), for some weighting function \( p : E \rightarrow [0,1] \). This can be interpreted probabilistically as each edge \( f \) “failing” independently with probability \( p(f) \), and conditioning on the resulting subgraph remaining connected; here \( E - E' \) is the set of failed edges. If we set \( p(f) = \frac{1}{1 + e^{\beta f}} \) for all edges \( f \), then the resulting distribution on connected subgraphs is a Gibbs distribution, with rate \( \beta \) and with coefficient sequence given by \( c_i = N_{|E|-i} \).

Guo & He [8] subsequently improved the algorithm runtime; we summarize their result as follows:
Observe that \( \beta \) is close to the Gibbs distribution. It is traditional in FPRAS algorithms to take an estimate \( \beta \) of \( \beta \) and \( \beta \) as well. The number of coefficients in the Gibbs distribution is \( n = |E| - |V| + 1 \). Also, note that \( c_{n-1}/c_n \) and \( c_1/c_0 \) are both at most \( |E| \), since to enumerate a connected graph with \( |V| \) edges we may select a spanning tree and any other edge in the graph, and to enumerate a graph with \( |E| - 1 \) edges we simply select an edge of \( G \) to delete. Therefore, we apply Theorem 45 setting \( \beta_{\text{max}} = \log |E| > \log \frac{c_{n-1}}{c_n}, \beta_{\text{min}} = -\log |E| \leq \log \frac{c_0}{c_1}, \) and hence \( F = \log |E| \).

So Theorem 45 shows that we need \( O(n \log |E|) \log^2 n \log n \log ^2 \epsilon^2 \) samples. It is traditional in analyzing FPRAS to take \( \gamma = O(1) \), and since \( n = |E| \) we overall use \( O(|E| \log^3 |E| / \epsilon^2) \) samples. With these parameters \( \beta_{\text{min}}, \beta_{\text{max}} \), Theorem 46 shows that each call to the sampling oracle has cost \( O(|E|^2 |V|) \).

The work [2] sketches an FPRAS for this problem as well; the precise complexity is unspecified and appears to be much larger than Theorem 4. We also note that Anari et al. [2] provide a general FPRAS for counting the number of independent sets in arbitrary matroids, which would include the number of connected subgraphs. This uses a very different sampling method, which is not based on the Gibbs distribution. They also do not provide concrete complexity estimates for their algorithm.

### 8.2 Counting matchings

Consider a graph \( G = (V, E) \) with \( |V| = 2v \) nodes which has a perfect matching. For \( i = 0, \ldots, n = v \), let \( M_i \) denote the number of \( i \)-edge matchings. Since \( G \) has a perfect matching these are all non-zero. As originally shown in [10], the sequence \( M_i \) is log-concave.

In [12, 13], Jerrum & Sinclair described an MCMC algorithm to approximately sample from the Gibbs distribution on matchings. To rephrase their result in our terminology:

**Theorem 47 ([13]).** There is an algorithm to approximately sample from the Gibbs distribution with coefficients \( c_i = M_i \) for any value \( \beta \); the expected runtime is \( \hat{O}(|E||V|^2 (1 + e^\beta) \log 1/\delta) \) to get within a total variation distance of \( \delta \).

There remains one complication to applying Theorem 45 for general graphs, the ratio between the number of perfect and near-perfect matchings, i.e. the ratio \( M_{v-1}/M_v \), could be exponential in \( n \). This would cause the parameter \( F \) to be too large in applying Theorem 45. This is the reason for our required bound on the ratio \( M_{v-1}/M_v \). With this stipulation, we prove Theorem 45.

**Proof of Theorem 45.** Observe that \( M_0 = 1 \), and so if we can estimate the coefficients \( c_i \), then we can estimate \( M_i \) as well. The number of coefficients in the Gibbs distribution is given by \( n = |V|/2 = v \).

For the first result, we determine the sample complexity needed for Theorem 45. To do so, we observe that \( c_{n-1}/c_0 \) is at most \( f \) by assumption, and \( c_1/c_0 \) is clearly at most \( |E| \). Therefore, we set \( \beta_{\text{min}} = -\log |E| \), \( \beta_{\text{max}} = \log f \), and \( F \leq \max \{ \log |E|, \log f \} \). So Theorem 45 shows that we need \( O(n \log |E| |f| \log n \log n \log n / \epsilon^2) \) samples.

By Corollary 3 we take \( \delta = \text{poly}(1/n, 1/f, \epsilon, \gamma) \) to ensure that the sampling oracle is sufficiently close to the Gibbs distribution. It is traditional in FPRAS algorithms to take \( \gamma = O(1) \). With these choices, Theorem 47 requires \( O(|E||V|^2 f \log polylog(|V|, f, 1/\epsilon)) \) time per sample. Overall, our FPRAS has runtime of \( \hat{O}(|E||V|^3 f / \epsilon^2) \).

For the second result, [12] showed that if \( G \) has minimum degree at least \( |V|/2 \), then \( M_v > 0 \) and \( M_{v-1}/M_v \leq f = O(|V|^2) \). Also, clearly \( |E| \leq O(|V|^2) \).
9 Lower bounds on sample complexity

In [13], Kolmogorov showed lower bounds on the sample complexity of a generalization of \( P_{\text{ratio}} \). This is based on an “indistinguishability” lemma, wherein a target distribution \( \mu^{(0)} \) (a coefficient vector) is surrounded by an envelope of alternate probability distributions \( \mu^{(1)}, \ldots, \mu^{(d)} \), which all use the same ground set \( \mathcal{H} = \{0, \ldots, n\} \) and the same values \( \beta_{\min}, \beta_{\max} \). The lemma establishes a lower bound on the sample complexity needed to distinguish between Gibbs distributions with these different coefficients. In this section, we adapt this construction to show lower bounds on \( P_{\text{ratio}} \) and \( P_{\text{coef}}^{\mu_*} \) for a wider variety of parameters.

Let us define \( \mu_{\beta}(k \mid c^{(r)}) \) to be the Gibbs distributions with parameter \( \beta \) under the coefficient vectors \( c^{(r)} \). We also define \( q^{(r)} \) to be the corresponding value of \( q \) for distribution \( c^{(r)} \). For some parameter \( \mu_* \) (which will common to all distributions \( c^{(0)}, \ldots, c^{(d)} \), we likewise define \( \mathcal{H}^{(r)} \) to the set \( \mathcal{H}^{(r)} \) with respect to distribution \( c^{(r)} \).

For any \( k \in \mathcal{H} \), let us define

\[
U_{\beta}(k) = \prod_{r=1}^{d} \mu_{\beta}(k \mid c^{(r)}) = \prod_{r=1}^{d} \frac{c^{(0)}_k Z(\beta \mid c^{(r)})}{c^{(r)}_k Z(\beta \mid c^{(0)})}
\]

and let us define the key parameter

\[
\Psi = \max_{\beta \in [\beta_{\min}, \beta_{\max}], k \in \mathcal{H}} \log U_{\beta}(k)
\]

Lemma 48 ([15]). Let \( \Lambda \) be a randomized algorithm which generates a set of queries \( \beta_1, \ldots, \beta_T \in [\beta_{\min}, \beta_{\max}] \) and receives values \( K_1, \ldots, K_T \), wherein each \( K_i \) is drawn from distribution \( \mu_{\beta_i} \). At some point the procedure stops and either outputs either TRUE or FALSE. The queries \( \beta_i \) may be adaptive and may be randomized, and the stopping time \( T \) may also be randomized.

Suppose that, with probability at least \( 1 - \gamma \) algorithm \( \Lambda \) outputs TRUE on input \( c^{(0)} \), whereas with probability at least \( 1 - \gamma \) it outputs FALSE on inputs \( c^{(1)}, \ldots, c^{(d)} \), for some parameter \( \gamma < 1/4 \).

Then the expected sample complexity of \( \Lambda \) on instance \( c^{(0)} \) is \( \Omega\left(\frac{d \log(1/\gamma)}{\Psi}\right) \).

This lemma implies lower bounds on the sampling problems \( P_{\text{ratio}} \) and \( P_{\text{coef}}^{\mu_*} \):

Corollary 49. (a) Suppose that \( |q^{(0)} - q^{(r)}| > 2\varepsilon \) for all \( r = 1, \ldots, d \). Then any algorithm to solve \( P_{\text{ratio}} \) must have expected sampling complexity \( \Omega\left(\frac{d \log(1/\gamma)}{\Psi}\right) \) on problem instance \( c^{(0)} \).

(b) Fix some parameter \( \mu_* \) and some set \( \mathcal{K} \subseteq \mathcal{H} \). Suppose that for each \( r = 1, \ldots, d \) there exists parameters \( i, j \in \mathcal{K}^{(0)} \) such that \( |\log(c^{(0)}_i / c^{(0)}_j) - \log(c^{(r)}_i / c^{(r)}_j)| > 2\varepsilon \). Then any algorithm to solve \( P_{\text{coef}}^{\mu_*, \mathcal{K}} \) must have expected sampling complexity \( \Omega\left(\frac{d \log(1/\gamma)}{\Psi}\right) \) on problem instance \( c^{(0)} \). Note that \( i, j \) may depend on the value \( r \).

Proof. (a) Whenever \( P_{\text{ratio}} \) succeeds on problem instance \( c^{(0)} \), the estimate \( \hat{q} \) is within \( \pm \varepsilon \) of \( q^{(0)} \). Whenever \( P_{\text{ratio}} \) succeeds on problem instance \( c^{(r)} \), the estimate \( \hat{q} \) is within \( \pm \varepsilon \) of \( q^{(r)} \), and consequently it is not within \( \pm \varepsilon \) of \( q^{(0)} \). Thus, solving \( P_{\text{ratio}} \) allows us to distinguish \( c^{(0)} \) from \( c^{(1)}, \ldots, c^{(d)} \).

(b) Let us solve \( P_{\text{coef}}^{\mu_*, \mathcal{K}} \), obtaining estimate \( \hat{c} \). If there exists any pair \( i, j \in \mathcal{K}^{(0)} \) such that either \( \hat{c}_i = 0, \hat{c}_j = 0 \), or \( |\log(\hat{c}_i / \hat{c}_j) - \log(c^{(0)}_i / c^{(0)}_j)| > \varepsilon \) then we output FALSE; otherwise we output TRUE.

When run on problem instance \( c^{(0)} \), it holds with probability at least \( 1 - \gamma \) that the vector \( \hat{c} \) is an \( (\varepsilon, \mathcal{K}^{(r)}) \) estimate of \( c \). In this case, by definition, this procedure will output TRUE.

When run on problem instance \( c^{(r)} \), again with probability at least \( 1 - \gamma \) the vector \( \hat{c} \) is an \( (\varepsilon, \mathcal{K}^{(r)}) \) estimate of \( c \). In this case, let \( i, j \) be the pair guaranteed by the hypothesis. By definition, we either have \( \hat{c}_i = 0, \hat{c}_j = 0 \), or the value \( \hat{c}_i / \hat{c}_j \) is an \( \varepsilon \)-estimate of the true value \( c^{(r)}_i / c^{(r)}_j \). In all three of these cases, the procedure will output FALSE.

Thus, solving \( P_{\text{coef}}^{\mu_*, \mathcal{K}} \) allows us to distinguish \( c^{(0)} \) from \( c^{(1)}, \ldots, c^{(d)} \).
By constructing appropriate problem instances and applying Corollary 49, we will show the following lower bounds on the sampling problems:

**Theorem 50.** Let \( n \geq 2, \varepsilon < \varepsilon_{\max}, \gamma < \gamma_{\max}, q \geq q_{\min}, \mu_{\ast} \leq \mu_{\ast,\max} \), where \( \mu_{\ast,\max}, \varepsilon_{\max}, \gamma_{\max}, q_{\min} \) are some universal constants.

(a) Any algorithm to solve \( P_{\text{ratio}}^{\mu_{\ast}, \mathcal{H}} \) on log-concave problem instances with these parameters must have expected sample complexity

\[
\Omega\left( \min\{q, n^{2}\} \log \frac{1}{\varepsilon^{2}} \right)
\]

(b) Any algorithm to solve \( P_{\text{coef}}^{\mu_{\ast}, \mathcal{H}} \) on log-concave problem instances with these parameters must have expected sample complexity

\[
\Omega\left( \frac{1}{\mu_{\ast}} + \min\{q, n^{2}\} \log \frac{1}{\varepsilon^{2}} \right)
\]

(c) Any problem to solve \( P_{\text{coef}}^{\mu_{\ast}, \mathcal{H}} \) on general problem instances with these parameters must have expected sample complexity

\[
\Omega\left( \min\left\{ q + \frac{\sqrt{q}}{\mu_{\ast}}, n^{2} + \frac{n}{\mu_{\ast}} \right\} \log \frac{1}{\varepsilon^{2}} \right)
\]

### 9.1 Bounds for \( P_{\text{coef}} \) in terms of \( \mu_{\ast} \) in the log-concave setting

The construction here is very simple: we set \( \beta_{\min} = 0 \), and \( n = 1 \). We have three choices for the coefficients, namely \( c_{0}^{(0)} = 2\mu_{\ast}, c_{0}^{(1)} = 2\mu_{\ast}e^{-3\varepsilon}, c_{0}^{(2)} = 2\mu_{\ast}e^{3\varepsilon} \). In all three cases, we set \( c_{i}^{(i)} = 1 \). We can also add dummy extra coefficients \( c_{i} = 0 \) for \( i = 2, \ldots, n \). Note that \( c^{(0)} \) has log-concave coefficients.

Since \( Z(\beta_{\max}) \) is a continuous function of \( \beta_{\max} \) with \( Z(\infty) = \infty \), we can ensure that this problem instance has the desired value of \( q \) by setting \( \beta_{\max} \) sufficiently large.

This allows us to show one of the lower bounds of Theorem 50.

**Proposition 51.** Under the conditions of Theorem 50, any procedure to solve \( P_{\text{coef}}^{\mu_{\ast}, \mathcal{H}} \) for log-concave problem instances must have expected sample complexity \( \Omega\left( \frac{\log(1/\varepsilon)}{\mu_{\ast} \varepsilon^{2}} \right) \)

**Proof.** We show this using Corollary 49(b) with parameters \( i = 0, j = 1 \). It is clear that \( |\log(c_{i}^{(0)}/c_{j}^{(0)}) - \log(c_{i}^{(r)}/c_{j}^{(r)})| > 2\varepsilon \), and that 0, 1 \( \in \mathcal{H}^{(0)} \) with respect to parameter \( \mu_{\ast} \).

We need to compute the parameter \( \Psi \). We begin by computing \( Z(\beta \mid c^{(r)}) \) as:

\[
Z(\beta \mid c^{(0)}) = 2\mu_{\ast} + e^{\beta}, \quad Z(\beta \mid c^{(1)}) = 2\mu_{\ast}e^{-3\varepsilon} + e^{\beta}, \quad Z(\beta \mid c^{(2)}) = 2\mu_{\ast}e^{3\varepsilon} + e^{\beta}
\]

and thus

\[
U_{\beta}(0) = U_{\beta}(1) = \frac{(2\mu_{\ast}e^{-3\varepsilon} + e^{\beta})(2\mu_{\ast}e^{3\varepsilon} + e^{\beta})}{(2\mu_{\ast} + e^{\beta})^{2}} = 1 + \frac{2\mu_{\ast}e^{\beta}(e^{3\varepsilon} + e^{-3\varepsilon} - 2)}{(2\mu_{\ast} + e^{\beta})^{2}}
\]

Simple calculus shows that this is a decreasing function of \( \beta \) for \( \beta \geq 0 \). So its maximum value in the interval \([\beta_{\min}, \beta_{\max}]\) occurs at \( \beta = 0 \) and

\[
\Psi = \log U_{\beta}(0) = \log \left( 1 + \frac{2\mu_{\ast}(e^{3\varepsilon} + e^{-3\varepsilon} - 2)}{(1 + 2\mu_{\ast})^{2}} \right) \leq 2\mu_{\ast}(e^{3\varepsilon} + e^{-3\varepsilon} - 2) \leq O(\mu_{\ast} \varepsilon^{2})
\]

So by Corollary 49(b), \( P_{\text{coef}}^{\mu_{\ast}, \mathcal{H}} \) on instance \( c^{(0)} \) requires expected sample complexity \( \Omega\left( \frac{\log(1/\varepsilon)}{\mu_{\ast} \varepsilon^{2}} \right) \). □
9.2 Bounds for $P_{\text{coef}}$ in terms of $\mu_*$ in the general setting

In this construction, let us set a parameter $t$ (which we will determine later). We set $c_{2i}^{(0)} = 2^{-i^2}$ for $i = 0, \ldots, t$ and $c_{2i+1}^{(0)} = 2^{-i^2} \times 8\mu_*$ for $i = 0, \ldots, t-1$. The remaining coefficients $c_{2i+1}^{(0)}, \ldots, c_n^{(0)}$ are set to zero.

We will define $d = 2t$ related problem instances; for each index $i = 0, \ldots, t-1$, we construct a problem instance where we set $c_{2i+1}^{(2i)} = c_{2i+1}^{(0)} e^{\nu}$, and all other coefficients agree with $c^{(0)}$; we also create a problem instance where we set $c_{2i+1}^{(2i+1)} = c_{2i+1}^{(0)} e^{-\nu}$, and all other coefficients agree with $c^{(0)}$.

We select $\beta_{\min} = 0$; the parameter $\beta_{\max}$ will be specified later.

**Proposition 52.** For $\nu \leq O(1)$, the problem instances $c^{(0)}, \ldots, c^{(d)}$ have $\psi \leq O(\mu_* \nu^2)$.

**Proof.** Given value $\beta \in [\beta_{\min}, \beta_{\max}]$, let us compute $U_\beta(k)$ as:

$$U_\beta(k) = \prod_{r=1}^d \frac{Z(\beta | c^{(r)})}{Z(\beta | c^{(0)})} = \prod_{r=1}^d \frac{Z(\beta | c^{(r)})}{Z(\beta | c^{(0)})}$$

$$= \prod_{i=0}^{t-1} \frac{(e^{-\nu} - 1)2^{-i^2} \times 8\mu_* e^{(2i+1)\beta} + Z(\beta | c^{(0)})}{Z(\beta | c^{(0)})} \frac{(e^{-\nu} - 1)2^{-i^2} \times 8\mu_* e^{(2i+1)\beta} + Z(\beta | c^{(0)})}{Z(\beta | c^{(0)})}$$

$$\leq \exp\left(\frac{(e^{-\nu} - 2) \times 8\mu_* \sum_{i=0}^{t-1} 2^{-i^2} e^{(2i+1)\beta}}{Z(\beta | c^{(0)})}\right)$$

Let us define $S_i = 2^{-i^2} e^{(2i+1)\beta}$ and $S = \sum_{i=0}^{t-1} S_i$. We claim that

$$\frac{S}{Z(\beta | c^{(0)})} \leq O(1) \quad (20)$$

Note that $Z(\beta | c^{(0)}) \geq \sum_{i=0}^{t-1} c_{2i+1} e^{(2i+1)\beta} \geq \frac{1}{2} \sum_{i=0}^{t-1} Z_i$, where we define $Z_i = 2^{-i^2} e^{(2i)\beta} + 2^{-i^2} e^{(2i+1)\beta}$. Thus in order to show Eq. (20), it suffices to show that $S_i \leq O(Z_i)$ for all $i = 0, \ldots, t-1$. For this, we compute:

$$S_i = \frac{2^{-i^2} e^{(2i+1)\beta}}{2^{-i^2} e^{(2i)\beta} + 2^{-i^2} e^{(2i+1)\beta}} = \frac{2^{-i^2} e^{\beta}}{1 + 2^{-i\beta} e^{2\beta}} = \frac{2^{-i^2} e^{\beta}}{1 + (2^{-i\beta})^2/2} \leq 1/\sqrt{2}$$

This shows Eq. (20), and as $\nu \leq O(1)$ we have $\log U_\beta(k) \leq (e^{-\nu} - 2) \times 8\mu_* \leq O(\mu_* \nu^2)$. \hfill \Box

**Proposition 53.** Given some parameter $\nu \leq \nu_{\max}$, where $\nu_{\max}$ is a sufficiently small constant, it is possible to select the parameter $t \geq \Omega(\min\{n, \sqrt{n}\})$ so that the problem instance $c^{(0)}$ has the required values of $q$ and $n$ and so that $\{0, 1, 3, 5, \ldots, 2t - 1\} \subseteq \mathcal{H}^{(0)}$.

**Proof.** We will set $\beta_{\max} \geq t \log 2$, for some parameter $t$ to be chosen. By taking $t \leq n/2$, we ensure that the coefficients are in the range $\{0, \ldots, n\}$. We need to select $\beta_{\max}, t$ to ensure that problem instance has $q = q_0$ for a given target value $q_0$.

When $\beta_{\max} = t \log 2$, the problem instance $c^{(0)}$ has

$$Q = \frac{Z(\beta_{\max})}{Z(\beta_{\min})} = \frac{\sum_{i=0}^{t-1} 2^{-i^2} e^{2i\beta_{\max}} + \sum_{i=0}^{t-1} 2^{-i^2} e^{2i+1)\beta_{\max}} \times 8\mu_*}{1}$$

Simple calculus shows that these summands are increasing at a super-constant rate, and thus the sums can be bounded by their value at maximum index,

$$Q \leq O(2^{-t^2} e^{2\beta_{\max} t} + 2^{-t^2} e^{(2t-1)\beta_{\max}} \times 8\mu_*) \leq O(2^{t^2} + 2^t \times \mu_* \times (2/e)^t) \leq O(2^t)$$

35
So \( q \leq t^2 \log 2 + O(1) \). This implies that, by selecting \( t \leq a \sqrt{\log n} \) for some sufficiently small constant \( a \), we can ensure that \( q \leq q_0 \) for \( \beta_{\max} = t \log 2 \). By continuity, this in turn implies that we get \( q = q_0 \) for some choice \( \beta_{\max} \geq t \log 2 \).

Suppose now we have fixed such \( t \) and \( \beta_{\max} \). Let us show that a given coefficient \( 2k+1 \) is in \( \mathcal{H}^{s(0)} \). To witness this, take \( \beta = k \log 2 \in [0, \beta_{\max}] \). For this, we have:

\[
Z(\beta | c^{(0)}) = \sum_{i=0}^{t} 2^{-i^2} e^{i^2 \beta} + \sum_{i=0}^{t-1} 2^{-i^2} e^{2i+i+1)\beta} \times 8\mu_* = \sum_{i=0}^{t} 2^{2i+k-i^2} + 8\mu_* \sum_{i=0}^{t-1} 2^{-i^2+(2i+1)k}
\]

It is easy to see that in the first sum, the summands decay at rate at least 1/2 away from the peak value \( i = k \), while the in the second sum the summands decay at rate 1/4 from their peak values at \( i = k, k-1 \). So \( Z(\beta | c^{(0)}) \leq 3 \times 2^{k^2} + 8\mu_* \times \frac{8}{3} 2^{k^2} \), which is smaller than \( 2^{k^2+2} \) for \( \mu_* \) sufficiently small. So we get

\[
\mu_\beta(2k+1 | c^{(0)}) = \frac{c^{(0)}_{2k+1} e^{2k+1)\beta}}{Z(\beta | c^{(0)})} \geq \frac{2^{-k-k^2} e^{(2k+1)\beta} \times 8\mu_*}{2^{k^2+2}} \geq \mu_*
\]

A similar analysis with \( \beta = 0 \) shows that \( 0 \in \mathcal{H}^{s(0)} \) as well.

**Proposition 54.** Under the conditions of Theorem 51, any procedure to solve \( P_{\mu_\ast}^{H} \) for general problem instances must have expected sample complexity \( \Omega(\frac{\log(1/\gamma) \min\{n, \sqrt{q}\}}{\mu_\ast \varepsilon^2}) \).

**Proof.** Construct the problem instance with \( t = \Omega(\min\{\sqrt{q}, n\}) \) which has the desired parameters \( n, q \) and where we set \( \nu = 3\varepsilon \), for \( \varepsilon \leq \varepsilon_{\max} \) sufficiently small. Consider some \( r \in \{1, \ldots, d\} \). For this instance, we have \( |\log(c_i^{(0)}/c_j^{(0)}) - \log(c_i^{(r)}/c_j^{(r)})| = \nu > 2\varepsilon \) where \( i = 0, j = 2r + 1 \). Furthermore, \( i, j \in \mathcal{H}^{s(0)} \).

Applying Corollary 49 we see that \( P_{\mu_\ast}^{H} \) requires expected sample complexity of \( \Omega(\frac{d \log \frac{1}{\varepsilon}}{\Psi}) \). Here, we have \( \Psi = O(\mu_\ast \nu^2) = O(\mu_\ast \varepsilon^2) \). Also, we have \( d = 2t = \Omega(\min\{n, \sqrt{q}\}) \).

### 9.3 Bounds for \( P_{\text{ratio}} \) and \( P_{\text{coeff}} \) in terms of \( n, q \) in the log-concave setting

For this case, we adapt a construction of [15], with some slightly modified parameters and definitions. This construction will be based on Lemma 48 with \( d = 2 \). To simplify the notation, we write \( c, c^-, c^+ \) instead of \( c^{(0)}, c^{(1)}, c^{(2)} \). The vectors \( c^-, c^+ \) will be derived from \( c \) by setting

\[
c^-_k = c_k e^{-k\nu}, c^+_k = c_k e^{k\nu}
\]

for some parameter \( \nu > 0 \).

We define the values \( c_0, \ldots, c_n \) to be the coefficients of the polynomial \( g(x) = \prod_{k=0}^{n-1} (c_k + e^x) \); equivalently, we have \( Z(\beta | c) = \prod_{k=0}^{n-1} (c_k + e^x) \) for all values \( \beta \). Since this polynomial \( g(x) \) is real-rooted, the coefficients \( c_0, \ldots, c_n \) are log-concave [5].

There is another way to interpret the coefficients \( c_i \) which is useful for us. Consider independent random variables \( X_0, \ldots, X_{n-1} \), wherein \( X_i \) is Bernoulli-\( p_i \) for \( p_i = \frac{e^x}{e^x + e^x} \). Then \( \mu_\beta \) is the probability distribution on random variable \( X = X_0 + \cdots + X_{n-1} \). In particular, coefficient \( c_k \) is a scaled version of the probability \( \mu_0(k) \), in which turn is the probability that \( X = k \) at \( \beta = 0 \).

We will fix \( \beta_{\min} = 0 \). By a simple continuity argument, it is possible to select value \( \beta_{\max} \geq 0 \) to ensure that the problem instance \( c \) has any desired value of \( q > 0 \). Let us fix such \( \beta_{\max} \). We define

\[
z(\beta) = \log Z(\beta | c) = \sum_{k=0}^{n-1} \log(e^k + e^x).
\]

We recall a result of [15] calculating various parameters of the problem instances \( c, c^-, c^+ \).
Lemma 55 ([15]). Suppose that \( \nu \leq \nu_{\text{max}} \) for some constant \( \nu_{\text{max}} \). Define the parameters \( \kappa, \rho \) by

\[
\kappa = \sup_{\beta \in \mathbb{R}} z''(\beta), \quad \rho = |z'(\beta_{\text{max}}) - z'(\beta_{\text{min}})|
\]

Then the problem instances \( c^-, c^+, c \) have their corresponding values \( q^-, q^+ \) bounded by

\[
|q^\pm - q| \in [\rho \nu - \kappa \nu^2, \rho \nu + \kappa \nu^2]
\]

Furthermore, the triple of problem instances \( c, c^-, c^+ \) has \( \Psi \leq O(\kappa \nu^2) \).

We next estimate some parameters of these problem instances.

Proposition 56. For \( n < \sqrt{q} \), we have the following bounds:

\[
\beta_{\text{max}} \geq n, \quad z'(0) = \Theta(1), \quad z'(\beta_{\text{max}}) = \Theta(n), \quad \rho = \Theta(n), \quad \kappa \leq 4
\]

Proof. Let us first show the bound on \( \beta_{\text{max}} \). Because of the way we have chosen \( \beta_{\text{max}} \), it suffices to show that \( z(\beta) - z(0) \leq q \) for \( \beta = n \). We calculate this as follows:

\[
z(n) - z(0) = \sum_{k=0}^{n-1} \log(e^k + e^n) - \sum_{k=0}^{n-1} \log(e^k + 1) = \sum_{k=0}^{n-1} \log(e^k + e^n)
\]

Since \( k \leq n \), we have \( \frac{e^k + e^n}{e^k + 1} \leq e^n \), and hence this sum is at most \( n^2 \leq q \).

Next, we show the bounds on \( z'(\beta) \). Differentiating the function \( z \) gives \( z'(\beta) = \sum_{k=0}^{n-1} \frac{e^\beta}{e^k + e^n} \). So

\[
z'(0) = \sum_{k=0}^{n-1} \frac{1}{e^k + 1}, \quad \text{which is easily seen to be constant. Likewise, we have } z'(\beta_{\text{max}}) = \sum_{k=0}^{n-1} \frac{e^{\beta_{\text{max}}}}{e^k + e^{\beta_{\text{max}}}}.
\]

Since \( \beta_{\text{max}} \geq n \geq k \), each summand is \( \Theta(1) \), and the total sum is \( \Theta(n) \).

The bounds on \( z'(\beta_{\text{max}}) \) and \( z'(0) \) also show the bound for \( \rho \) (recalling that \( \beta_{\text{min}} = 0 \)).

Finally, we calculate \( \kappa \). Differentiating twice, we have \( z''(\beta) = \sum_{k=0}^{n-1} \frac{e^\beta}{(e^k + e^\beta)^2} \). Summing over \( k \leq \beta \) contributes at most \( \sum_{k=0}^{n-1} \frac{e^\beta}{e^k + e^\beta} \leq \sum_{k=0}^{n-1} e^{k \beta} \leq \frac{e^n}{e-1} \). Likewise, summing over \( k \geq \beta \) contributes at most \( \sum_{k=0}^{n-1} \frac{e^\beta}{e^k + e^\beta} \leq \sum_{k=0}^{n-1} e^{-k \beta} \leq \frac{e^n}{e-1} \).

We can now prove Theorem 50 part (a) and (b).

Proposition 57. Under the conditions of Theorem 50, any algorithm to solve \( P_{\text{ratio}} \) on log-concave problem instances with given values \( n, q \) must have expected sample complexity \( \Omega(\frac{\min(q, n^2) \log(1/\gamma)}{\varepsilon^2}) \).

Proof. Let us first show this for \( n < \sqrt{q} \). Let us set \( \nu = 3\varepsilon/\rho \). Then by Lemma 55, the values \( q, q^-, q^+ \) are separated by at least \( \rho \nu - \kappa \nu^2 = 3\varepsilon - 3\kappa \varepsilon^2/\rho^2 \). By Proposition 56 this is at least \( 3\varepsilon(1 - O(\varepsilon/n^2)) \).

For \( \varepsilon < \varepsilon_{\text{max}} \) and \( \varepsilon_{\text{max}} \) a sufficiently small constant, this is at least \( 2\varepsilon \). So the overall separation between \( q, q^- \) is at least \( 2\varepsilon \).

By Lemma 55 these problem instances have \( \Psi = O(\kappa \nu^2) = O(\kappa \varepsilon^2/\rho^2) \). By Propositions 56 this is \( O(\varepsilon^2/n^2) \). Therefore, by Corollary 49 the expected sample complexity of \( P_{\text{ratio}} \) on \( c \) is \( \Omega(n^2 \log(1/\gamma)/\varepsilon^2) \).

Next, suppose that \( n > \sqrt{q} \). Then we may construct the problem instance with \( n' = \min(2, \lfloor \sqrt{q} \rfloor) \); for \( q \geq q_{\text{min}} \) this satisfies \( n' \geq \Omega(\sqrt{q}) \). We add dummy zero coefficients, which does not change the value \( q \) for any of three problem instances \( c, c^+, c^- \). Solving \( P_{\text{ratio}} \) on this expanded problem instance with \( n \) variables thus is equivalent to solving \( P_{\text{ratio}} \) on the problem instance with \( n' \) variables, which requires sample complexity \( \Omega(n'^2 \log(1/\gamma)/\varepsilon^2) \).

Proposition 58. Suppose that \( n < \sqrt{q} \). Then, under the conditions of Theorem 50, any algorithm to solve \( P_{\text{ratio}, n, K}^{\text{coeff}} \) on log-concave problem instances with given parameters \( n, q \) and for \( K = \{0, n\} \) must have expected sample complexity \( \Omega(\frac{n^2 \log(1/\gamma)}{\varepsilon^2}) \).
Proof. We can calculate $\mu_0(0) = P[X_0 = \cdots = X_{n-1} = 0] = \prod_{k=0}^{n-1} e^{\frac{k}{\epsilon + \gamma}}$. Routine calculations show that this is $\Omega(1)$. Similarly, we have $\mu_{\beta_{\max}}(n) = P[X_0 = \cdots = X_{n-1} = 1] = \prod_{k=0}^{n-1} e^{\frac{-k}{\epsilon + \gamma_{\max}}}$. Since $k \leq n \leq \beta_{\max}$, this product is also $\Omega(1)$.

Now let us set $\nu = 3\epsilon/n$ to construct the problem instances $c^+, c^-$. We will now apply Corollary 19 for either of the problem instances $c^-, c^+$, let us set $i = 0, j = n$. We have shown that $i, j \in \mathcal{K}^*$ with respect to problem instance $c$, for some sufficiently small constant $\mu_*$.

Observe that $|\log(c_i/c_j) - \log(c_i^+/c_j^+)| = |\log(c_i/c_j) - \log(c_i^-/c_j^-)| = n\nu = 3\epsilon$. Therefore, the hypotheses of Corollary 19 are satisfied so the expected sample complexity of $P_k^{\mu_*, \mathcal{K}}$ is $\Omega\left(\frac{\log(1/\nu)}{\nu}\right)$. By Lemma 55, we have $\Psi = O(n\nu)$; by Proposition 56 and with our definition of $\nu$, this is $O(\epsilon^2/n^2)$. □

Corollary 59. Under the conditions of Theorem 57, any algorithm to solve $P_k^{\mu_*, \mathcal{H}}$ on log-concave problem instances with given parameters $n, q$ must have expected sample complexity $\Omega\left(\frac{\min\{q, n^2\} \log \frac{1}{\epsilon}}{\epsilon^2}\right)$.

Proof. If $n < \sqrt{q}$, this follows immediately from Proposition 58. Otherwise, we consider the problem instance for Proposition 58 corresponding to the alternate value $n' = \min\{2, \lfloor \sqrt{q} \rfloor \}$, and we add $n - n'$ dummy zero coefficients. Solving $P_{\mathcal{H}, \mathcal{K}}^{\mu_*, \mathcal{K}}$ on the full instance allows us to solve $P_{\mathcal{H}, \mathcal{K}}^{\mu_*, \mathcal{K}}$ for $\mathcal{K} = \{0, n'\}$, contradicting Proposition 58. □

A Proof of Theorem 8 (correctness with approximate oracles)

One can construct a coupling between $\mu_{\beta}$ and $\tilde{\mu}_{\beta}$ such that samples $k \sim \mu_{\beta}$ and $k \sim \tilde{\mu}_{\beta}$ are identical with probability at least $1 - ||\mu_{\beta} - \tilde{\mu}_{\beta}||_V \geq 1 - \frac{\gamma}{T}$. Assume that the $k^{th}$ call to $\mu_{\beta}$ in $\mathcal{A}$ is coupled with the $k^{th}$ call to $\tilde{\mu}_{\beta}$ in $\tilde{\mathcal{A}}$ when $\beta = \tilde{\beta}$. We say that the $k^{th}$ call is good if the produced samples are identical. Note, $P[k^{th} \text { call is good} | \text{all previous calls were good}] \geq 1 - \frac{\gamma}{T}$, since the conditioning event implies $\beta = \tilde{\beta}$. Also, if all calls are good then $\mathcal{A}$ and $\tilde{\mathcal{A}}$ give identical results.

Let $Z$ be the number of the calls to the sampling oracle by algorithm $\mathcal{A}$; by assumption, we have $E[Z] = T$. The union bound gives $\mathbb{P}[\text{all calls are good} | Z = k] \geq 1 - \frac{\gamma}{T} k$, and therefore

$$\mathbb{P}[\text{all calls are good}] = \sum_{k=0}^{\infty} \mathbb{P}[Z = k] \cdot \mathbb{P}[\text{all calls are good} | Z = k] \geq \sum_{k=0}^{\infty} \mathbb{P}[Z = k] \cdot \left(1 - \frac{\gamma}{T} k\right) = 1 - \frac{\gamma}{T} \cdot E[Z] = 1 - \gamma$$

We now have $\mathbb{P}[\text{output of } \tilde{\mathcal{A}} \text{ satisfies } C] \geq \mathbb{P}[\text{output of } \mathcal{A} \text{ satisfies } C \land \text{all calls are good}] \geq 1 - 2\gamma$ where the last inequality is by the union bound (recall that $\mathbb{P}[\text{output of } \mathcal{A} \text{ satisfies } C] \geq 1 - \gamma$).

B Proof of Lemma 11 (properties of the binomial distribution)

We first consider the case where $p \geq e^{-\epsilon}p_0$. For this, we use two variants of the Chernoff bound for binomials:

$$\mathbb{P}[\hat{p} \geq p + x] \leq e^{\frac{-Nx^2}{2(p+x)}}, \quad \mathbb{P}[\hat{p} \leq p - x] \leq e^{\frac{-Nx^2}{2p}}$$

Setting $x = (e^{-\epsilon} - 1)p$ and $x = (1 - e^{-\epsilon})$ respectively, these give us the bounds

$$\mathbb{P}[\hat{p} \geq e^{\epsilon}p] \leq e^{\frac{N(e^{\epsilon} - 1)^2p^2}{2e^{\epsilon}p}} \leq e^{\frac{N(e^{\epsilon} - 1)^2e^{-\epsilon}p_0}{2e^\epsilon}} = \exp\left(-N \times \frac{(1 - e^{-\epsilon})^2p_0}{2}\right)$$

$$\mathbb{P}[\hat{p} \leq e^{-\epsilon}p] \leq e^{\frac{N(1 - e^{-\epsilon})^2p^2}{2p}} \leq e^{\frac{N(1 - e^{-\epsilon})^2e^{-\epsilon}p_0}{2}}$$
These terms are both below $\gamma/2$ as long as $N \geq \frac{2e^x \log(2/\gamma)}{(1-e^{-x})^2}$. The union bound now gives the claim.

Next, consider the case where $p < e^{-x}p_0$. For fixed values $p_0$ and $N \geq \frac{2e^x \log(2/\gamma)}{(1-e^{-x})^2}$, let us define the function $f : [0, 1] \to [0, 1]$ by $f(z) = P[\tilde{p} \geq p_0 \mid \hat{p} \sim \frac{1}{N}\text{Binom}(N, z)]$. Clearly, $f(z)$ is a non-decreasing function of $z$. Also, $f(e^{-x}p_0) \leq \gamma/2$, as shown previously. Thus $f(p) \leq \gamma$ for $p < e^{-x}p_0$.

**C Proof of Lemma 10**

Let $a_1, \ldots, a_m$ be a vector satisfying the preconditions of the lemma. Let $k \in \{1, \ldots, m\}$ be chosen to maximize the value $ka_k$ (breaking ties arbitrarily). Clearly $a_k \leq 1/k$. If $a_k = 0$, then due to maximality of $k$ we have $a_1 = \cdots = a_m = 0$ and the result obviously holds. Otherwise, due to maximality of $k$, for $k > 1$ we have $(k-1)a_k \leq ka_k$, i.e. $\frac{a_{k-1}}{a_k} \leq \frac{k}{k-1}$. Similarly, if $k < m$ we have $\frac{a_{k+1}}{a_k} \leq \frac{k}{k+1}$.

Let us define the sequence $y_1, \ldots, y_m$ by:

$$y_i = \begin{cases} \frac{1}{k} \frac{k-1}{k} y_i^{k-1} & \text{if } i < k \\ \frac{1}{k} \frac{k-1}{k} y_i^{k-1} & \text{if } i \geq k \end{cases}$$

Note that $\frac{y_{k-1}}{y_k} = \frac{k}{k-1} \geq \frac{a_{k-1}}{a_k}$ and $\frac{y_{k+1}}{y_k} = \frac{k}{k+1} \geq \frac{a_{k+1}}{a_k}$ (assuming that $k > 1$ and $k < m$, respectively). Also, $y_k = \frac{1}{k} \geq a_k$. Since $\limsup_{i \to \infty} y_i$ is linear on $i \in \{1, \ldots, k\}$ and on $i \in \{k, \ldots, m\}$, log-concavity of sequences $a$ and $y$ shows that $a_i \leq y_i$ for $i = 1, \ldots, m$. We can thus write

$$\sum_{i=1}^{m} a_i \leq \sum_{i=1}^{\infty} y_i = \sum_{i=1}^{k-1} \frac{1}{k} \left( \frac{k-1}{k} \right)^{i-1} + \sum_{i=k}^{\infty} \frac{1}{k} \left( \frac{k}{k+1} \right)^{i-k}$$

$$= \frac{1}{k} \left[ \frac{(k-1)^{1-k} - 1}{1 - \frac{k-1}{k}} + \frac{1}{1 - \frac{k}{k+1}} \right] = \left( 1 - \frac{1}{k} \right)^{1-k} + \frac{1}{k}$$

Let us now define the function $g(x) = (1 - x)^{1-1/x} + x$. We have shown that $\sum a_i \leq g(1/k)$, and note that $1/k \in (0, 1/2]$. To finish the proof, we will show that $g(x) < e$ for all $x \in (0, 1/2]$.

Since $\lim_{x \to 0} g(x) = e$ and $\lim_{x \to 0} g'(x) = 1 - e/2 < 0$, it suffices to show that $g''(x) < 0$ in the interval $(0, 1/2)$. We compute:

$$g''(x) = \frac{(1-x) \log^2(1-x) - x^2}{x^4(1-x)^{1/x}}$$

Routine calculus shows that $(1-x) \log^2(1-x) < x^2$ in the range $(0, 1/2)$, which implies $g''(x) < 0$ as desired, and hence that $\sum a_i \leq g(y) \leq e$.

**D Alternate algorithms for $P_{\text{ratio}}^\mathcal{B}$**

As mentioned in the introduction, an alternative algorithm for problem $P_{\text{ratio}}$ with expected sample complexity $O\left( \frac{q \log n \log \log n}{e^2} \right)$ was proposed in [11, 15]. We begin by reviewing this algorithm and showing how to extend it to solve the batched $P_{\text{ratio}}^\mathcal{B}$ problem.

Let us define a cooling schedule to be a sequence $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_\ell)$ with $\beta_{\min} = \alpha_0 < \ldots < \alpha_\ell = \beta_{\max}$. We define the length of $\alpha$ as $|\alpha| = \ell + 1$. We also define

$$\kappa(\alpha) = \sum_{i=1}^{\ell} (z(\alpha_{i-1}) - 2z(\frac{\alpha_{i-1} + \alpha_i}{2}) + z(\alpha_i))$$

where $z(\beta) = \log Z(\beta)$. Note that $z(\cdot)$ is an increasing convex function, and therefore $\kappa(\alpha) \geq 0$. The algorithms in [11, 15] first compute a cooling schedule $\alpha$ with small values of $|\alpha|$ and $\kappa(\alpha)$. The following result of [15] summarizes this:
Theorem 60 ([15] Theorem 8]). Fix a constant $\gamma_1 \in (0, 1)$. There is a randomized algorithm that for given values $\kappa_0 > 0$ and $\lambda \in (0, 1)$ produces a cooling schedule $\alpha$ such that $\mathbb{P}[\kappa(\alpha) \leq \kappa_0] \geq 1 - \gamma_1$ and $\mathbb{E}[|\alpha|] = O\left(\frac{(1+\log n)}{\kappa_0 + \log(1-\lambda)} + 1\right)$. Its expected sample complexity is $O\left(\frac{(1+\log n)}{\kappa_0 + \log(1-\lambda)} + 1\right)$.

Given such a cooling schedule $\alpha$, we can estimate $Q$ using the paired product estimator of Huber [11]:

Algorithm 10: Paired product estimator. Input: cooling schedule $\alpha = (\alpha_0, \ldots, \alpha_\ell)$, integer $r$.

1. foreach $j \in \{1, \ldots, r\}$ do
2.   foreach $i \in \{0, 1, \ldots, \ell\}$ sample $k_i \sim \mu_{\alpha_i}$
3.   set $W(j) = \exp\left(\sum_{i=1}^\ell \frac{\alpha_i - \alpha_{i-1}}{2} \cdot k_{i-1}\right)$ and $V(j) = \exp\left(-\sum_{i=1}^\ell \frac{\alpha_i - \alpha_{i-1}}{2} \cdot k_i\right)$
4. compute $\hat{W} = \frac{1}{\ell} \sum_{j=1}^r W(j)$ and $\hat{V} = \frac{1}{\ell} \sum_{j=1}^r V(j)$, output $\hat{Q} = \hat{W}/\hat{V}$

Theorem 61 ([11] [15]). Let $r$ be a positive integer, $\gamma_2 \in (0, 1)$, and let $\alpha$ be a cooling schedule with $\kappa(\alpha) \leq \log(1 + \frac{3}{4} \gamma_2 r (1 - e^{-(\gamma_2/2)^2}))$. Then the output of Alg. 10 satisfies $\mathbb{P}\left[\hat{Q}/Q \in [e^{-\varepsilon}, e^\varepsilon]\right] \geq 1 - \gamma_2$.

These two results give the following algorithm to estimate $Q$: fix positive constants $\kappa_0, \lambda, \gamma_1, \gamma_2$ with $(1 - \gamma_1)(1 - \gamma_2) = \frac{3}{4}$, run the algorithm from Theorem 60 and then Algorithm 10 with the obtained schedule $\alpha$ and $r = \left\lceil \frac{2(e^{\kappa_0} - 1)}{\gamma_2 (1-e^{-(\gamma/2)^2})} \right\rceil$. The resulting output $\hat{Q}$ satisfies $\mathbb{P}\left[\hat{Q}/Q \in [e^{-\varepsilon}, e^\varepsilon]\right] \geq \frac{3}{4}$, and the expected sample complexity is $O\left(\frac{q \log n}{\varepsilon^2}\right)$. The success probability can be boosted to $1 - \gamma$ by repeating the algorithm $\Theta(\log \frac{1}{\gamma})$ times and taking the median of estimates. This gives the following result:

Theorem 62 ([15]). There is an algorithm for $P_{\text{ratio}}$ with expected sample complexity $O\left(\frac{q \log n \log \frac{r}{\varepsilon^2}}{\varepsilon^2}\right)$.

Used directly, this estimates $Z(\beta)$ for a single value $\beta = \beta_{\text{max}}$. With small modifications, it can be used in a batch mode. We use Algorithm 11 below, where $\kappa_0, \lambda, \gamma_1, \gamma_2$ are fixed constants with $(1 - \gamma_1)(1 - \gamma_2) = \frac{3}{4}$ and $r = \left\lceil \frac{2(e^{\kappa_0} - 1)}{\gamma_2 (1-e^{-(\gamma/2)^2})} \right\rceil$.

Algorithm 11: Estimating $Q_{\beta}$. Input: subset $B \subseteq [\beta_{\text{min}}, \beta_{\text{max}}]$, parameter $\varepsilon > 0$.

1. call the algorithm of Theorem 60 to get cooling schedule $\alpha = (\alpha_0, \ldots, \alpha_\ell)$
2. foreach $j \in \{1, \ldots, r\}$ do
3.   foreach $i \in \{0, 1, \ldots, \ell\}$ do
4.      sample $k_i \sim \mu_{\alpha_i}$
5.      foreach $\beta \in B \cap (\alpha_i, \alpha_{i+1})$ do
6.         sample $k_\beta \sim \mu_\beta$
7.         compute $W_\beta(j) = \exp\left(\frac{\beta - \alpha_i}{2} \cdot k_i + \sum_{i=1}^\ell \frac{\alpha_i - \alpha_{i-1}}{2} \cdot k_{i-1}\right)$ and $V_\beta(j) = \exp\left(-\frac{\beta - \alpha_i}{2} \cdot k_\beta - \sum_{i=1}^\ell \frac{\alpha_i - \alpha_{i-1}}{2} \cdot k_i\right)$
8.     foreach $\beta \in B$ compute $\hat{W}_\beta = \frac{1}{r} \sum_{j=1}^r W_\beta(j)$ and $\hat{V}_\beta = \frac{1}{r} \sum_{j=1}^r V_\beta(j)$, output $\hat{Q}_\beta = \hat{W}_\beta/\hat{V}_\beta$

Algorithm 11 has expected sample complexity $O\left(\frac{|B| + q \log n}{\varepsilon^2}\right)$. For each fixed $\beta \in B$, Algorithm 11 can be viewed as a special case of Algorithm 10 with the cooling schedule $\alpha(\beta) = (\alpha_0, \ldots, \alpha_i, \beta)$, where $\beta \in (\alpha_i, \alpha_{i+1})$. To analyze this, we use the following observation:

Lemma 63. For $\beta \in (\beta_{\text{min}}, \beta_{\text{max}})$, we have $\kappa(\alpha(\beta)) \leq \kappa(\alpha)$.

Proof. Let $\beta \in (\alpha_i, \alpha_{i+1})$. Denote $f(x) = z(\alpha_i) - 2z\left(\frac{\alpha_i + x}{2}\right) + z(x)$. Since $z(\cdot)$ is convex, the function $z'(x)$ is non-decreasing on $[\alpha_i, \alpha_{i+1}]$. Therefore $f'(x) = z'(\frac{\alpha_i + x}{2}) \geq 0$. In particular, $f(\beta) \leq f(\alpha_{i+1})$. So $\kappa(\alpha_0, \ldots, \alpha_i, \beta) \leq \kappa(\alpha_0, \ldots, \alpha_i, \alpha_{i+1}) \leq \kappa(\alpha)$. \hfill \Box

In light of Theorem 61 and Lemma 63, we see that $\mathbb{P}\left[\hat{Q}_\beta/Q_\beta \in [e^{-\varepsilon}, e^\varepsilon]\right] \geq \frac{3}{4}$ for any fixed $\beta \in B$. Using a simple median-amplification technique, we can immediately get the following:
Theorem 64. There is an algorithm which takes as input a set \( \mathcal{B} \subseteq [\beta_{\min}, \beta_{\max}] \), and returns estimates \( \hat{Q}_\beta \) for \( \beta \in \mathcal{B} \) such that, with probability at least \( \gamma \), we have \( \frac{\hat{Q}_\beta}{Q_\beta} \in [e^{-\varepsilon}, e^{\varepsilon}] \) for all \( \beta \in \mathcal{B} \). The expected sample complexity is \( O\left( \frac{|\mathcal{B}| + q \log n}{\varepsilon^2 \log \frac{|\mathcal{B}|}{\varepsilon^2}} \right) \).

Acknowledgments

We thank Heng Guo for helpful explanations of algorithms for sampling connected subgraphs and matchings, and Maksym Serbyn for bringing to our attention the Wang-Landau algorithm and its use in physics.

The author Vladimir Kolmogorov is supported by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 616160.

References

[1] K. Adiprasito, J. Huh, and E. Katz. Hodge theory for combinatorial geometries. *Annals of Mathematics*, 188(2):381–452, 2018.
[2] N. Anari, K. Liu, S. O. Gharan, and C. Vinzant. Log-concave polynomials II: High-dimensional walks and an FPRAS for counting bases of a matroid. In *Proc. 51st annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1–12, 2019.
[3] R. E. Belardinelli and V. D. Pereyra. Wang-Landau algorithm: A theoretical analysis of the saturation of the error. *The Journal of Chemical Physics*, 127(18):184105, 2007.
[4] I. Bezáková, D. Štefankovič, V. V. Vazirani, and E. Vigoda. Accelerating simulated annealing for the permanent and combinatorial counting problems. *SIAM J. Comput.*, 37:1429–1454, 2008.
[5] P. Brändén. Unimodality, log-concavity, realrootedness and beyond. In *Handbook of Enumerative Combinatorics*, chapter 7, pages 438–483. CRC Press, 2015.
[6] M. Dyer and A. Frieze. Computing the volume of convex bodies: A case where randomness provably helps. In *Proc. AMS Symposium on Probabilistic Combinatorics and Its Applications 44*, pages 123–170, 1991.
[7] G. Fort, B. Jourdain, E. Kuhn, T. Leliévre, and G. Stoltz. Convergence of the Wang-Landau algorithm. *Mathematics of Computation*, 84(295):2297–2327, 2015.
[8] H. Guo and K. He. Tight bounds for popping algorithms. *ArXiv e-prints*, 2018.
[9] H. Guo and M. Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM J. Comput.*, 48(3):964–978, 2019.
[10] O. J. Heilmann and E. H. Lieb. Theory of monomer-dimer systems. In *Statistical Mechanics*, pages 45–87. Springer, 1972.
[11] M. Huber. Approximation algorithms for the normalizing constant of Gibbs distributions. *The Annals of Applied Probability*, 25(2):974–985, 2015.
[12] M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989.
[13] M. Jerrum and A. Sinclair. The Markov Chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1996.
[14] R. M. Karp and R. Kleinberg. Noisy binary search and its applications. In *Proc. 18th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 881–890, 2007.
[15] V. Kolmogorov. A faster approximation algorithm for the Gibbs partition function. *Proceedings of Machine Learning Research*, 75:228–249, 2018.
[16] Y. Komura and Y. Okabe. Difference of energy density of states in the Wang-Landau algorithm. *Phys. Rev. E*, 85, 010102(R), 2012.
[17] L. N. Shchur. On properties of the Wang-Landau algorithm. *Journal of Physics: Conference Series*, 1252, 2019.
[18] D. Štefankovič, S. Vempala, and E. Vigoda. Adaptive simulated annealing: A near-optimal connection between sampling and counting. *J. of the ACM*, 56(3) Article #18, 2009.
[19] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.
[20] F. Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.*, 86(10):2050–2053, 2001.