On the Complexity of Constrained Determinantal Point Processes

L. Elisa Celis[1], Amit Deshpande[2], Tarun Kathuria[3], Damian Straszak[4], and Nisheeth K. Vishnoi[5]

1 **École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland**
2 **Microsoft Research, Bangalore, India**
3 **Microsoft Research, Bangalore, India**
4 **École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland**
5 **École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland**

Abstract

Determinantal Point Processes (DPPs) are probabilistic models that arise in quantum physics and random matrix theory and have recently found numerous applications in theoretical computer science and machine learning. DPPs define probability distributions over subsets of a given ground set, they exhibit interesting properties such as negative correlation, and, unlike other models of negative correlation such as Markov random fields, have efficient algorithms for sampling. When applied to kernel methods in machine learning, DPPs favor subsets of the given data with more diverse features. However, many real-world applications require efficient algorithms to sample from DPPs with additional constraints on the sampled subset, e.g., partition or matroid constraints that are important from the viewpoint of ensuring priors, resource or fairness constraints on the sampled subset. Whether one can efficiently sample from DPPs in such constrained settings is an important problem that was first raised in a survey of DPPs for machine learning by Kulesza and Taskar and studied in some recent works.

The main contribution of this paper is the first resolution of the complexity of sampling from DPPs with constraints. On the one hand, we give exact efficient algorithms for sampling from constrained DPPs when the description of the constraints is in unary; this includes special cases of practical importance such as a small number of partition, knapsack or budget constraints. On the other hand, we prove that when the constraints are specified in binary, this problem is **#P**-hard via a reduction from the problem of computing mixed discriminants; implying that it may be unlikely that there is an FPRAS. Technically, our algorithmic result benefits from viewing the constrained sampling problem via the lens of polynomials and we obtain our complexity results by providing an equivalence between computing mixed discriminants and sampling from partition constrained DPPs. As a consequence, we obtain a few corollaries of independent interest: 1) An algorithm to count, sample (and, hence, optimize) over the base polytope of regular matroids when there are additional (succinct) budget constraints and, 2) An algorithm to evaluate and compute mixed characteristic polynomials, that played a central role in the resolution of the Kadison-Singer problem, for certain special cases.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems, G.2.1 Counting Problems

Keywords and phrases determinantal point processes, constraints, matroids, sampling and counting, polynomials, mixed discriminant

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2017.36

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017).
Editors: Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala; Article No. 36; pp. 36:1–36:22

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Algorithms for sampling from a discrete set of objects are sought after in various disciplines of computer science, optimization, mathematics and physics due to their far reaching applications. For instance, sampling from the Gibbs distribution was one of the original optimization methods (see, e.g., [17]) and sampling from dependent distributions is often used in the design of approximation algorithms (see, e.g., [5, 8, 19]). In machine learning, algorithms for sampling from discrete probability distributions are desired in various summarization, inference and learning tasks [33, 28, 24]. A particular class of probability distributions that has received much attention are the Determinantal Point Processes (DPP). In the discrete setting, a DPP is a distribution over subsets of a finite data set $[m] \stackrel{\text{def}}{=} \{1, 2, \ldots, m\}$. Here, a data point i is associated to a feature vector $v_i \in \mathbb{R}^d$, and an $m \times m$ positive semidefinite (PSD) kernel L gives the dot product of the feature vectors of any two data points as a measure of their pairwise similarity. Determinants, then, provide a natural measure of the diversity of a subset of data points, often backed by a physical intuition based on volume or entropy. A DPP is thus defined with respect to the kernel L such that for all $S \subseteq [m]$ we have $\mathbb{P}(S) \propto \det(L_{S,S})$, where $L_{S,S}$ is the principal minor of L corresponding to rows and columns from S.[1] The quantity $\det(L_{S,S})$ can be interpreted as the squared volume of the $|S|$-dimensional parallelepiped spanned by the vectors $\{v_i : i \in S\}$ and, intuitively, the larger the volume, the more *diverse* the set of vectors. Hence such distributions tend to prefer most diverse or *informative* subsets of data points. Mathematically, the fact that the probabilities are derived from determinants allows one to deduce elegant and non-trivial properties of such distributions, such as negative correlation and concentration of measure. Efficient polynomial time algorithms for sampling from DPPs (see [20, 10]) is what sets them apart from the other probabilistic models of negative correlation such as Markov random fields. As a consequence, sampling from DPPs has been successfully applied to a number of problems, such as document summarization, sensor placement and recommendation systems [26, 23, 37, 36, 35].

Given the wide applicability of DPPs, a natural question is whether they can be generalized to incorporate priors, budget or fairness constraints, or other natural combinatorial constraints. In other words, given an $m \times m$ kernel L and a family $\mathcal{C} \subseteq 2^{[m]}$ that represents constraints on the subsets, can we efficiently sample from the DPP distribution supported only on $\mathcal{C}$; that is, $\mathbb{P}(S) \propto \det(L_{S,S})$ for $S \in \mathcal{C}$, and $\mathbb{P}(S) = 0$ otherwise. Here are two important special cases.

- *Fairness (or Partition) constraints:* Consider the setting where $[m]$ is a collection of data points and each point is associated with a sensitive attribute such as gender. Then $\mathcal{C}$ is the family of attribute-unbiased subsets of $[m]$ – e.g., those subsets that contain an equal number of male and female points. Thus, the corresponding $\mathcal{C}$-constrained DPP outputs a diverse set of points while maintaining fairness with respect to the sensitive attribute; see [7] for this and other applications of constrained DPPs to eliminating algorithmic bias.
- *Budget constraints:* In data subset selection or active learning, when there is a cost $c_i \in \mathbb{Z}$ associated with each data point, it is natural to ask for a diverse training sample S from a corresponding DPP such that its cost $\sum_{i \in S} c_i$ is bounded from above by $C \in \mathbb{Z}$. See also [34] for a related optimization variant.

[1] We treat DPPs via *L-ensembles*, while commonly they are defined using *kernel matrices*, for practical purposes these two definitions are equivalent.

In their survey, [24] posed the open question of efficiently sampling from DPPs with additional combinatorial constraints on the support of the distribution. Sampling from constrained DPPs is algorithmically non-trivial, as many natural heuristics fail. The probability mass on the constrained family of subsets can be arbitrarily small, hence, ruling out a rejection sampling approach. For partition constraints, a natural heuristic is to sample from independent smaller-sized DPPs, each defined over a different part. However, such a product distribution would select two (potentially very similar) items from two different parts independently, whereas in a constrained DPP distribution they must be negatively correlated. Unlike DPPs and the special case of cardinality-constrained k-DPPs (in which $\mathcal{C}$ is the family of *all* subsets of size k – see Section 1.2), it is not clear that there is a clean expression for the partition function or the marginals of a constrained DPP. Another approach to approximately sample from constrained DPPs is via Markov Chain Monte Carlo (MCMC) methods as in the recent work of [25]. This approach can be shown to be efficient when the underlying Markov chain is connected and the DPP kernel is close to a diagonal matrix (or nearly-log-linear; see Theorem 4 of [25]). However, the above conditions do not hold for sampling partition-constrained subsets – even with constant number of parts – from most DPP kernels. Thus, while the problem of sampling from constrained DPPs has attracted attention, its complexity has remained open.

The main contribution of our paper is the first resolution of the problem of sampling from constrained DPPs. Our results give a dichotomy for the complexity of this problem: On the one hand, we give exact algorithms which are polynomial time when the description of $\mathcal{C}$ (in terms of the costs and budgets) is in *unary*; this includes special cases of practical importance such as the fairness, partition or budget constraints mentioned above. On the other hand, we prove that in general this problem is #**P**-hard when the constraints of $\mathcal{C}$ are specified in binary. Our algorithmic results go beyond the MCMC methods and include special cases of practical importance such as (constantly-many) partition or fairness constraints (studied, e.g., by [7]) and a more general class of budget constraints and linear families defined in the following section.

Our algorithmic results benefit from viewing the probabilities arising in constrained DPPs as coefficients of certain multivariate polynomials. This viewpoint also allows us to extend our result on constrained DPPs to derive important consequences of independent interest. For instance, using the intimate connection between linear matroids and DPPs, we arrive at efficient algorithms to sample a basis of regular matroids when there are additional budget constraints – significantly extending results of [12, 6] for spanning trees. To prove the hardness result, we present an *equivalence* between the problem of *computing* the mixed discriminant of a tuple of PSD matrices and that of *sampling* from partition-constrained DPPs. Mixed discriminants (see Section 4.1 for a definition) generalize the permanent, arise in the proof of the Kadison-Singer problem ([27], see [18] for a survey on this topic) and are closely related to mixed volumes (see, e.g., [4]). However, unlike the result for permanent [21] and volume computation [11], there is evidence that the mixed discriminant problem may be much harder and may not admit an FPRAS; see [15]. Thus, in light of our equivalence between mixed discriminants and partition DPPs, it may be unlikely that we can even approximately sample from partition DPPs (with an arbitrary number of parts) efficiently. Further, this connection implies that important special cases of the mixed discriminant problem, for instance computing the higher order coefficients of the mixed-characteristic polynomial or evaluating the mixed characteristic polynomial of low rank matrices at a given point, can be solved efficiently, which may be of independent interest.

1.1 Our Framework and Results

The starting point of our work is the observation that if we let μ be the measure on subsets of $[m]$ corresponding to the kernel matrix L (i.e., $\mu(S) \stackrel{\text{def}}{=} \det(L_{S,S})$), then given L, there is an efficient algorithm to evaluate the polynomial

$$g_\mu(x) \stackrel{\text{def}}{=} \sum_{S \subseteq [m]} \mu(S) x^S$$

where x^S denotes $\prod_{i \in S} x_i$ for any setting of its variables. Indeed, consider the Cholesky decomposition of the kernel $L = VV^\top$. Then, the polynomial $x \mapsto \det(V^\top X V + I)$ (where X denotes the diagonal matrix with x on the diagonal) is equal to $g_\mu(x)$ (see Fact 8) and hence can be efficiently evaluated using Gaussian elimination for any input x. We say that such a μ has an *efficient evaluation oracle* and, as it turns out, this is the *only* property we need from DPPs and our results generalize to any measure μ for which we have such an evaluation oracle. Before we explain our results, we formally introduce the sampling problem in this general framework.

▶ **Definition 1** (Sampling). Let $\mu : 2^{[m]} \to \mathbb{R}_{\geq 0}$ be a function assigning non-negative real values to subsets of $[m]$ and let $\mathcal{C} \subseteq 2^{[m]}$ be any family of subsets of $[m]$. We denote the (sampling) problem of selecting a set $S \in \mathcal{C}$ with probability $p_S = \frac{\mu(S)}{\sum_{T \in \mathcal{C}} \mu(T)}$ by $\textsc{Sample}[\mu, \mathcal{C}]$.

Building up on the equivalence between sampling and counting [22], we show that if one is given oracle access to the generating polynomial g_μ and if μ is a nonnegative measure, the problem $\textsc{Sample}[\mu, \mathcal{C}]$ is essentially equivalent to the following counting problem; see Theorem 21 in Appendix B.

▶ **Definition 2** (Counting). Let $\mu : 2^{[m]} \to \mathbb{R}_{\geq 0}$ be a function assigning non-negative real values to subsets of $[m]$ and let $\mathcal{C} \subseteq 2^{[m]}$ be any family of subsets of $[m]$. We denote the (counting) problem of computing the sum $\sum_{S \in \mathcal{C}} \mu(S)$ by $\textsc{Count}[\mu, \mathcal{C}]$.

In particular, a polynomial time algorithm for $\textsc{Count}[\mu, \mathcal{C}]$ can be translated into a polynomial time algorithm for $\textsc{Sample}[\mu, \mathcal{C}]$. Interestingly, this relation holds no matter what $\mathcal{C}$ is; in particular, no specific assumptions on how the access to $\mathcal{C}$ is provided are required.

Towards developing counting algorithms in our framework, we focus on a class of families $\mathcal{C} \subseteq 2^{[m]}$, which we call *Budget Constrained Families*, where a cost vector $c \in \mathbb{Z}^m$ and a budget value $C \in \mathbb{Z}$ are given, and the family consists of all sets $S \subseteq [m]$ of total cost $c(S) \stackrel{\text{def}}{=} \sum_{i \in S} c_i$ at most C. We call the counting and sampling problems for this special case $\textsc{BCount}[\mu, c, C]$ and $\textsc{BSample}[\mu, c, C]$ respectively.

Our key result is that the BCount problem (and hence also BSample) is efficiently solvable whenever the costs are not too large in magnitude.

▶ **Theorem 3** (Counting under Budget Constraints). *There is an algorithm, which given a function $\mu : 2^{[m]} \to \mathbb{R}$ (via oracle access to g_μ), a cost vector $c \in \mathbb{Z}^m$ and a cost value $C \in \mathbb{Z}$ solves the $\textsc{BCount}[\mu, c, C]$ problem in polynomial time with respect to m and $\|c\|_1$.*

The proof of Theorem 3 (see Section 2) benefits from an interplay between probability measures and polynomials. It reduces the counting problem to computing the coefficients of a certain univariate polynomial which, in turn, can be evaluated efficiently given access to the generating polynomial for μ. We can then employ interpolation in order to recover the required coefficients.

It is not hard to see that Theorem 3 also implies the same result for families with a single *equality* constraint ($c(S) = C$) or for any constraint of the form $c(S) \in K$, where $K \subseteq \mathbb{Z}$ is given as input together with $c \in \mathbb{Z}^m$ and $C \in \mathbb{Z}$. Furthermore, our framework can be easily extended to the case of multiple (constant number of) such constraints.

As mentioned earlier, what makes DPPs attractive is that their generating polynomial, arising from a determinant, is efficiently computable. Using this fact, Theorem 3 and the equivalence between sampling and counting, we can deduce the following result.

▶ **Corollary 4.** *There is an algorithm, which given a PSD matrix $L \in \mathbb{R}^{m\times m}$, a cost vector $c \in \mathbb{Z}^m$ and a cost value $C \in \mathbb{Z}$ samples a set S of cost $c(S) \leq C$ with probability proportional to $\det(L_{S,S})$. The running time of the algorithm is polynomial with respect to m and $\|c\|_1$.*

From the above one can derive efficient sampling algorithms for several classes of constraint families $\mathcal{C}$ which have *succinct descriptions*. Indeed, we establish counting and sampling algorithms for a general class of *linear families* of the form

$$\mathcal{C} = \{S \subseteq [m] : c_1(S) \in K_1, c_2(S) \in K_2, \ldots, c_p(S) \in K_p\} \tag{1}$$

where $c_1, c_2, \ldots, c_p \in \mathbb{Z}^m$ and $K_1, \ldots, K_p \subseteq \mathbb{Z}$. We prove the following

▶ **Corollary 5.** *There is an algorithm, which given a PSD matrix $L \in \mathbb{R}^{m\times m}$ and a description of a linear family $\mathcal{C}$ as in (1), samples a set $S \in \mathcal{C}$ with probability proportional to $\det(L_{S,S})$. The running time of the algorithm is polynomial in m and $\prod_{j=1}^{p} \left(\|c_j\|_1 + 1\right)$.*

One particular class of families for which the above yields polynomial time sampling algorithms are partition families (families of bases of partition matroids) over constantly many parts (see Corollary 10). An important open problem that remains is to come up with even faster algorithms.

Another application of Theorem 3, which we present in Section 6, is to combinatorial sampling and counting problems. More precisely, we note that the indicator measure of bases of regular matroids has an efficiently computable generating polynomial; hence, we can solve their corresponding budgeted versions of counting and sampling problems.

One may ask if the dependence on $\|c\|_1$ in Theorem 3 can be improved. We prove that the answer to this question is no in a very strong sense. To state our hardness result, we introduce ECount – a natural variant of the BCount problem – in which the sum is over subsets of cost equal to a given value C instead of at most C (such a problem is no harder than BCount). We provide an approximation preserving reduction showing that ECount$[\mu, c, C]$ is at least as hard as computing *mixed discriminants* of tuples of positive semidefinite (PSD) matrices when c and C are given in binary, and can be exponentially large in magnitude. Recall that for a tuple of $m \times m$ PSD matrices $A_1, \ldots, A_m$, their mixed discriminant is the coefficient of the monomial $\prod_{i=1}^{m} x_i$ in the polynomial $\det(\sum_{i=1}^{m} x_i A_i)$.

▶ **Theorem 6** (Hardness of Counting under Budget Constraints)**.** BCount$[\mu, c, C]$ *is $\#\mathbf{P}-$hard. Moreover, when μ is a determinantal function,* ECount$[\mu, c, C]$ *is at least as hard to approximate as mixed discriminants of tuples of PSD matrices.*

To prove this result we show an *equivalence* between the counting problem corresponding to partition-constrained DPPs (with a large, super-constant number of parts) and computing mixed discriminants. Unlike permanents [21], no efficient approximation scheme is known for estimating mixed discriminants and there is some evidence [15] that there may be none. To further understand to what extent g_μ is the cause of computational hardness, in Appendix A (see Theorem 20) we provide another hardness result; it considers a μ that is a 0/1 indicator function for spanning trees in a graph (with efficiently computable g_μ). We prove that

ECount$[\mu, c, C]$ is at least as hard to approximate as the number of perfect matchings in general (non-bipartite) graphs, which is another problem for which existence of an FPRAS is open.

Finally, this connection between partition-DPPs and mixed discriminants, along with our results to efficiently solve the counting problem for partition-DPPs with constantly many parts, gives us other applications of independent interest. 1) The ability to compute the top few coefficients of the *mixed characteristic polynomial* that arises in the proof of the Kadison-Singer problem; see Theorem 15. 2) The ability to compute in polynomial time, the mixed characteristic polynomial *exactly*, when the linear matrix subspace spanned by the input matrices has constant dimension; see Theorem 17 and Corollary 18.

1.2 Other Related Work

For sampling from k-DPPs there are exact polynomial time algorithms (see [20, 10, 24]). There is also recent work on faster approximate MCMC algorithms for sampling from various unconstrained discrete point processes (see [31, 1] and the references therein), and algorithms that are efficient for constrained DPPs under certain restrictions on the kernel and constraints (see [25] and the references therein). To the best of our knowledge, our result is the first efficient sampling algorithm that works for all kernels and for any constraint set with small description complexity. Recently, approximate algorithms for the counting and sampling were presented in [32]. On the practical side, diverse subset selection and DPPs arise in a variety of contexts such as structured prediction [30], recommender systems [14] and active learning [34], where the study of DPPs with additional constraints is of importance.

2 Counting with Budget Constraints

Proof of Theorem 3. Let us first consider the case in which the cost vector c is nonnegative, i.e., $c \in \mathbb{N}^m$. We introduce a new variable z and consider the polynomial

$$h(z) \stackrel{\text{def}}{=} g_\mu(z^{c_1}, z^{c_2}, \ldots, z^{c_m}).$$

Since $g_\mu(x_1, \ldots, x_m) = \sum_{S \subseteq [m]} \mu(S) \prod_{i \in S} x_i$, we have

$$h(z) = \sum_{S \subseteq [m]} \mu(S) \prod_{i \in S} z^{c_i} = \sum_{S \subseteq [m]} \mu(S) z^{c(S)} = \sum_{0 \le d \le \|c\|_1} z^d \sum_{S:\ c(S)=d} \mu(S).$$

Hence, the coefficient of z^d in $h(z)$ is equal to the sum of $\mu(S)$ over all sets S such that $c(S) = d$. In particular, the output is the sum of coefficients over $d \le C$.

It remains to show how to compute the coefficients of h. Note that we do not have direct access to g_μ. However, we can evaluate $g_\mu(x)$ at any input $x \in \mathbb{R}^m$, which in turn allows us to compute $h(z)$ for any input $z \in \mathbb{R}$. Since $h(z)$ is a polynomial of degree at most $\|c\|_1$, in order to recover the coefficients of h, it suffices to evaluate it at $\|c\|_1 + 1$ inputs and perform interpolation. When using FFT, the total running time becomes:

$$(\|c\|_1 + 1) \cdot T_\mu + \widetilde{O}(\|c\|_1),$$

where T_μ is the running time of the evaluation oracle for g_μ.

In order to deal with the case in which c has negative entries, consider a modified version of h:

$$h(z) \stackrel{\text{def}}{=} z^{\|c\|_1} g_\mu(z^{c_1}, z^{c_2}, \ldots, z^{c_m}).$$

Clearly, $h(z)$ is a polynomial of degree at most $2 \cdot \|c\|_1$ whose coefficients encode the desired output. ◀

▶ Remark. Note that the bit complexity of the output of the proposed algorithm is polynomial in the input size since it is a result of solving a linear system with all the coefficients being polynomially bounded.

We also state a simple consequence of the above proof that is often convenient to work with.

▶ **Corollary 7.** *There is an algorithm that, given a vector $c \in \mathbb{Z}^m$, a value $C \in \mathbb{Z}$ and oracle access to g_μ computes the sum $\sum_{S:\ c(S)=C} \mu(S)$ in time polynomial with respect to m and $\|c\|_1$.*

In the above, note the equality $c(S) = C$ instead of $c(S) \leq C$ as in BCount.

3 Determinantal Point Processes

A Determinantal Point Process (DPP) is a probability distribution μ over subsets of $[m]$ defined with respect to a symmetric positive semidefinite matrix $L \in \mathbb{R}^{m\times m}$ by $\mu(S) \propto \det(L_{S,S})$; i.e.,

$$\mu(S) \stackrel{\text{def}}{=} \frac{\det(L_{S,S})}{\sum_{T\subseteq[m]} \det(L_{T,T})}.$$

We will often use a different matrix to represent the measure μ; let $V \in \mathbb{R}^{m\times n}$ be a matrix, such that $L = VV^\top$ (the Cholesky decomposition of L). Then, $\det(L_{S,S}) = \det(V_S V_S^\top)$.

An important open problem related to DPPs is the sampling problem under additional combinatorial constraints imposed on the ground set $[m]$. We prove that these problems are polynomial time solvable for succinct budget constraints, as in Theorem 3. We start by establishing the fact that generating polynomials for determinantal distributions are efficiently computable.

▶ **Fact 8.** *Let $L \in \mathbb{R}^{m\times m}$ be a PSD matrix with $L = VV^\top$ for some $V \in \mathbb{R}^{m\times n}$. If $\mu : 2^{[m]} \to \mathbb{R}_{\geq 0}$ is defined as $\mu(S) \stackrel{\text{def}}{=} \det(L_{S,S})$ then $\det(V^\top XV + I) = \sum_{S\subseteq[m]} x^S \mu(S)$, where X is the diagonal matrix of indeterminates $X = \mathsf{Diag}(x_1, \ldots, x_m)$ and I is the $n \times n$ identity matrix.*

Proof. We start by applying the Sylvester's determinant identity

$$\det(V^\top XV + I) = \det\left(\left(\sqrt{X}V\right)\left(\sqrt{X}V\right)^\top + I\right).$$

It is well known that for a symmetric matrix $A \in \mathbb{R}^{m\times n}$ the coefficient of t^k in the polynomial $\det(A + tI)$ is equal to $\sum_{|S|=n-k} \det(A_{S,S})$. Applying this result to $A = \left(\sqrt{X}V\right)\left(\sqrt{X}V\right)^\top$, we get

$$\det(A_{S,S}) = x^S \det(V_S V_S^\top) = x^S \det(L_{S,S}),$$

which concludes the proof by simply taking $t = 1$. ◀

Now we are ready to deduce Corollary 4.

Proof of Corollary 4. A polynomial time counting algorithm follows directly from Theorem 3 and Fact 8. To deduce sampling we apply the result on equivalence between sampling and counting Theorem 21. In fact when applied to an exact counting algorithm we obtain an exact sampling procedure. ◀

We move to the general result on sampling for linear families – Corollary 5. One can deduce it directly from Theorem 3, but this leads to a significantly suboptimal algorithm. Instead we take a different path and reprove Theorem 3 in a slightly higher generality.

Proof of Corollary 5. We will show how to solve the counting problem – sampling will then follow from Theorem 21. Also, for simplicity we assume that all the entries in the cost vectors are nonnegative, this can be extended to the general setting as in the proof of Theorem 3.

Let g be the generating polynomial of the determinantal function $\mu(S) = \det(L_{S,S})$, which is efficiently computable by Fact 8. For notational clarity we will use superscripts to index constraints. For every constraint "$c^{(j)}(S) \in K_i$" ($j = 1, 2, \ldots, p$) introduce a new formal variable y_j. For every index $i \in [m]$ define the monomial:

$$s_i = \prod_{j=1}^{p} y_j^{c_i^{(j)}}.$$

The above encodes the cost of element i with respect to all cost vectors $c^{(j)}$ for $j = 1, 2, \ldots, p$. Consider the polynomial $h(y_1, \ldots, y_p) = g(s_1, s_2, \ldots, s_m)$. It is not hard to see that the coefficient of a given monomial $\prod_{j=1}^{p} y_j^{d_j}$ in h is simply the sum of $\mu(S)$ over all sets S satisfying $c^{(1)}(S) = d_1, c^{(2)}(S) = d_2, \ldots, c^{(p)}(S) = d_p$. Hence the solution to our counting problem is simply the sum of certain coefficients of h. It remains to show how to recover all the coefficients efficiently.

Note that we can efficiently evaluate the polynomial h at every input $(y_1, \ldots, y_p) \in \mathbb{R}^p$. One can then apply interpolation to recover all coefficients of h. The running time is polynomial in the total number of monomials in h (this is the number of variables of a linear system which can be used to find the coefficients), which can be bounded from above by $\prod_{j=1}^{p} \left(\left\| c^{(j)} \right\|_1 + 1 \right)$. ◀

We derive now one interesting application of Corollary 5 – sampling from partition constrained DPPs. Let us first define *partition families* formally.

▶ **Definition 9.** Let $[m] = P_1 \cup P_2 \cup \cdots \cup P_p$ be a partition of $[m]$ into disjoint, nonempty sets and let $b_1, b_2, \ldots, b_p$ be integers such that $0 \leq b_i \leq |P_i|$. A family of sets of the form

$$\mathcal{C} = \{S \subseteq [m] : |S \cap P_j| = b_j, \text{ for every } j = 1, 2, \ldots, p\}$$

is called a partition family.

We prove the following consequence of Corollary 5, which asserts that polynomial time counting and sampling is possible for DPPs under partition constraints for constant p.

▶ **Corollary 10.** *Given a DPP defined by $L \in \mathbb{R}^{m \times m}$ and a partition family $\mathcal{C}$ with a constant number of parts, there exists a polynomial time sampling algorithm for the distribution*

$$\mu_{\mathcal{C}}(S) \stackrel{\text{def}}{=} \frac{\det(L_{S,S})}{\sum_{T \in \mathcal{C}} \det(L_{T,T})} \qquad \text{for } S \in \mathcal{C}.$$

Proof. In light of Corollary 5 it suffices to show that every partition family has a succinct representation as a linear family. We show that it is indeed the case. Consider a partition family $\mathcal{C}$ induced by the partition $P_1 \cup P_2 \cup \ldots \cup P_p = [m]$ and numbers $b_1, b_2, \ldots, b_p$. Define the following cost vectors: $c_j = 1_{P_j}$, for $j = 1, 2, \ldots, p$, i.e., the indicator vectors of the sets $P_1, P_2, \ldots, P_p$. Moreover define K_j to be $\{b_j\}$ for every $j = 1, 2, \ldots, p$. It is then easy to see that "$c_j(S) \in K_j$" is implementing the constraint $|P_j \cap S| = b_j$. In other words the family $\mathcal{C}$ is equal to the linear family defined by cost vectors $c_1, c_2, \ldots, c_p$ and sets $K_1, K_2, \ldots, K_p$. It remains to observe that $\|c_j\|_1 = |P_j| \leq m$ and hence $\prod_{j=1}^{p} (\|c_j\| + 1) = O(m^p)$. Since $p = O(1)$ the algorithm from Corollary 5 runs in polynomial time. ◀

4 Hardness Result

In this section we study hardness of BCount$[\mu, c, C]$. Theorem 3 implies that BCount is polynomial time solvable whenever we measure the complexity with respect to the unary encoding length of the cost vector c. Here we prove that if c is given in binary, the problem becomes $\#\mathbf{P}-$hard. Moreover, existence of an efficient approximation scheme for a closely related problem (instead of counting all objects of cost *at most* C, count objects of cost *exactly* C) would imply existence of such schemes for counting perfect matchings in non-bipartite graphs (see Appendix A) and for computing mixed discriminants. In both cases, these are notorious open questions and the latter is believed to be unlikely.

4.1 Mixed Discriminants

We relate the BCount problem to the well studied problem of computing mixed discriminants of PSD matrices and prove Theorem 6. Recall the definition:

▶ **Definition 11.** Let $A_1, A_2, \ldots, A_m \in \mathbb{R}^{d\times d}$ be symmetric matrices of dimension d. The mixed discriminant of a tuple $(A_1, A_2, \ldots, A_d)$ is defined as

$$D(A_1, A_2, \ldots, A_d) \stackrel{\text{def}}{=} \frac{\partial^d}{\partial z_1 \ldots \partial z_d} \det(z_1 A_1 + z_2 A_2 + \cdots + z_d A_d).$$

Computing mixed discriminants of PSD matrices is known to be $\#\mathbf{P}$-hard, since they can encode the permanent. However, as opposed to the permanent, there is no FPRAS known for computing mixed discriminants, and the best polynomial time approximation algorithms by [4, 16] have an exponentially large approximation ratio.

The main technical component in our proof of Theorem 6 is the following lemma.

▶ **Lemma 12.** *There is a polynomial time reduction, which given a tuple $(A_1, \ldots, A_n)$ of PSD $n \times n$ matrices outputs a PSD matrix $L \in \mathbb{R}^{m\times m}$, a cost vector $c \in \mathbb{Z}^m$ and a cost value $C \in \mathbb{Z}$ such that*

$$n! \cdot D(A_1, A_2, \ldots, A_n) = \sum_{S\subseteq[m],\ c(S)=C} \mu(S),$$

where $\mu(S) = \det(L_{S,S})$, for $S \subseteq [m]$. Moreover, $\|c\|_1 \le 2^{O(n \log n)}$.

Before proving Lemma 12 let us first state several important properties of mixed discriminants, which we will rely on; for proofs of these facts we refer the reader to [3].

▶ **Fact 13** (Properties of Mixed Discriminants). *Let $A, B, A_1, A_2, \ldots, A_n$ be symmetric $n \times n$ matrices.*

1. *D is symmetric, i.e.,*

$$D(A_1, A_2, \ldots, A_n) = D(A_{\sigma(1)}, A_{\sigma(2)}, \ldots, A_{\sigma(n)}), \text{ for any permutation } \sigma \in S_n.$$

2. *D is linear with respect to every coordinate, i.e.,*

$$D(\alpha A + \beta B, A_2, \ldots, A_n) = \alpha D(A, A_2, \ldots, A_n) + \beta D(B, A_2, \ldots, A_n).$$

3. *If $A = \sum_{i=1}^n v_i v_i^\top \in \mathbb{R}^{n\times n}$ then we have: $\det(A) = n!\ D(v_1 v_1^\top, \ldots, v_n v_n^\top)$.*

Proof of Lemma 12. Consider a tuple $(A_1, A_2, \ldots, A_n)$ of PSD matrices. The first step is to decompose them into rank-one summands:

$$A_i = \sum_{j=1}^{r} v_{i,j} v_{i,j}^\top,$$

where $v_{i,j} \in \mathbb{R}^n$ for $1 \leq i, j \leq n$ (some $v_{i,j}$'s can be zero if $\operatorname{rank}(A_i) < n$). This step can be performed using the Cholesky decomposition.

Let $M = \{(i,j) : 1 \leq i, j \leq n\}$ and for every $i = 1, 2, \ldots, n$ define $P_i = \{i\} \times [n]$. We take $m = |M| = n^2$ and define a family $\mathcal{C}$ of $n-$subsets of M to be

$$\mathcal{C} = \{S \subseteq [m] : |S \cap P_i| = 1 \text{ for every } i = 1, 2, \ldots, n\}.$$

Let V denote an $m \times n$ matrix with rows indexed by M, for which the eth row is v_e as above ($e \in M$, i.e., $e = (i,j)$ for some $i, j \in [n]$). We also set $L = VV^\top$, hence L is an $m \times m$ symmetric, PSD matrix. Finally, let $\mu(S) = \det(L_{S,S})$. Note that for sets S of cardinality n we have

$$\mu(S) = \det(L_{S,S}) = \det(V_S V_S^\top) = \det(V_S^\top V_S) = \det\left(\sum_{e \in S} v_e v_e^\top\right).$$

In the calculation below we rely on properties of mixed discriminants listed in Fact 13 and on the fact that $|S| = n$ for $S \in \mathcal{C}$.

$$\begin{aligned}
D(A_1, A_2, \ldots, A_n) &= D\left(\sum_{j=1}^{n} v_{1,j} v_{1,j}^\top, \sum_{j=1}^{n} v_{2,j} v_{2,j}^\top, \ldots, \sum_{j=1}^{n} v_{n,j} v_{n,j}^\top\right) \\
&= \sum_{1 \leq j_1, j_2, \ldots, j_n \leq n} D(v_{1,j_1} v_{1,j_1}^\top, v_{2,j_2} v_{2,j_2}^\top, \ldots, v_{n,j_n} v_{n,j_n}^\top) \\
&= \sum_{e_1 \in P_1, e_2 \in P_2, \ldots, e_n \in P_n} D(v_{e_1} v_{e_1}^\top, v_{e_2} v_{e_2}^\top, \ldots, v_{e_n} v_{e_n}^\top) \\
&= \sum_{\{e_1, e_2, \ldots, e_n\} \in \mathcal{C}} \frac{1}{n!} \det(v_{e_1} v_{e_1}^\top + v_{e_2} v_{e_2}^\top + \ldots + v_{e_n} v_{e_n}^\top) = \frac{1}{n!} \sum_{S \in \mathcal{C}} \mu(S).
\end{aligned}$$

It remains to show that the partition family $\mathcal{C}$ can be represented as $\mathcal{C} = \{S \subseteq M : c(S) = C\}$ for some cost vector $c \in \mathbb{Z}^M$ and $C \in \mathbb{Z}$, such that $\|c\|_1 = 2^{O(n \log n)}$. Indeed, by a reasoning as in Corollary 10 we can represent $\mathcal{C}$ as a linear family with n constraints of the form $c^{(i)}(S) = 1$ for $i = 1, 2, \ldots, n$ and $c^{(i)} \in \{0,1\}^{n \times n}$. It is not hard to see that these can be combined into one constraint $c(S) = C$ with $\|c\|_1 = (n^2)^{n + O(1)} = 2^{O(n \log n)}$. Now, it remains to observe that all the steps of the reduction are efficient (since the cost vector is represented in binary here). ◀

Proof of Theorem 6. In light of Lemma 12, the problem of computing $\sum_{S \subseteq [m], c(S) = C} \mu(S)$ for determinantal functions μ is at least as hard as computing mixed discriminants. The BCount problem is very similar, with the only difference that it is computing the sum over all sets of cost $c(S)$ at most C. However, clearly by solving the BCount problem for C and $C - 1$ one can compute $\sum_{S \subseteq [m], c(S) = C} \mu(S)$ by just subtracting the obtained results. ◀

5 Mixed Discriminants and Mixed Characteristic Polynomials

Mixed Characteristic Polynomials played a crucial role in the proof of the Kadison-Singer conjecture. Making this proof algorithmic is an outstanding open question that naturally leads

to the problem of computing the maximum root of these mixed characteristic polynomials. In this section, we show how Corollary 10 implies a polynomial time algorithm for higher-order coefficients of such polynomials. We start by defining mixed characteristic polynomials. We use the following simplified notation for partial derivatives: $\partial_{x_i} f(x)$ is an abbreviation for $\frac{\partial}{\partial x_i} f(x)$.

▶ **Definition 14.** Let $A_1, A_2, \ldots, A_m \in \mathbb{R}^{d\times d}$ be symmetric matrices of dimension d. The mixed characteristic polynomial of $A_1, A_2, \ldots, A_m$ is defined as

$$\mu[A_1, \ldots, A_m](x) \stackrel{\text{def}}{=} \prod_{i=1}^{m} (1 - \partial_{z_i}) \det\left(xI + \sum_{i=1}^{m} z_i A_i\right)\Big|_{z_1=\cdots=z_m=0}.$$

Note in particular that while mixed discriminants are defined for a tuple whose length matches the dimension d of the matrices, for the case of mixed characteristic polynomials the number m can be arbitrary. In fact, when $m = d$, the constant term in the mixed characteristic polynomial is (up to sign) equal to the mixed discriminant of the input tuple.

However, one may wonder whether all of the coefficients in these polynomials are hard to compute. The following result shows that higher-degree coefficients are computable in polynomial time. Roughly, the proof relies on the observation that the higher-degree coefficients in the mixed characteristic polynomial are sums of mixed discriminants that only have constantly many *distinct* matrices. As we demonstrate, computing such mixed discriminants reduces to counting for DPPs under partition constraints with a constant number of parts, which allows us to apply Corollary 10. The formal statement of the theorem follows [2].

▶ **Theorem 15.** *Given a set of m symmetric, PSD matrices $A_1, \ldots, A_m \in \mathbb{R}^{d\times d}$, one can compute the coefficient of x^{d-k} in $\mu[A_1, \ldots, A_m](x)$, in $\mathrm{poly}(m^k)$ time.*

An important component in the proof of Theorem 15 is a reduction from counting for partition constrained DPPs to mixed discriminants. In fact we use it as a subroutine for computing higher-order coefficients of the mixed characteristic polynomial. In Section 4 we provided a reduction in the opposite direction, thus establishing an *equivalence* between mixed discriminants and counting for partition constrained DPPs.

▶ **Lemma 16.** *Given a set of m vectors $v_1, \ldots, v_m \in \mathbb{R}^r$ and a partition of $[m] = P_1 \cup \cdots \cup P_p$ into disjoint, non-empty sets, consider a partition family $\mathcal{C} = \{S \subseteq [m] : |S \cap P_j| = b_j$ for every $j = 1, 2, \ldots, p\}$ such that $\sum_{j=1}^{p} b_j = r$. Let $(A_1, \ldots, A_r)$ be an r-tuple of PSD $r \times r$ matrices such that $(A_1, A_2, \ldots, A_r) = (\overbrace{B_1, \ldots, B_1}^{b_1 \text{ times}}, \overbrace{B_2, \ldots, B_2}^{b_2 \text{ times}}, \ldots, \overbrace{B_p, \ldots, B_p}^{b_p \text{ times}})$ where $B_i = \sum_{e\in P_i} v_e v_e^\top$ for every partition P_i, the following equality holds:*

$$\prod_{i=1}^{p} b_i! \cdot D(A_1, A_2, \ldots, A_r) = \sum_{S\in\mathcal{C}} \det(V_S V_S^\top),$$

where $V \in \mathbb{R}^{m\times r}$ denotes the matrix formed by arranging the vectors $v_1, \ldots, v_m$ row-wise.

Proof. Consider the quantities B_i and $(A_1, A_2, \ldots, A_r)$ as defined in the theorem. By applying linearity multiple times to all coordinates of $D(A_1, A_2, \ldots, A_r)$ we find that:

$$D(A_1, A_2, \ldots, A_r) = \alpha \sum_{S\in\mathcal{B}} D(v_{e_1} v_{e_1}^\top, v_{e_2} v_{e_2}^\top, \ldots, v_{e_r} v_{e_r}^\top),$$

[2] Independent of our work which first appeared in [9], a recent preprint [2] devise a different algorithm to obtain a similar result

where S is $\{e_1, e_2, \ldots, e_r\}$ in the summation above and α is $\prod_{i=1}^{p} b_i!$. This is because $D(v_{e_1}v_{e_1}^\top, v_{e_2}v_{e_2}^\top, \ldots, v_{e_r}v_{e_r}^\top) = 0$ whenever $e_1, e_2, \ldots, e_r$ are not pairwise distinct. We use Fact 13 again to obtain that

$$D(v_{e_1}v_{e_1}^\top, v_{e_2}v_{e_2}^\top, \ldots, v_{e_r}v_{e_r}^\top) = \frac{1}{r!}\det(v_{e_1}v_{e_1}^\top + v_{e_2}v_{e_2}^\top + \ldots + v_{e_r}v_{e_r}^\top) = \det(V_S V_S^\top).$$

This concludes the proof. Furthermore, it is evident that the r-tuple $(A_1, A_2, \ldots, A_r)$ is efficiently computable given the partition family $\mathcal{C}$ and matrix V. ◀

Proof of Theorem 15. First note that without loss of generality we can assume that $d \leq m$, as otherwise – if $d > m$ we can add $(d-m)$ zero-matrices which does not change the result but places us in the $d \leq m$ case. The starting point of our proof is an observation made in [27] which provides us with another expression for the mixed characteristic polynomial in terms of mixed discriminants:

$$\mu[A_1, \ldots, A_m](x) = \sum_{k=0}^{d} x^{d-k}(-1)^k \sum_{S \in \binom{[m]}{k}} D((A_i)_{i \in S}) \tag{2}$$

where we denote $D(A_1, \ldots, A_k) = \frac{1}{(d-k)!} D(A_1, \ldots, A_k, I, \ldots, I)$ with the identity matrix I repeated $d-k$ times. Therefore, our task reduces to computing $O(m^k)$ mixed discriminants of the form $D(A_1, \ldots, A_k, I, \ldots, I)$. Below we show that such a quantity is computable in $\mathrm{poly}(d^k)$ time which concludes the proof.

Consider the Cholesky decomposition of A_i for $i = 1, 2, \ldots, k+1$ (we set $A_{k+1} = I$ for convenience)

$$A_i = \sum_{j=1}^{d} u_{i,j} u_{i,j}^\top.$$

Let $M = \{(i,j) : 1 \leq i \leq k+1, 1 \leq j \leq d\}$ be the ground set of a partition family of size $m \stackrel{\text{def}}{=} (k+1)d$. Define an $m \times d$ matrix U by placing $u_{i,j}$'s as rows of U.

Further, consider a partition $M = P_1 \cup \cdots \cup P_{k+1}$ with $P_i = \{i\} \times [d]$ for all $i = 1, \ldots, k+1$ and let $b_1 = \ldots = b_k = 1$ and $b_{k+1} = d - k$. This gives rise to a partition family

$$\mathcal{C} = \{T \in M : |T \cap P_i| = b_i \text{ for all } i = 1, \ldots, k+1\}.$$

We claim that

$$\prod_{i=1}^{k+1} b_i! \sum_{T \in \mathcal{C}} \det(U_T U_T^\top) = D(A_1, \ldots, A_k, I \ldots, I). \tag{3}$$

This follows from Lemma 16 by considering this partition family $\mathcal{C}$ and matrix U as defined here. Equation (3) combined with the counting result for DPPs under partition constraints (Corollary 10) conclude the proof. ◀

The second observation is more general in its nature and tries to answer the question whether computing mixed characteristic polynomials is strictly harder than computing mixed discriminants. In fact, as noted above, the coefficients of mixed characteristic polynomials are expressed as sums of (an exponential number of) mixed discriminants. We show that these exponential sums can be computed by evaluating a *single* mixed discriminant of matrices of size at most $d+n$. Moreover, our reduction is *approximation-preserving*, hence demonstrating

that approximating mixed discriminants are computationally equally hard as approximating the coefficients of the mixed characteristic polynomials. We remark that our reduction can be thought of as a generalization of a result for approximating the number of k-matchings in a bipartite graph ([13]).

▶ **Theorem 17.** *Given a tuple of m symmetric, positive semi-definite matrices $A_1, \ldots, A_m \in \mathbb{R}^{d\times d}$ with $d \le m$ and $k \in \{1, \ldots, d\}$, there exist a tuple of $m + d - k$ symmetric, positive semi-definite matrices $B_1, \ldots, B_{m+d-k} \in \mathbb{R}^{(m+d-k)\times(m+d-k)}$ such that the coefficient of x^{d-k} in the mixed characteristic polynomial $\mu[A_1, \ldots, A_m](x)$,*

$$\sum_{S\in\binom{[m]}{k}} D((A_i)_{i\in S}) = \frac{1}{(m-k)!(d-k)!} D(B_1, \ldots, B_{m+d-k})$$

Proof. We first show how to construct the $m + d - k$ matrices $B_1, \ldots, B_{m+d-k}$ from $A_1, \ldots, A_m$. The matrices $B_1, \ldots, B_{m+d-k}$ that we consider are 2-by-2 block diagonal matrices that we construct by taking appropriate direct sums. Recall that the direct sum of two matrices A and B of size $d_1 \times d_1$ and $d_2 \times d_2$ is a matrix of size $(d_1 + d_2) \times (d_1 + d_2)$ defined as

$$G = \left[\begin{array}{c|c} A & \mathbf{0}_{d_1\times d_2} \\ \hline \mathbf{0}_{d_2\times d_1} & B \end{array}\right]$$

where $\mathbf{0}_{m\times n}$ is an m-times-n matrix consisting of all zeros. We define the first m matrices to be direct sums of the A_i matrices with the identity matrix of order $m - k$, i.e., I_{m-k} and the remaining $d - k$ matrices to all be equal to the direct sum of the identity matrix of order d, i.e., I_d with the square zero matrix of order $m - k$, i.e., $\mathbf{0}_{m-k}$. Formally,

$$B_i = \begin{cases} A_i \oplus I_{m-k} & \text{for } i \in \{1, \ldots, k\}, \\ I_d \oplus \mathbf{0}_{m-k} & \text{otherwise} \end{cases}$$

We now proceed to prove the claim of the theorem from the definition of the mixed discriminant in Definition 14. For any subset $S \subseteq [m]$, denote $\partial^S = \prod_{i\in S} \partial_{z_i}$.

$$\begin{aligned}
&D(B_1, \ldots, B_{m+d-k}) \\
&= \partial_{z_1} \ldots \partial_{z_{m+d-k}} \det(z_1 B_1 + \ldots + z_{m+d-k} B_{m+d-k}) \\
&= \partial_{z_1} \ldots \partial_{z_{m+d-k}} \det \left[\begin{array}{c|c} \sum_{i=1}^{m} z_i A_i + \sum_{i=1}^{d-k} z_{m+i} I_d & \mathbf{0}_{d\times(m-k)} \\ \hline \mathbf{0}_{(m-k)\times d} & \sum_{i=1}^{m} z_i I_{m-k} \end{array}\right] \\
&= \partial_{z_1} \ldots \partial_{z_{m+d-k}} (z_1 + \ldots + z_m)^{m-k} \det\left(\sum_{i=1}^{m} z_i A_i + \sum_{i=1}^{d-k} z_{m+i} I_d\right) \\
&= \sum_{\substack{S\subseteq[m] \\ |S|=m-k}} \left[\partial^S (z_1 + \ldots + z_m)^{m-k}\right] \left[\partial^{S^c} \prod_{i=1}^{d-k} \partial_{z_{m+i}} \det\left(\sum_{i=1}^{m} z_i A_i + \sum_{i=1}^{d-k} z_{m+i} I_d\right)\right] \\
&= \sum_{\substack{S\subseteq[m] \\ |S|=m-k}} (m-k)! \partial^{S^c} \prod_{i=1}^{d-k} \partial_{z_{m+i}} \det\Big(\sum_{i\in S^c} z_i A_i + (z_{m+1} + \ldots z_{m+d-k}) I_d\Big)
\end{aligned}$$

$$= (m-k)! \sum_{\substack{S\subseteq[m]\\|S|=k}} D((A_i)_{i\in S}, \overbrace{I,\ldots,I}^{d-k \text{ times}})$$
$$= (m-k)!(d-k)! \sum_{\substack{S\subseteq[m]\\|S|=k}} D((A_i)_{i\in S})$$

The fourth to last equality follows simply from chain rule. Since we have an equality in the expression, the reduction is clearly approximation preserving and we are done. ◀

The above theorem in particular allows us to compute in polynomial time, the mixed characteristic polynomial *exactly*, when the linear matrix subspace spanned by the input matrices has constant dimension. This follows by combining Theorem 17 with Theorem 5.1 in [15].

▶ **Corollary 18.** *Suppose $A_1, A_2, \ldots, A_m \in \mathbb{R}^{d\times d}$ span a linear space of dimension k, then there exists a deterministic algorithm to compute $\mu[A_1,\ldots,A_m](x)$ in $\mathrm{poly}(m^k)$ time.*

Proof. In the proof of Theorem 17, the mixed discriminants computed are not of $A_1,\ldots,A_m$ but rather are of modified matrices. However, it is easy to see that for all tuples on which mixed discriminant is called, the dimension of the linear space spanned by them is at most $k+1$. It is proved in [15] that such mixed discriminants can be computed in $O(m^{2k+2})$ time. ◀

6 Budget-Constrained Sampling and Counting for Regular Matroids

Consider the following problem: given an undirected graph G with weights $c \in \mathbb{R}^m$ on its edges, sample a uniformly random spanning tree of cost at most C in G. This generalizes the problem of sampling uniformly random spanning trees [29] and sampling a random spanning tree of minimum cost [12]. Below we study the generalized version of this problem by considering regular matroids, indeed spanning trees arise as bases of the graphic matroid, which is known to be regular. We prove that the counting and sampling problem in this setting can be solved efficiently whenever c is polynomially bounded.

▶ **Theorem 19** (Counting and Sampling Bases of Matroids)**.** *Let $\mathcal{M}$ be a regular matroid on a ground set $[m]$ with a set of bases $\mathcal{B}$. There exists a counting algorithm which, given a cost vector $c \in \mathbb{Z}^m$ and a value $C \in \mathbb{Z}$, outputs the cardinality of the set $\{S \in \mathcal{B} : c(S) \le C\}$ and a sampling algorithm which, given a cost vector $c \in \mathbb{Z}^m$ and a value $C \in \mathbb{Z}$, outputs a random element in the set $\{S \in \mathcal{B} : c(S) \le C\}$. The running time of both algorithms is polynomial in m and $\|c\|_1$.*

Proof of Theorem 19. Let $\mathcal{M} \subseteq 2^{[m]}$ be a regular matroid and $\mathcal{B} \subseteq 2^{[m]}$ be its set of bases. We prove that the generating polynomial $\sum_{S\in\mathcal{B}} x^S$ is efficiently computable. We use the characterization of regular matroids as those which can be linearly represented by a totally unimodular matrix. In other words, there exists a totally unimodular matrix $A \in \mathbb{Z}^{m\times d}$ such that if we denote by $A_e \in \mathbb{Z}^d$ the e^{th} row of A it holds that:

$$S \in \mathcal{M} \quad \Leftrightarrow \quad \{A_e : e \in S\} \text{ is linearly independent.} \tag{4}$$

Let $r \le d$ be the rank of the matroid $\mathcal{M}$, i.e., the cardinality of any set in $\mathcal{B}$. We claim that without loss of generality one can assume that $d = r$. Indeed, we prove that there is

a submatrix $A' \in \mathbb{Z}^{m \times r}$ of A, such that (4) still holds with A replaced by A'. To this end suppose that $d > r$. It is easy to see that the rank of A is r, otherwise, by (4) there would be a set S of cardinality at least $r+1$ with $S \in \mathcal{M}$. Hence there is a column in A which is a linear combination of the remaining columns, we can freely remove this column from A, while (4) will be still true. By doing so, we finally obtain a matrix A' with exactly r rows, which satisfies (4).

By the fact that A has r columns we have:

$$S \in \mathcal{B} \quad \Leftrightarrow \quad A_S \text{ is nonsingular}, \tag{5}$$

where by A_S we mean the $|S| \times r$ submatrix of A corresponding to rows from S. In particular, for a set $S \subseteq [m]$ of cardinality r we have:

$$S \in \mathcal{B} \quad \Leftrightarrow \quad \det(A_S) \neq 0 \quad \Leftrightarrow \quad \det(A_S^\top A_S) = 1, \tag{6}$$

where the last equivalence follows from A being totally unimodular. Let us now consider the polynomial

$$g(x_1, x_2, \ldots, x_m) = \det\left(\sum_{e=1}^{m} x_e A_e A_e^\top\right).$$

By the Cauchy-Binet theorem we obtain:

$$g(x_1, x_2, \ldots, x_m) = \sum_{|S|=r} \det\left(\sum_{e \in S} x_e A_e A_e^\top\right) = x^S \det(A_S^\top A_S).$$

In other words, g is equal to g_μ – the generating polynomial of the function $\mu : 2^{[m]} \to \mathbb{R}$ given by

$$\mu(S) = \begin{cases} 1 & \text{if } S \in \mathcal{B} \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, since g_μ is efficiently computable, by Theorem 3 the BCount$[\mu, c, C]$ is efficiently solvable. This fact, together with Theorem 21 imply that sampling also can be made efficient. ◀

References

1. N. Anari, S. O. Gharan, and A. Rezaei. Monte Carlo Markov Chain Algorithms for Sampling Strongly Rayleigh Distributions and Determinantal Point Processes. In *COLT*, pages 103–115, 2016.
2. N. Anari, S. O. Gharan, A. Saberi, and N. Srivastava. Approximating the largest root and applications to interlacing families. *CoRR*, abs/1704.03892, 2017. URL: `http://arxiv.org/abs/1704.03892`.
3. R. B. Bapat. Mixed discriminants of positive semidefinite matrices. *Lin. Algebra & Applications*, 126, 1989.
4. A. Barvinok. Computing mixed discriminants, mixed volumes, and permanents. *Discrete & Computational Geometry*, 18, 1997.
5. D. Bertsimas and S. Vempala. Solving convex programs by random walks. *J. ACM*, July 2004.
6. A. Z. Broder and E. W. Mayr. Counting minimum weight spanning trees. *J. of Algorithms*, 24(1), 1997.

7 L. E. Celis, A. Deshpande, T. Kathuria, and N. K. Vishnoi. How to be fair and diverse? *Fairness, Accountability, and Transparency in Machine Learning*, 2016.

8 C. Chekuri, J. Vondrak, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*, 2010.

9 A. Deshpande, T. Kathuria, D. Straszak, and N. K. Vishnoi. Combinatorial Determinantal Point Processes. *ArXiv e-prints*, 2016. `arXiv:1608.00554`.

10 A. Deshpande and L. Rademacher. Efficient Volume Sampling for Row/Column Subset Selection. In *FOCS*, Oct 2010.

11 M. E. Dyer, A. M. Frieze, and R. Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991.

12 D. Eppstein. *Representing all minimum spanning trees with applications to counting and generation*. UC Irvine, 1995.

13 S. Friedland and D. Levy. A polynomial-time approximation algorithm for the number of k-matchings in bipartite graphs. *ArXiv*, 2006. `arXiv:0607135`.

14 M. Gartrell, U. Paquet, and N. Koenigstein. Bayesian low-rank determinantal point processes. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 349–356, 2016.

15 L. Gurvits. On the complexity of mixed discriminants and related problems. In *MFCS*, 2005.

16 L. Gurvits and A. Samorodnitsky. A deterministic algorithm for approximating the mixed discriminant and mixed volume, and a combinatorial corollary. *Discrete & Computational Geometry*, 27(4), 2002.

17 B. Hajek. Cooling schedules for optimal annealing. *Mathematics of operations research*, 13(2), 1988.

18 N. Harvey. An introduction to the Kadison-Singer problem and the paving conjecture, 2013.

19 N. Harvey and N. Olver. Pipage rounding, pessimistic estimators and matrix concentration. In *SODA'14*, pages 926–945, 2014.

20 J. B. Hough, M. Krishnapur, Y. Peres, and B. Virág. Determinantal Processes and Independence. *ArXiv Mathematics e-prints*, March 2005. `arXiv:math/0503110`.

21 M. Jerrum, A. Sinclair, and E. Vigoda. A Polynomial-time Approximation Algorithm for the Permanent of a Matrix with Nonnegative Entries. *J. ACM*, 51(4), July 2004.

22 M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

23 A. Krause, A. Singh, and C. Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *J. Mach. Learn. Res.*, 9, June 2008.

24 A. Kulesza and B. Taskar. Determinantal point processes for machine learning. *ArXiv*, July 2012. `arXiv:1207.6083`.

25 C. Li, S. Jegelka, and S. Sra. Markov chain sampling in discrete probabilistic models with constraints. In *NIPS*, 2016.

26 H. Lin and J. Bilmes. A Class of Submodular Functions for Document Summarization. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT'11, 2011.

27 A. W. Marcus, D. A. Spielman, and N. Srivastava. Interlacing families. II: Mixed characteristic polynomials and the Kadison-Singer problem. *Ann. Math. (2)*, 182(1):327–350, 2015.

28 M. Mezard and A. Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

29 R. Pemantle. Uniform random spanning trees. *arXiv preprint math/0404099*, 2004.

30 A. Prasad, S. Jegelka, and D. Batra. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *Advances in Neural Information Processing Systems, December 8-13 2014, Montreal, Quebec, Canada*, pages 2645–2653, 2014.

31 P. Rebeschini and A. Karbasi. Fast mixing for discrete point processes. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 1480–1500, 2015.

32 Damian Straszak and Nisheeth K. Vishnoi. Real stable polynomials and matroids: Optimization and counting. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing*. ACM, 2017.

33 M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

34 K. Wei, R. K. Iyer, and J. A. Bilmes. Submodularity in data subset selection and active learning. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1954–1963, 2015.

35 Y. Yue and T. Joachims. Predicting Diverse Subsets Using Structural SVMs. In *ICML*, 2008.

36 C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval. In *SIGIR*, 2003.

37 T. Zhou, Z. Kuscsik, J. Liu, M. Medo, J. R. Wakeling, and Y. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *PNAS*, 107(10), 2010.

A Hardness for Spanning Trees

We show that BCount is at least as hard as counting perfect matchings in a non-bipartite graph. The proof relies on a combinatorial reduction from counting perfect matchings in a graph to counting budget constrained spanning trees.

▶ **Theorem 20.** *There is a polynomial time reduction which given a graph $G = (V, E)$ with n vertices and m edges outputs a graph G' with n vertices and $O(m + n^2)$ edges, a cost vector $c \in \mathbb{N}^m$ with $\|c\|_1 \leq 2^{O(m \log m)}$ and a value $C \in \mathbb{N}$, such that:*

$$PM(G) = \alpha \cdot ST_C(G')$$

where $PM(G)$ denotes the number of perfect matchings in G, $ST_C(G')$ denotes the number of spanning trees of total cost C in G' and $\alpha = \frac{n^2}{2}(2n)^{-n/2}$.

Proof. Let $G = (V, E)$ be an undirected graph, let $n = |V|$ and $m = |E|$. We construct a new graph G' and a cost vector c, such that counting perfect matchings in G is equivalent to counting spanning trees of specified cost C in G' .

The graph $G' = (V, E')$ is obtained by adding a complete graph to G, i.e., $\binom{n}{2}$ edges, one between every pair of vertices. We call the set of new edges F, hence $E' = E \cup F$. Note that E' is a multiset. To all edges $e \in F$ we assign cost $c_e = 0$, while for the original edges the costs are positive and defined below.

Let $b = m' + 1$, where $m' = |E'|$ is the number of edges in G'. We define the cost of an edge $e = ij \in E$ to be:

$$c_e = b^i + b^j.$$

Note that from the choice of b and c it follows that given a cost $c(S)$ of some set $S \subseteq E$, we can exactly compute how many times a given vertex appears as an endpoint of an edge in S. Indeed, if we have:

$$c(S) = \sum_{i=1}^{n} \delta_i b^i$$

such that $0 \leq \delta_i \leq b-1$ (the $b-$ary representation of $c(S)$), then the degree of vertex i in S is δ_i. This follows from the fact that b is chosen to avoid carry overs when computing $c(S)$ in the $b-$ary numerical system. Therefore, it is now a natural choice to define $C \stackrel{\text{def}}{=} \sum_{i=1}^{n} b^i$. We claim that every perfect matching in G corresponds to exactly $\alpha = \frac{n^2}{2}(2n)^{-n/2}$ different spanning trees of cost C in G'.

To prove this claim, fix any spanning tree S of cost $c(S) = C$. Note first that we have $c(S \cap E) = c(S)$ because all of the edges $e \notin E$ have cost 0. Moreover, the set $M \stackrel{\text{def}}{=} S \cap E$ is a perfect matching in G, because $c(M) = C$ implies that the degree of every vertex in M is one. It remains to show that every perfect matching M in G corresponds to exactly α spanning trees of cost C in G.

Fix any perfect matching M_0 in G. We need to calculate how many ways are there to add $\frac{n}{2} - 1$ edges from E' to obtain a spanning tree of G'. By contracting the matching M_0 to $\frac{n}{2}$ vertices and considering edges in E' only, we obtain a complete graph on $\frac{n}{2}$ vertices with 4 parallel edges going between every pair of vertices. The answer is the number of spanning trees of the obtained graph. Cayley's formula easily implies that this number is $4^{\frac{n}{2}-1}\left(\frac{n}{2}\right)^{\frac{n}{2}-2}$ which equals α^{-1}. ◀

B Equivalence Between Counting and Sampling

In this section we state and prove a theorem that implies that the $\textsc{Count}[\mu, \mathcal{C}]$ and $\textsc{Sample}[\mu, \mathcal{C}]$ problems are essentially equivalent. We prove that, for a given type of constraints $\mathcal{C}$, a polynomial time algorithm for counting can be transformed into a polynomial time algorithm for sampling and vice versa. This section follows the convention that $\mu : 2^{[m]} \to \mathbb{R}_{\geq 0}$ is any function that assigns nonnegative values to subsets of $[m]$ and $\mathcal{C} \subseteq 2^{[m]}$ is any family of subsets of $[m]$.

▶ **Theorem 21** (Equivalence Between Approximate Counting and Approximate Sampling)**.** *Consider any function $\mu : 2^{[m]} \to \mathbb{R}_{\geq 0}$ and a family $\mathcal{C}$ of subsets of $[m]$. Let $\mu_{\mathcal{C}} : \mathcal{C} \to [0,1]$ be a distribution over $S \in \mathcal{C}$ such that $\mu_{\mathcal{C}}(S) \propto \mu(S)$. We assume evaluation oracle access to the generating polynomial g_μ of μ, and define the following two problems:*

- **Approximate $\mathcal{C}$-sampling:** *given a precision parameter $\varepsilon > 0$, provide a sample S from a distribution $\rho : \mathcal{C} \to [0,1]$ such that $\|\mu_{\mathcal{C}} - \rho\|_1 < \varepsilon$.*
- **Approximate $\mathcal{C}$-counting:** *given a precision parameter $\varepsilon > 0$, output a number $X \in \mathbb{R}$ such that $X(1+\varepsilon)^{-1} \leq \sum_{S \in \mathcal{C}} \mu(S) \leq X(1+\varepsilon)$.*

The time complexities of the above problems differ by at most a multiplicative factor of $\mathrm{poly}(m, \varepsilon^{-1})$.

▶ Remark. Note that the above theorem establishes equivalence between *approximate* variants of $\textsc{Count}[\mu, \mathcal{C}]$ and $\textsc{Sample}[\mu, \mathcal{C}]$. This is convenient for applications, because the exact counting variants of these problems are often $\#\mathbf{P}-$hard. Still, for some of them, efficient approximation schemes are likely to exist. Further, we mention that the implication from exact counting to exact sampling holds, hence the sampling algorithms that we obtain in this paper are exact.

Theorem 21 follows from a self-reducibility property [22] of the counting problem. Before we present the proof of Theorem 21, we introduce some terminology and state assumptions for the remaining part of this section. The function $\mu : 2^{[m]} \to \mathbb{R}_{\geq 0}$ is given as an evaluation oracle for $g_\mu(x) = \sum_{S \subseteq [m]} \mu(S) x^S$. In particular, we measure complexity with respect to the number of calls to such an oracle. An algorithm which, for a fixed family $\mathcal{C} \subseteq 2^{[m]}$ and every

function μ, given access to g_μ computes $\sum_{S\in\mathcal{C}}\mu(S)$ is called a $\mathcal{C}$-counting oracle. Similarly, we define a $\mathcal{C}$-sampling oracle to be an algorithm which, given access to g_μ, provides samples from the distribution

$$\mu_{\mathcal{C}}(S) \stackrel{\text{def}}{=} \frac{\mu(S)}{\sum_{T\in\mathcal{C}}\mu(T)} \qquad \text{for } S\in\mathcal{C}.$$

B.1 Counting Implies Sampling

We now show how counting implies sampling. It proceeds by inductively conditioning on certain elements not being in the sample. For this idea to work one has to implement conditioning using the $\mathcal{C}-$sampling oracle and access to the generating polynomial only. Below we state the implication from counting to sampling in the exact variant. The approximate variant also holds, with an analogous proof.

▶ **Lemma 22** (Counting Implies Sampling). *Let $\mathcal{C}$ denote a family of subsets of $[m]$. Suppose access to a $\mathcal{C}$-counting oracle is given. Then, there exists a $\mathcal{C}$-sampling oracle which, for any function $\mu : 2^{[m]} \to \mathbb{R}_{\geq 0}$, makes $\mathrm{poly}(m)$ calls to the counting oracle and to g_μ and outputs a sample from the distribution $\mu_{\mathcal{C}}$.*

Proof. Let $\mathbf{S}$ be the random variable corresponding to the sample our algorithm outputs; our goal is to have $\mathbf{S}\sim\mu_{\mathcal{C}}$. The sampling algorithm proceeds as follows: It sequentially considers each element $e\in[m]$ and tries to decide (at random) whether to include $e\in\mathbf{S}$ or not. To do so, it first computes the probability $\mathbb{P}(e\in\mathbf{S})$ *conditioned on all decisions thus far*. It then flips a biased coin with this probability, and includes e in $\mathbf{S}$ according to its outcome. More formally, the sampling algorithm can be described as follows:

1. Input: $V\in\mathbb{R}^{m\times r}$, a number $k\leq r$.
2. Initialize: $Y=\emptyset$, $N=\emptyset$.
3. For $e=1,2,\ldots,m$:
   a. Compute the probability $p=\mathbb{P}(e\in\mathbf{S} : Y\subseteq\mathbf{S}, N\cap\mathbf{S}=\emptyset)$ under the distribution $\mathbf{S}\sim\mu_{\mathcal{C}}$.
   b. Toss a biased coin with success probability p. In case of success add e to the set Y, otherwise add e to N.
4. Output: $\mathbf{S}=Y$.

It is clear that the above algorithm correctly samples from $\mu_{\mathcal{C}}$. It remains to show that $\mathbb{P}(e\in S : Y\subseteq S, N\cap S=\emptyset)$ can be computed efficiently. This follows from Lemma 23 below. ◀

▶ **Lemma 23.** *Let Y and N be disjoint subsets of $[m]$ and consider any $e\in[m]$. Suppose $\mathbf{S}$ is distributed according to $\mu_{\mathcal{C}}$. If we are given access to a $\mathcal{C}$-counting oracle and to g_μ, then $\mathbb{P}(e\in\mathbf{S} : Y\subseteq\mathbf{S}, N\cap\mathbf{S}=\emptyset)$ can be computed in $\mathrm{poly}(m)$ time.*

Proof. Assume $e\in[m]\setminus(Y\cup N)$; otherwise the probability is clearly 0 or 1. Let $Y'=Y\cup\{e\}$, then

$$\mathbb{P}(e\in\mathbf{S} : Y\subseteq\mathbf{S}, N\cap\mathbf{S}=\emptyset) = \frac{\sum_{S\in\mathcal{C}, Y'\subseteq S, N\cap S=\emptyset}\mu(S)}{\sum_{S\in\mathcal{C}, Y\subseteq S, N\cap S=\emptyset}\mu(S)}.$$

We now show how to compute such sums: Introduce a new variable y, and for every $e\in[m]$ define:

$$w_e \stackrel{\text{def}}{=} \begin{cases} yx_e & \text{for } e\in Y,\\ 0 & \text{for } e\in N,\\ x_e & \text{otherwise.}\end{cases}$$

We interpret the expression $g_\mu(w_1, w_2, \ldots, w_m)$ as a generating polynomial for a certain function $\mu'(y) : 2^{[m]} \to \mathbb{R}$; i.e.,

$$g_{\mu'}(x) \stackrel{\text{def}}{=} g_\mu(w_1, w_2, \ldots, w_m) = \sum_{S \cap N = \emptyset} y^{|S \cap Y|} x^S \mu(S).$$

Define a polynomial

$$h(y) \stackrel{\text{def}}{=} \sum_{S \in \mathcal{C}, S \cap N = \emptyset} y^{|S \cap Y|} \mu(S).$$

It follows that $h(y)$ is a polynomial of degree at most $|Y|$. In fact, the sum we are interested in is simply the coefficient of $y^{|Y|}$ in $h(y)$. The last thing to note is that we can compute $h(y)$ exactly by evaluating it for $|Y| + 1$ different values of y and then performing interpolation. Hence, we just need to query the $\mathcal{C}$-counting oracle $(|Y| + 1)$ times giving it μ' as input (for various choices of y).[3] ◀

B.2 Sampling Implies Counting

We show the implication from sampling to counting in Theorem 21. Similarly as for the opposite direction we assume for simplicity that the sampling algorithm is exact, i.e., we prove the following lemma. The approximate variant holds with an analogous proof.

▶ **Lemma 24** (Sampling Implies Counting). *Let $\mathcal{C}$ denote a family of subsets of $[m]$. Suppose we have access to a $\mathcal{C}$-sampling oracle. Then, there exists a $\mathcal{C}$-counting oracle which for any input function $\mu : 2^{[m]} \to \mathbb{R}$ (given as an evaluation oracle for g_μ) and for any precision parameter $\varepsilon > 0$ makes $\mathrm{poly}(m, 1/\varepsilon)$ calls to the sampling oracle, and approximates the sum:*

$$\sum_{S \in \mathcal{C}} \mu(S)$$

within a multiplicative factor of $(1 + \varepsilon)$. The algorithm has failure probability exponentially small in m.

Let us first state the algorithm which we use to solve the counting problem. Later in a sequence of lemmas we explain how to implement it in polynomial time and reason about its correctness. In the description, $\mathbf{S}$ denotes a random variable distributed according to $\mu_{\mathcal{C}}$.

1. Initialize $U \stackrel{\text{def}}{=} [m]$, $X \stackrel{\text{def}}{=} 1$.
2. Repeat
   a. Estimate the probability $\mathbb{P}(\mathbf{S} = U : \mathbf{S} \subseteq U)$, if it is larger than $(1 - \frac{1}{m})$, terminate the loop.
   b. Find an element $e \in U$ so that $\mathbb{P}(e \notin \mathbf{S} : \mathbf{S} \subseteq U) \geq \frac{1}{m^2}$.
   c. Approximate $p_e \stackrel{\text{def}}{=} \mathbb{P}(e \notin \mathbf{S} : \mathbf{S} \subseteq U)$ up to a multiplicative factor $\frac{\varepsilon}{m}$.
   d. Update $X \stackrel{\text{def}}{=} X \cdot \rho_e$, where ρ_e is the estimate for p_e.
   e. Remove e from U, i.e., set $U \stackrel{\text{def}}{=} U \setminus \{e\}$.
3. Return $X \cdot \mu(U)$.

[3] The provided argument does not generalize directly to the case when the counting oracle is only approximate (because of the interpolation step). However, as we need to compute the top coefficient of a polynomial $h(y)$ only, we can alternatively do it by evaluating $h(y)$ and dividing by y^d (for $d = \deg(h)$) at a very large input $y \in \mathbb{R}$.

▶ **Lemma 25.** *Given $U \subseteq [m]$ and $e \in U$, assuming access to a $\mathcal{C}$-sampling oracle, we can approximate the quantity*

$$p_e = \mathbb{P}(e \notin \mathbf{S} : \mathbf{S} \subseteq U)$$

where $\mathbf{S}$ is distributed according to $\mu_{\mathcal{C}}$, up to an additive error $\delta > 0$ in time $\frac{\text{poly}(m)}{\delta^2}$. The probability of failure can be made $\frac{1}{m^c}$ for any $c > 0$.

Proof. We sample a set $S \in \mathcal{C}$ from the distribution $\mathbb{P}(S) \propto \mu(S)$ conditioned on $S \subseteq U$. This can be done using the sampling oracle, however instead of sampling with respect to μ one has to sample with respect to a modified function μ' which is defined as $\mu'(S) = \mu(S)$ for $S \subseteq U$ and $\mu'(S) = 0$ otherwise. Note that the generating polynomial for μ' can be easily obtained from g_μ by just plugging in zeros at positions outside of U. Given a sample S from μ' we define

$$X = \begin{cases} 1 & \text{if } e \notin S, \\ 0 & \text{otherwise.} \end{cases}$$

Repeat the above independently N times, to obtain $X_1, X_2, \ldots, X_N$ and finally compute the estimator:

$$Z = \frac{X_1 + X_2 + \cdots + X_N}{N}.$$

By Chebyshev's inequality, we have:

$$\mathbb{P}(|Z - p_e| \geq \delta) \leq \frac{1}{N\delta^2}.$$

Thus, by taking $N = \frac{\text{poly}(m)}{\delta^2}$ samples, with probability $\geq 1 - \frac{1}{\text{poly}(m)}$ we can obtain an additive error of at most δ. ◀

▶ **Lemma 26.** *If $U \subseteq [m]$ is such that $\mathbb{P}(\mathbf{S} = U : \mathbf{S} \subseteq U) \leq (1 - \frac{1}{m})$ then there exists an element $e \in U$ such that $\mathbb{P}(e \notin \mathbf{S} : \mathbf{S} \subseteq U) \geq \frac{1}{m^2}$, where $\mathbf{S}$ is distributed according to $\mu_{\mathcal{C}}$.*

Proof. Let $\mathbf{T}$ be the random variable $\mathbf{S}$ conditioned on $\mathbf{S} \subseteq U$. Denote $q_e = \mathbb{P}(e \in \mathbf{S} : \mathbf{S} \subseteq U)$, we obtain

$$\sum_{e \in U} q_e = \mathbb{E}(|\mathbf{T}|) \leq \left(1 - \frac{1}{m}\right)|U| + \frac{1}{m}(|U| - 1) = |U| - \frac{1}{m}.$$

The inequality in the above expression follows from the fact that the worst case upper bound would be achieved when the probability of $|\mathbf{T}| = |U|$ is *exactly* $1 - \frac{1}{m}$ and with the remaining probability, $|\mathbf{T}| = |U| - 1$. Hence $\sum_{e \in U}(1 - q_e) \geq \frac{1}{m}$, which implies that $(1 - q_e) \geq \frac{1}{m^2}$ for some $e \in U$. ◀

We are now ready to prove Lemma 24.

Proof of Lemma 24. We have to show that the algorithm given above can be implemented in polynomial time and it gives a correct answer.

Step 2(a) can be easily implemented by taking poly(m) samples conditioned on $\mathbf{S} \subseteq U$ (as in the proof of Lemma 25). This gives us an approximation of $q_U = \mathbb{P}(\mathbf{S} = U : \mathbf{S} \subseteq U)$

up to an additive error of at most m^{-2} with high probability. If the estimate is less than $(1-\frac{1}{2m})$ then with high probability $q_U \leq (1-\frac{1}{m})$ otherwise, with high probability we have

$$\mu(U) \leq \sum_{S \in \mathcal{C}, S \subseteq U} \mu(S) \leq \left(1+\frac{4}{m}\right)\mu(U) \tag{7}$$

and the algorithm terminates.

When performing step 2(b) we have a high probability guarantee for the assumption of Lemma 26 to be satisfied. Hence, we can assume that (by using Lemma 26 and Lemma 25) we can find an element $e \in U$ with $p_e = \mathbb{P}(e \notin \mathbf{S} : \mathbf{S} \subseteq U) \geq \frac{1}{2m^2}$. Again using Lemma 25 we can perform step 2(c) and obtain a multiplicative $(1+\frac{\varepsilon}{m})$-approximation ρ_e to p_e.

Denote the set U at which the algorithm terminated by U' and the elements chosen at various stages of the algorithm by $e_1, e_2, ..., e_l$ with $l = m - |U'|$. The output of the algorithm is:

$$X \stackrel{\text{def}}{=} \rho_{e_1}\rho_{e_2} \cdot \dots \cdot p_{e_l}\mu(U').$$

While the exact value of the sum is

$$Z \stackrel{\text{def}}{=} p_{e_1}p_{e_2} \cdot \dots \cdot p_{e_l} \cdot \sum_{S \in \mathcal{C}, S \subseteq U'} \mu(S).$$

Recall that for every $i = 1, 2, \dots, l$ with high probability it holds that:

$$\left(1+\frac{\varepsilon}{m}\right)^{-1} \leq \frac{p_{e_i}}{\rho_{e_i}} \leq \left(1+\frac{\varepsilon}{m}\right).$$

This, together with (7) implies that with high probability:

$$\left(1+\frac{\varepsilon}{m}\right)^{-l} \leq \frac{X}{Z} \leq \left(1+\frac{\varepsilon}{m}\right)^{l} \cdot \left(1+\frac{4}{m}\right),$$

which finally gives $(1+2\varepsilon)^{-1} \leq \frac{X}{Z} \leq (1+2\varepsilon)$ with high probability, as claimed. Note that the algorithm requires $\text{poly}(m, \frac{1}{\varepsilon})$ samples from the oracle in total. ◀