High Dimensional Discrete Integration over the Hypergrid

Raj Kumar Maity **Arya Mazumdar** **Soumyabrata Pal**

College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01003
{rajkmaity,arya,spal}@cs.umass.edu

Abstract

Recently Ermon et al. (2013) pioneered a way to practically compute approximations to large scale counting or discrete integration problems by using random hashes. The hashes are used to reduce the counting problem into many separate discrete optimization problems. The optimization problems then can be solved by an NP-oracle such as commercial SAT solvers or integer linear programming (ILP) solvers. In particular, Ermon et al. showed that if the domain of integration is $\{0,1\}^n$ then it is possible to obtain a solution within a factor of 16 of the optimal (16-approximation) by this technique.

In many crucial counting tasks, such as computation of partition function of ferromagnetic Potts model, the domain of integration is naturally $\{0,1,\ldots,q-1\}^n, q>2$, the hypergrid. The straightforward extension of Ermon et al.'s method allows a q^2-approximation for this problem. For large values of q, this is undesirable. In this paper, we show an improved technique to obtain an approximation factor of $4+O(1/q^2)$ to this problem. We are able to achieve this by using an idea of optimization over multiple bins of the hash functions, that can be easily implemented by inequality constraints, or even in unconstrained way. The NP oracle in this setting can be simulated by using an ILP solver as in Ermon et. al. We provide simulation results to support the theoretical guarantees of our algorithms.

1 INTRODUCTION

Large scale counting problems, such as computing the permanent of a matrix or computing the partition function of a graphical probabilistic generative model, come up often in variety of inference tasks. These problems can, without loss of generality, be written as *discrete integration*: the summation of evaluations of a nonnegative function $w:\Omega\rightarrow\mathbb{R}_+\cup\{0\}$ over all elements of Ω:

$$S_\Omega(w)\equiv\sum_{\sigma\in\Omega}w(\sigma). \quad (1)$$

These problems can be computationally intractable because of the often exponential (and sometime super-exponential) size of Ω. A special case is the set of problems #P, counting problems associated with the decision problems in NP. For example, one might ask how many variable assignments a given CNF (conjunctive normal form) formula satisfies. The complexity class #P was defined by Valiant (Valiant, 1979), in the context of computing the *permanent* of a matrix. The permanent of a matrix A is defined as,

$$\mathrm{Perm}(A)\equiv\sum_{\sigma\in S_n}\prod_{i=1}^{n}A_{i,\sigma(i)}, \quad (2)$$

where S_n is the symmetric group of n elements and $A_{i,j}$ is the (i,j)-th element of A. Clearly, here S_n is playing the role of Ω, and $w(\sigma)=\prod_{i=1}^{n}A_{i,\sigma(i)}$. Therefore computing permanent of a nonnegative matrix is a canonical example of a problem defined by eq. (1).

Similar counting problems arise when one wants to compute the partition functions of the well-known probabilistic generative models of statistical physics, such as the Ising model, or more generally the Ferromagnetic Potts Model (Potts, 1952). Given a graph $G(V,E)$, and a label-space $Q\equiv\{0,1,2,\ldots,q-1\}$, the partition function $Z(G)$ of the Potts model is given by,

$$\sum_{\sigma\in Q^{|V|}}\exp\Big(-\zeta\Big(J\sum_{(u,v)\in E}\delta(\sigma(u),\sigma(v)) \quad (3)$$
$$+H\sum_{u\in V}\delta(\sigma(u),0)\Big)\Big),$$

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

where ζ, J and H are system-constants (representing the temperature, spin-coupling and external force respectively), $\delta(x, y)$ is the delta-function that is 1 if and only if $x = y$ and otherwise 0, and σ represents a label-vector, where $\sigma(u)$ is the label of vertex u.

It has been shown that, under the availability of an NP-oracle, every problem in #P can be approximated within a factor of $(1 \pm \epsilon), \epsilon > 0$, with high probability via a randomized algorithm (Stockmeyer, 1985). This result says #P can be approximated by $\mathrm{BPP}^{\mathrm{NP}}$ and the power of an NP-oracle and randomization is sufficient. However, depending on the weight function $w(\cdot)$, eq. (1) may not be in #P. There are related approaches to count the number of models of propositional formulas based on SAT-solvers, such as (Birnbaum and Lozinskii, 1999; Jr. and Pehoushek, 2000; Wei and Selman, 2005; Pesant, 2005; Chakraborty et al., 2014; Chakraborty et al., 2016) among others.

The standard techniques to evaluate eq. (1) include the very influential fast variational methods (Wainwright et al., 2008), and Markov-Chain-Monte-Carlo based sampling schemes (Jerrum and Sinclair, 1996). In practice, except for limited number of cases, these approaches are mostly used in a heuristic manner without nonasymptotic qualitative guarantees. Recently, Ermon et al. proposed an alternative approach (that they call `WISH` - Weighted-Integrals-And-Sums-By-Hashing) to solve these counting problems (Ermon et al., 2013a; Ermon et al., 2013b) by breaking them into multiple optimization problems. Namely, they use families of hash functions $h : \Omega \to \tilde{\Omega}, |\tilde{\Omega}| < |\Omega|$, and use a (possibly NP) oracle that can return the correct solution of the optimization problem: $\max_{\sigma : h(\sigma) = a} w(\sigma)$, for any $a \in \tilde{\Omega}$. We call this oracle a MAX-oracle. In particular, when $\Omega = \{0, 1\}^n$, and $h(\cdot)$ is a random hash function, assuming the availability of a MAX-oracle, Ermon et al. (Ermon et al., 2013a) propose a randomized algorithm that approximates the discrete sum within a factor of sixteen (a 16-approximation) with high probability. Ermon et al. use simple linear sketches over $\mathbb{F}_2$ (the finite field of size 2), i.e., the hash function $h_{A,b} : \mathbb{F}_2^n \to \mathbb{F}_2^m, A \in \mathbb{F}_2^{m \times n}, b \in \mathbb{F}_2^m$ is defined to be

$$h_{A,b}(x) = Ax + b, \tag{4}$$

where the arithmetic operations are over $\mathbb{F}_2$. The matrix A and the vector b are randomly and uniformly chosen from the respective sample spaces. The MAX-oracle in this case simply provides solutions to the optimization problem: $\max_{\sigma \in \mathbb{F}_2^n : A\sigma = b} w(\sigma)$.

The constraint space $\{\sigma \in \mathbb{F}_2^n : A\sigma = b\}$ is nice since it is a coset of the nullspace of A, and experimental results showed them to be manageable by optimization softwares/SAT solvers. In particular it was observed that being Integer Programming constraints, real-world instances are often solved in reasonable time. Since the implementation of the hash function heavily affects the runtime, it makes sense to keep constraints of the MAX-oracle as an affine space as above. These constraints are also called parity constraints. The idea of using such constraints to show reduction among class of problems appeared in several papers before, including (Sipser, 1983; Valiant and Vazirani, 1986; Gomes et al., 2006; Thurley, 2011; Gomes et al., 2007) among others. The key property that the hash functions $\{h_{A,b}\}$ satisfy is that they are pairwise independent. This property can be relaxed somewhat - and in a subsequent paper Ermon et al. show that a hash family would work even if the matrix A is sparse and random, thus effectively reducing the randomness as well as making the problem more tractable empirically (Ermon et al., 2014). Subsequently, Achlioptas and Jiang (Achlioptas and Jiang, 2015) have shown another way of achieving similar guarantees. Instead of arriving at the set $\{\sigma \in \mathbb{F}_2^n : A\sigma = b\}$ as a solution of a system of linear equations (over $\mathbb{F}_2$), they view the set as the image of a lower-dimensional space. This is akin to the generator matrix view of a linear error-correcting code as opposed to the parity-check matrix view. This viewpoint allows their MAX-oracle to solve just an unconstrained optimization problem.

Drawbacks of obvious extensions of (Ermon et al., 2013a) to large alphabets. Note that, some crucial counting problems, such as computing the partition function of the Ferromagnetic Potts model of Eq. (3), naturally have $\Omega = \{0, 1, \ldots, q-1\}^n, q > 2$, i.e., a hypergrid. It is worth noting that while there exists polynomial time approximation (FPRAS) for the Ising model ($q = 2$), FPRAS for general Potts model ($q > 2$) is significantly more challenging (and likely impossible (Goldberg and Jerrum, 2012)). There are a few possible obvious extensions of Ermon et al. (Ermon et al., 2013a) to larger alphabets.

- (The straightforward extension). The method of (Ermon et al., 2013a) can be used for q-ary in stead of binary. However, the drawback is that it provides a q^2-approximation at best that is particularly bad if q is large or growing with n.
- (Convert q-ary to binary). To use the binary-domain algorithm of (Ermon et al., 2013a) for any $\Omega = \{0, 1, \ldots, q-1\}^n$, we need to use a look-up table to map q-ary numbers to binary. In this process the number of variables (and also the number of constraints) increases by a factor of $\log q$. This makes the MAX-oracle significantly slower, especially when q is large. Also, for the permanent problem, where $|\Omega| = \exp(n \log n)$, this creates a computational bottleneck. It would be useful to extend the

method of (Ermon et al., 2013a) for $\Omega = \mathbb{F}_q^n$ without increasing the number of variables.

Furthermore, when q is not a power of 2, by converting q-ary configurations to binary, we introduce exponentially many invalid configurations. To account for these, the MAX-oracle must be adjusted accordingly which is a difficult task. This motivates us to keep the problem in its original domain and not convert the domain to binary.

- For the binary setting, it has been noted in (Ermon et al., 2013a, section 5.3) that the approximation ratio can be improved to any $\alpha > 1$ by increasing the number of variables, which extends to this q-ary setting. However this also results in an increase in number of variables by a factor of $\log_\alpha(q^2)$ which is undesirable.

Our contributions. Our first contribution in this paper is to provide a new and improved algorithm to handle counting problems over nonbinary domains. For any hypergrid $\Omega = \{0, 1, \ldots, q-1\}^n$, q is a power of prime, our algorithm provides a $4(1+\frac{1}{q-1})^2$-approximation, when q is odd, and $4(1+\frac{2}{q-2})^2$-approximation, when $q > 2$ is even, to the optimization problem of (1) assuming availability of the MAX-oracle. Our algorithm utilizes an idea of using optimization over multiple bins of the hash function that can be easily implemented via inequality constraints. The constraint space of the MAX-oracle remains an affine space and still can be represented as a modular integer linear program (ILP). Our multi-bin technique can also be used to extend the generator-matrix based algorithm of Achlioptas and Jiang (Achlioptas and Jiang, 2015). As a result, we need the MAX-oracle to only perform unconstrained maximization, as opposed to constrained. This lead to significant speed-up in the system, while resulting in the same approximation guarantees.

Finally, we show the performance of our algorithms to compute the partition function of the ferromagnetic Potts model by running experiments on both synthetic datasets and real-worlds datasets. While in this paper we concentrate on theoretical results, the experiments serve as good ‘proof of concepts’ for applications. We also use our algorithm to compute the Total Variation (TV) distance between two joint probability distributions over a large number of variables. In addition to comparing with the straightforward generalization of Ermon et al.'s method (Ermon et al., 2013a), we also show comparisons with the popular Markov-Chain-Monte-Carlo (MCMC) method and the belief propagation method for discrete integration. All the experiments exhibit good performance guarantees.

Organization. In Section 2 we describe the technique by (Ermon et al., 2013a) called the `WISH` algorithm, and then elaborate our new ideas and main results. In Section 3, we provide the main technical results that lead to an improved approximation. We provide an algorithm with unconstrained optimization oracle (similar to (Achlioptas and Jiang, 2015)) and its analysis in Section 4. The experimental results on computation of partition functions and total variation distance are provided in Section 5. Most of the proofs and some experimental results are delegated to the appendix in the supplementary material.

While only of auxiliary interest here, we note that it is possible to derandomize the hash families based on parity-constraints to the optimal extent while maintaining the essential properties necessary for their performance. Namely, it can be ensured that the hash family can still be represented as $\{x \mapsto Ax + b\}$ while using information theoretically optimal memory to generate them. We discuss this in Appendix C in the supplement.

It turns out that, by using our technique and some modifications to the MAX-oracle, it is possible to obtain close-to-4-approximation to the problem of computing permanent of nonnegative matrices (assuming existence of NP-oracles). The NP-oracle still is amenable to be implemented in a commercial optimization solver. The idea of optimization over multiple bins is crucial here, since the straightforward generalization of Ermon et al.'s result would have given an approximation factor of $\Omega(n^2)$. Since there exists polynomial time randomized approximation scheme ($1 \pm \epsilon$-approximation) of permanent of a nonnegative matrix (Jerrum et al., 2004), the point of this exercise is to show that our method extends to find permanent of a matrix (albeit not with the best guarantees). We discuss this in Appendix D in the supplementary material.

2 BACKGROUND AND OUR TECHNIQUES

In this section we describe the main ideas developed by (Ermon et al., 2013a) and provide an overview of the techniques that we use to arrive at our new results.

Let the elements in Ω be $\sigma_1, \sigma_2, \ldots, \sigma_{|\Omega|}$ arranged according to a decreasing order of their weight, i.e., $w(\sigma_1) \geq w(\sigma_2) \geq \cdots \geq w(\sigma_{|\Omega|})$. Let $\beta_i = w(\sigma_{q^i})$, for $i = 0, 1, \ldots, n'$, where n' is the smallest integer such that $q^{n'} \geq |\Omega|$. When $q^{n'} > |\Omega|$ we set $\beta_{n'} = 0$.

Clearly $\beta_0 \geq \beta_1 \geq \cdots \geq \beta_{n'}$. As we have not made any assumption on the values of the weight function, β_i and β_{i+1} can be far from each other. On the other hand we can try to bound the sum $S_\Omega(w)$ by bounding the area of the slice between β_i and β_{i+1}. This area is at least $q^i(\beta_i - \beta_{i+1})$ and at most $q^{i+1}(\beta_i - \beta_{i+1})$. Therefore: $\sum_{i=0}^{n'-1} q^i(\beta_i - \beta_{i+1}) + q^{n'}\beta_{n'} \leq S_\Omega(w) \leq \sum_{i=0}^{n'-1} q^{i+1}(\beta_i - \beta_{i+1}) + q^{n'}\beta_{n'}$

which implies

$$\beta_0 + (q-1)\sum_{i=1}^{n'} q^{i-1}\beta_i \le S_\Omega(w) \le \beta_0 + (q-1)\sum_{i=1}^{n'} q^{i}\beta_i. \tag{5}$$

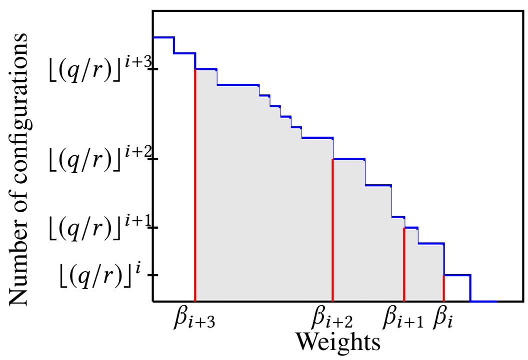

Figure 1: The $T(u)$ vs. u curve and the illustration of β_is.

Hence $\beta_0 + (q-1)\sum_{i=1}^{n'} q^{i-1}\beta_i$ is a q-factor approximation of $S_\Omega(w)$ and if we are able to find a k-approximation of each value of β_i we will be able to obtain a kq-factor approximation of $S_\Omega(w)$. In (Ermon et al., 2013a), subsequently the main idea is to estimate the coefficients $\{\beta_i, 0 \le i \le n'\}$.

Now note that, $q^i = |\{\sigma \in \Omega : w(\sigma) \ge \beta_i\}|$, for $i = 0, 1, \ldots, n'-1$. This also hold for $i = n'$ unless $q^{n'} > |\Omega|$ in which case $\beta_{n'} = 0$. Suppose, using a random hash function $h : \Omega \to \{0, 1, \ldots, q^i - 1\}$ we compute hashes of all elements in Ω. The pre-image of an entry in $\{0, 1, \ldots, q^i - 1\}$ is called the *bin* corresponding to that value, i.e., $\{\sigma \in \Omega : h(\sigma) = x\}$ is the bin corresponding to the value $x \in \{0, 1, \ldots, q^i - 1\}$. In every bin for the hash function, there is on average one element σ such that $w(\sigma) \ge \beta_i$. So for a randomly and arbitrarily chosen bin $x \in \{0, 1, \ldots, q^i - 1\}$, if $w^* = \max_{\sigma : h(\sigma)=x} w(\sigma)$, then w^* is a 'good' approximation of β_i (this will be made rigorous later). Indeed, suppose one performs this random hashing $\ell = O(\log n')$ times and then take the aggregate (in this case the median) value of w^*s. That is, let $\hat{w}^* = \text{median}(w_1^*, \ldots, w_\ell^*)$; then by using the independence of the hash functions, it can be shown that the aggregate is an upper bound on β_i with high probability. In (Ermon et al., 2013a), $\Omega = \mathbb{F}_2^n$ and if the hash family is pairwise independent, then by using the Chebyshev inequality it was shown that $\hat{w}^* \in [\beta_{i+2}, \beta_{i-2}]$ with high probability. The `WISH` algorithm proposed by (Ermon et al., 2013a) makes use of the above analysis and provides a $2^{2\cdot 2} = 16$-approximation of $S_w(\Omega)$. If we naively extend this algorithm for $S_w(\Omega) = \mathbb{F}_q^n, q > 2$, then it can be shown that $\hat{w}^* \in [\beta_{i+1}, \beta_{i-1}]$ with high probability. This gives an approximation factor of q^2. E.g., for a ternary alphabet, $\Omega = \mathbb{F}_3^n$, we have a 9-approximation to $S_w(\Omega)$.

Instead of using a straightforward analysis for the q-ary case, in this paper we use a MAX-oracle that can optimize over multiple bins of the hash function. Using this oracle we proposed a modified `WISH` algorithm and call it `MB-WISH` (Multi-Bin `WISH`). Just as in the case of (Ermon et al., 2013a; Ermon et al., 2014), the MAX-oracle constraints can be integer linear programming constraints and commercial softwares such as CPLEX can be used. The main intuition of using an optimization over multiple bins is that it boosts the probability that the w^* we are getting above is close to β_i. To be precise, we redefined $\beta_i \equiv w(\sigma_{\lfloor(\frac{q}{r})^i\rfloor})$ for $i = 1, 2, \ldots, n' \equiv \lceil n \log_{q/r} q \rceil$. If we define $T(u) \equiv |\{\sigma \in \Omega : w(\sigma) \ge u\}|$, then Figure 1 illustrate the $T(u)$ vs. u curve and locates β_is therein. Note that, we would like to find the area under the $T(u)$ vs. u curve, for which we use the sum of the vertical slices. Now to estimate the new β_i, we choose a hash function as before, and optimize over r^i bins of the hash function. These steps are made rigorous in Section 3. However if we restrict ourselves to the binary alphabet then (as will be clear later) there is no immediate way to represent such multiple bins in a compact way in the MAX-oracle. For the non-binary case, it is possible to represent multiple bins of the hash function as simple inequality constraints.

This idea leads to an improvement in the approximation factor of $S_w(\Omega)$ to $4 + \epsilon$, where ϵ decays to 0 proportional to q^{-1}. Note that we need to choose q to be a power of prime so that $\mathbb{F}_q$ is a field.

In (Achlioptas and Jiang, 2015), the bins (as described above) are produced as images of some function, and not as pre-images of hashes. Since we want the number of bins to be q^i, this can be achieved by looking at images of $g : \mathbb{F}_q^{n-i} \to \Omega$ where $|\{g(\sigma) : \sigma \in \mathbb{F}_q^{n-i}\}| = q^{n-i}$. The rest of the analysis of (Achlioptas and Jiang, 2015) is almost same as above. The benefit of this approach is that the MAX-oracle just has to solve an unconstrained optimization here. Implementing our multi-bin idea for this perspective of (Achlioptas and Jiang, 2015) is not straightforward as we can no longer use inequality constraints for this. However, as we show later, we found a way to combine bins here in a succinct way generalizing the design of g. As a result, we get the same approximation guarantee as in `MB-WISH`, with the oracle load heavily reduced (this algorithm, that we call `Unconstrained MB-WISH`, can be found in Section 4).

3 THE `MB-WISH` ALGORITHM AND ANALYSIS

Let us assume $\Omega = \mathbb{F}_q^n$ where q is a prime-power. Let us also fix an ordering among the elements of $\mathbb{F}_q \equiv \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$ and write $\alpha_0 \prec \alpha_1 \prec \cdots \prec \alpha_{q-1}$. In this section, the symbol '$\prec$' just signifies a fixed ordering and has no real meaning over the finite field. Extending this notation, for any two distinct vectors $x, y \in \mathbb{F}_q^m$, we will say $x \prec y$ if and only if the ith coordinates of x and y, satisfy $x_i \prec y_i$ for all $i = 1, \ldots, m$. Below $\mathbf{1}$ denotes an all-one vector of a dimension that would be clear from context. Also, for any event $\mathcal{E}$ let $\mathbb{1}[\mathcal{E}]$ denote the indicator for the event $\mathcal{E}$.

The MAX-oracle for MB-WISH performs the following optimization, given $A \in \mathbb{F}_q^{m\times n}; b, s \in \mathbb{F}_q^m$:

$$\max_{\sigma \in \mathbb{F}_q^n : A\sigma + b \prec s} w(\sigma). \tag{6}$$

The modified WISH algorithm is presented as Algorithm 1. The main result of this section is below.

Algorithm 1 MB-WISH algorithm for $\Omega = \mathbb{F}_q^n$, a weight function w and an input parameter $r \leq \lfloor \frac{q-1}{2} \rfloor$

Initialize: $\gamma = \frac{q}{3r}(\frac{1}{2} - \frac{r}{q})^2$, $\ell = \lceil \frac{1}{\gamma} \ln \frac{2n}{\delta} \rceil$, $n' = \lceil n \log_{q/r} q \rceil$
  $M_0 \equiv \max_{\sigma \in \mathbb{F}_q^n} w(\sigma)$
  for $i \in \{1, 2, \ldots, n'\}$ **do**
    for $k \in \{1, \ldots, \ell\}$ **do**
      Sample hash functions $h_i \equiv h_{A^i, b^i}$ uniformly at random from $\mathcal{H}_{i,n}$ as defined in (7)
      $w_i^{(k)} = \max_{\sigma : A^i\sigma + b^i \prec \alpha_r \cdot \mathbf{1}} w(\sigma)$
    end for
    $M_i = \text{Median}(w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(\ell)})$
  end for
  Return $M_0 + (\frac{q}{r} - 1) \sum_{i=0}^{n'-1} M_{i+1} \left(\frac{q}{r}\right)^i$

Theorem 1. *Suppose $q > 2$ is a prime power, $\Omega = \mathbb{F}_q^n$ and $r \leq \lfloor \frac{q-1}{2} \rfloor$ is a positive integer. For any $\delta > 0$, Algorithm 1 makes $\Theta(n \log \frac{n}{\delta})$ calls to the MAX-oracle, and with probability $\geq 1 - \delta$ outputs a $(\frac{q}{r})^2$-approximation of $S_w(\Omega)$.*

By setting $r = \lfloor \frac{q-1}{2} \rfloor$, our algorithm provides a $4(1 + \frac{1}{q-1})^2$-approximation, when q is odd, and $4(1 + \frac{2}{q-2})^2$-approximation, when $q > 2$ is even.

The constant in the big-O term in the number of calls to the oracle is a function of q and r. In particular, when $r = \lfloor \frac{q-1}{2} \rfloor$ and q odd, this constant varies as $q^2 \log q$. We can tune the value of r to reduce the number of calls to the oracle at the expense of the approximation factor.

The theorem will be proved by a series of lemmas. The key trick that we are using is to ask the MAX-oracle to solve an optimization problem over not a single bin, but *multiple bins* of the hash function. This is going to boost the probability that our estimates of β_is are good. In particular we will solve the optimization over r^m bins of the hash function. The hash family is defined in the following way. We have $h_{A,b} : \mathbb{F}^n \to \mathbb{F}^m : x \mapsto Ax + b$, the operations are over $\mathbb{F}_q$. Let

$$\mathcal{H}_{m,n} = \{h_{A,b} : A \in \mathbb{F}_q^{m\times n}, b \in \mathbb{F}_q^m\}. \tag{7}$$

For readers familiar with coding theory, the basis behind our technique is simple. The set of configurations $\{\sigma \in \mathbb{F}_q^n : A\sigma = \mathbf{0}\}$ forms a linear code of dimension $n - m$. The bins of the hash function define the cosets of this linear code. We would like to chose q^r cosets of a random linear code and the find the optimum value of w over the configurations of these cosets as the MAX-oracle. To choose a hash function uniformly and randomly from $\mathcal{H}$, we can just choose the entries of A and b uniformly at random from $\mathbb{F}_q$ independently.

Note that, the hash family $\mathcal{H}_{m,n}$ as defined in (7) is uniform and pairwise independent.

Lemma 1. *Let us define Z_σ to be the indicator random variable denoting $A\sigma + b \prec \alpha_r \cdot \mathbf{1}$ for some $r \in \{0, \ldots, q-1\}$ and A, b randomly and uniformly sampled from $\mathcal{H}_{m,n}$. Then $\Pr(Z_\sigma = 1) = \left(\frac{r}{q}\right)^m$ and for any two distinct configurations $\sigma_1, \sigma_2 \in \mathbb{F}_q^n$ the random variables Z_{σ_1} and Z_{σ_2} are independent.*

Fix an ordering of the configurations $(\sigma_i, 1 \leq i \leq q^n)$ such that $1 \leq j \leq q^n, w(\sigma_j) \geq w(\sigma_{j+1})$. For $i \in \{0, 1, 2, \ldots, n' \equiv \lceil n \log_{q/r} q \rceil\}$, define $\beta_i = w(\sigma_{\lfloor t^i \rfloor})$, where $t = \frac{q}{r}$. We take $w(\sigma_k) = 0$ for $k > q^n$. See Figure 1 for an illustration.

To prove Thm. 1 we need the following crucial lemma.

Lemma 2. *Let $M_i = \text{Median}(w_i^{(1)}, \ldots, w_i^{(\ell)})$ be defined as in the Algorithm 1. Then for $\gamma = \frac{q}{3r}(\frac{1}{2} - \frac{r}{q})^2$, we have,*

$$\Pr\left(M_i \in [\beta_{\min(i+1,n')}, \beta_{\max(i-1,0)}]\right) \geq 1 - 2\exp(-\gamma\ell).$$

From Lemma 2, the output of the algorithm lies in the range $[L', U']$ with probability at least $1 - \delta$ where $L' = \beta_0 + (t-1)\sum_{i=0}^{n'-1} \beta_{\min\{i+2,n'\}} t^i$ and $U' = \beta_0 + (t-1)\sum_{i=0}^{n'-1} \beta_i t^i$. L' and U' are a factor of t^2 apart. Now, following an argument similar to (5), we can show $L' \leq S_w(\Omega) \leq U'$. Therefore Algorithm 1 provides a t^2-approximation to $S_\Omega(w)$ and the total number of calls to the MAX-oracle is $n'\ell + 1 = O(n \log(n/\delta))$. The full proof of Theorem 1 is deferred to the Appendix A in the supplementary material.

To exemplify this result, suppose $q = 3$. In this case the algorithm provides a 9-approximation. Later, in the experimental section, we have used a ferromagnetic Potts model with $q = 5$. MB-WISH provides a $\frac{25}{4} = 6.25$-approximation in that case. Note that, for a 5-ary Potts model, it is only natural to use our algorithm instead of converting it to binary in conjunction with the original algorithm of Ermon et al.

Instead of pairwise independent hash families, if we employ k-wise independent families, it leads to a better decay probability of error. However it does not improve the approximation factor.

4 `MB-WISH` WITH UNCONSTRAINED OPTIMIZATION ORACLE

In this section, we modify and generalize the results of Achlioptas and Jiang (Achlioptas and Jiang, 2015) to formulate a version of `MB-WISH` that can use unconstrained optimizers as the MAX-oracle. We call this algorithm `Unconstrained MB-WISH`. Let us assume $\Omega = \mathbb{F}_q^n$ where q is a prime-power. As before, let us fix an ordering among the elements of $\mathbb{F}_q \equiv \{\alpha_0, \alpha_1, \dots, \alpha_{q-1}\}$ and write $\alpha_0 \prec \alpha_1 \prec \dots \prec \alpha_{q-1}$. Recall that, here the symbol '$\prec$' signifies a fixed ordering and has no real meaning over the finite field.

The MAX-oracle for `Unconstrained MB-WISH` performs an unconstrained optimization of the following form, given $A \in \mathbb{F}_q^{n\times m}, b \in \mathbb{F}_q^n$ and a set $B \subseteq \mathbb{F}_q^m$:

$$\max_{\sigma \in B} w(A\sigma + b). \tag{8}$$

The aim is to carefully design B so that all the desirable statistical properties are satisfied. This part is quite different from the hashing-based analysis and not an immediate extension of (Achlioptas and Jiang, 2015). We provide the algorithm (`Unconstrained MB-WISH`) and its analysis in the next section.

The `Unconstrained MB-WISH` algorithm is presented as Algorithm 2. The main result of this section is the following.

Theorem 2. *Suppose $q > 2$ is a power of a prime and a positive integer $r \le \lfloor \frac{q-1}{2} \rfloor$. Let $\Omega = \mathbb{F}_q^n$. For any $\delta > 0$, Algorithm 2 makes $\Theta(n \log \frac{n}{\delta})$ calls to the MAX-oracle (cf. (8)), and with probability at least $1-\delta$ outputs a $(\frac{q}{r})^2$-approximation of $S_w(\Omega)$.*

To prove this theorem we borrow some ideas from coding theory. We define a linear q-ary code C of dimension $n-m$ and length n as the set of vectors $\{Ax : x \in \mathbb{F}_q^{n-m}\}$ where A is a full-rank matrix of size $n \times n-m$ and rank $n-m$. For a vector $a \in \mathbb{F}_q^n$, we define the set $\{a + C\}$ as a coset of C. It is well known that $\mathbb{F}_q^n$ is partitioned by the q^m distinct cosets, each of size q^{n-m}. The main technique behind our algorithm is that for a random linear code C of size q^{n-m}, we randomly sample r^m distinct cosets of C. Subsequently, we find the maximum value $w(x)$ of an element among those r^m cosets.

Algorithm 2 `Unconstrained MB-WISH` algorithm for $\Omega = \mathbb{F}_q^n$ and a weight function w

Initialize: $\ell \to \lceil \frac{1}{\gamma} \ln \frac{2n}{\delta} \rceil$, $r, n' = \lceil n \log_{q/r} q \rceil$
  $M_0 \equiv \max_{\sigma \in \mathbb{F}_q^n} w(\sigma)$
  for $i \in \{1, 2, \dots, n\}$ **do**
    for $k \in \{1, \dots, \ell\}$ **do**
      Sample a full rank matrix uniformly at random from the set of all full rank $n \times n$ matrices in $\mathbb{F}_q^{n\times n}$ and construct matrices A and R by taking the first $n-i$ columns and the last i columns respectively.
      Sample $b \in \mathbb{F}_q^n$ uniformly at random
      $w_i^{(k)} = \max_{\substack{x \in \mathbb{F}_q^{n-i} \\ y \in \{\alpha_0, \alpha_1, \dots, \alpha_{r-1}\}^i}} w(Ax + Ry + b)$
    end for
    $M_i = \text{Median}(w_i^{(1)}, w_i^{(2)}, \dots, w_i^{(\ell)})$
  end for
  for $i \in \{n+1, \dots, n'\}$ **do**
    for $k \in \{1, \dots, \ell\}$ **do**
      Sample full rank matrix $A \in \mathbb{F}_q^{n\times n}, b \in \mathbb{F}_q^n$ uniformly at random. Set $\mathcal{S}_i$ as defined in Equation (10)
      $w_i^{(k)} = \max_{y \in \mathcal{S}_i} w(Ay + b)$
    end for
    $M_i = \text{Median}(w_i^{(1)}, w_i^{(2)}, \dots, w_i^{(\ell)})$
  end for
  Return $M_0 + (\frac{q}{r} - 1) \sum_{i=0}^{n'-1} M_{i+1} \left(\frac{q}{r}\right)^i$

Let $E \in \mathbb{F}_q^{n\times n}$ be an $n \times n$ full rank matrix randomly and uniformly chosen from the set of all $n \times n$ rank-n matrices over $\mathbb{F}_q$. One can choose such a matrix via rejection sampling: independently and uniformly sample the entries of the matrix from $\mathbb{F}_q$ and then reject the matrix and resample it if it is not full rank. Let A denote the random matrix formed by the first $n-m$ columns of E as columns and let R be the random matrix formed by the remaining m columns of E as columns. Also let b be a vector sampled randomly and uniformly from $\mathbb{F}_q^n$. The MAX-oracle for `Unconstrained MB-WISH` is going to perform the following optimization when $m \le n$:

$$\max_{\sigma_1 \in \mathbb{F}_q^{n-m}, \sigma_2 \in \{\alpha_0, \alpha_1, \dots, \alpha_{r-1}\}^m} w(A\sigma_1 + R\sigma_2 + b). \tag{9}$$

Analogous to Theorem 1, here we are creating union of r^m distinct random bins. If we can prove that, for any element of $\mathbb{F}_q^n$, the probability that it belongs to one of these bins is $(\frac{r}{q})^m$ and for any pair of different elements from $\mathbb{F}_q^n$, whether they belong to one of these bins are independent (pairwise independence), the rest of the proof of Theorem 2 will just follow that of Theorem 1.

In particular, we just have to prove the lemma that is analogous to Lemma 1. Define a set

$$S_{A,R,b} \equiv \{Ax + b + Ry \mid x \in \mathbb{F}_q^{n-m},\ y \in \{\alpha_0, \alpha_1, \ldots, \alpha_{r-1}\}^m\}.$$

For each configuration $\sigma \in \mathbb{F}_q^n$, associate an indicator random variable Z_σ denoting whether $\sigma \in S_{A,R,b}$.

Lemma 3. *For each configuration $\sigma \in \mathbb{F}_q^n$, we must have* $\Pr(Z_\sigma = 1) = \left(\frac{r}{q}\right)^m$ *and moreover for any two distinct distinct configurations $\sigma_1, \sigma_2 \in \mathbb{F}_q^n$, we must have* $\Pr(Z_{\sigma_1} = 1 \wedge Z_{\sigma_2} = 1) \leq (\Pr(Z_\sigma = 1))^2$.

Although the two random variables Z_{σ_1} and Z_{σ_2} defined above are not independent, we show that they are *negatively correlated*. Note that, the pairwise independence was then subsequently used in computing a variance for the Chebyshev's inequality (see Lemma 2). However, the negative correlation is sufficient to obtain an upper bound on the variance. From Algorithm 2 it is clear that Lemma 3 allows us to obtain the values of M_i for $i \in \{1, 2, \ldots, n\}$. Indeed, the MAX-oracle is not well defined when $m > n$. In order to obtain the values of M_i for $i \in \{n+1, \ldots, n'\}$, we propose the following technique.

Recall that the elements of $\mathbb{F}_q^n$ can be represented as n dimensional vectors where each element belongs to $\mathbb{F}_q$. Moreover we defined an ordering over the elements of the finite field $\mathbb{F}_q \equiv \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$ so that $\alpha_i \prec \alpha_j$ for $i < j$. Consider the lexicographic ordering of the elements (vectors) of $\mathbb{F}_q^n$. Let s_m be the $\lceil \frac{r^m}{q^{m-n}} \rceil$ th element in this ordering of $\mathbb{F}_q^n$. Define the set

$$\mathcal{S}_m = \{x \in \mathbb{F}_q^n \mid x \prec s_m\} \tag{10}$$

for all $m > n$. Now, let $A \in \mathbb{F}_q^{n \times n}$ be an $n \times n$ full rank matrix randomly and uniformly chosen from the set of all $n \times n$ rank-n matrices over $\mathbb{F}_q$, which can be generated by rejection sampling as before. Let $b \in \mathbb{F}_q^n$ be a uniform random vector. Subsequently, the MAX-Oracle for `Unconstrained MB-WISH` solves the following optimization problem for $m > n$:

$$\max_{y \in \mathcal{S}_m} w(Ay + b).$$

In order to analyze the statistical properties of this oracle, define the random set $T_{A,b,m} \equiv \{Ay + b \mid y \in \mathcal{S}_m\}$. Again, for each configuration $\sigma \in \mathbb{F}_q^n$, associate an indicator random variable Z_σ denoting $\sigma \in T_{A,b,m}$.

Lemma 4. *For each configuration $\sigma \in \mathbb{F}_q^n$, we must have* $\left(\frac{r}{q}\right)^m - \frac{1}{q^n} \leq \Pr(Z_\sigma = 1) \leq \left(\frac{r}{q}\right)^m$ *and moreover for any two distinct configurations $\sigma_1, \sigma_2 \in \mathbb{F}_q^n$,* $\Pr(Z_{\sigma_1} = 1 \wedge Z_{\sigma_2} = 1) \leq (\Pr(Z_\sigma = 1))^2$.

Given the two lemmas, the remainder of the proof of Theorem 2 follows that of Theorem 1 straightforwardly.

5 EXPERIMENTAL RESULTS

All the experiments were performed in a shared parallel computing environment that is equipped with 50 compute nodes with 28 cores Xeon E5-2680 v4 2.40GHz processors with 128GB RAM. Further experiments on estimating the TV distance is reported in Appendix B.

Experiments on simulated Potts model (regular degree graph). We implemented our algorithm to estimate the partition function of Potts Model. Recall that the partition function of the Potts model on a graph $G = (V, E)$ is given in Eq. (3). First of all, we computed partition functions for small graphs where a brute-force algorithm can also be used to compute the ground truth function values. For our simulation, we have randomly generated the graph G with number of nodes $n \equiv |V|$ varying in $4, 5, 6, 7, 8, 9$, and corresponding regular degree $d = 2, 2, 4, 4, 4, 4$, using a python library `networkx`. We took the number of states of the Potts model $q = 5$, the external force H and the spin-coupling J to be 0.1 and then varied the values of ζ. The partition functions for different cases are calculated using both brute force and our algorithm (`MB-WISH`). We have used a python module `constraint` to handle the constrained optimization for MAX-oracle. The obtained approximation factors for different ζ are listed in Table 1. The worst approximation factor observed in all these trials is 5.442. Note that the theoretical guarantee on the approximation ratio for this setting obtained from Theorem 1 is 6.25. This experiment shows that, for small graphs the partition functions computed by `MB-WISH` are good approximations to the actual values.

For graphs with larger number of vertices, it is not possible to compute the ground truth partition function of Potts Model by brute force. Therefore, we compare the partition function computed by `Unconstrained MB-WISH` ($\hat{Z}$) with two standard techniques: Belief propagation (BP) (Koller and Friedman, 2009) and Markov-Chain-Monte-Carlo (MCMC) (Jerrum and Sinclair, 1996). It is known that BP provides exact result when the underlying graph is cycle-free (Koller and Friedman, 2009). To implement this we use the PGMPY library in `python` (PGM,). For MCMC, we employ the popular Metropolis-Hastings (MH) algorithm (Kroese et al., 2011) to sample random points from Ω, where we evaluate the function $w : \Omega \rightarrow \mathbb{R}$ and take a scaled-sum to estimate the discrete integration problem. We have calculated the average of the partition function over 10 different trials of the MH algorithm, and each trial was given the same time as that of `Unconstrained MB-WISH`.

Again, for our simulation, we have randomly generated the graph G with number of nodes $n \equiv |V|$ vary-

ζ	$n = 4, d = 2$	$n = 5, d = 2$	$n = 6, d = 4$	$n = 7, d = 4$	$n = 8, d = 4$	$n = 9, d = 4$
0	0.976	1.220	0.610	1.907	0.953	1.192
5	0.580	0.708	1.639	0.755	0.630	0.599
10	0.7470	1.191	3.271	0.989	1.875	1.25
15	1.430	1.036	1.013	1.224	1.399	1.692
20	1.032	1.590	1.141	1.173	1.365	1.491
25	0.839	1.118	1.339	1.035	1.429	1.326
30	0.510	4.0562	2.226	1.060	0.690	2.122
35	1.073	**5.442**	0.489	2.871	1.639	1.263
40	1.210	2.434	0.980	0.582	0.666	0.969
45	1.127	4.640	2.348	1.336	0.3673	1.341
50	1.152	1.025	2.511	3.4307	1.1522	2.636

Table 1: The ratio of the partition function calculated by `MB-WISH` ($r = 2$) and the actual value calculated by brute force: $\frac{\hat{Z}}{Z}$.

n	$\zeta = 1$			$\zeta = 2$			$\zeta = 5$		
	MB-WISH	BP	MCMC	MB-WISH	BP	MCMC	MB-WISH	BP	MCMC
10	15.16	15.51	12.60	14.35	14.98	12.06	13.07	13.56	10.65
15	23.10	23.27	20.51	22.35	22.47	19.70	19.95	20.35	17.59
20	31.04	31.03	28.69	29.98	29.96	27.62	26.93	27.13	24.80
25	38.28	38.79	36.63	37.41	37.45	35.29	33.41	33.92	31.76
30	46.23	46.55	44.49	44.51	44.94	42.89	40.82	40.705	38.65
40	61.88	62.06	59.75	59.55	59.92	57.61	54.96	54.27	51.96
50	77.31	77.58	75.28	74.69	74.90	72.59	68.62	67.84	65.54

Table 2: Log-partition function computed by `unconstrained MB-WISH`, Belief Propagation (BP) and Markov Chain Monte Carlo (MCMC) respectively for the cases of $\zeta = 1, 2$ and 5.

ing in $10, \dots, 50$, and with regular degree $d = 4$ using a python library `networkx`. We took the number of states of the Potts model $q = 5$, the external force H and the spin-coupling to be 0.1 and then varied the values of ζ. In our experiments each optimization instances are run with a timeout of $10, 15, 20, 20, 25$ minutes for $n = 20, 25, 30, 40, 50$ respectively (we let the $n = 10$ case run without a time constraint). The results are summarized in Table 2. It can be observed that the partition functions computed with MCMC deviate somewhat from that computed with belief propagation, whereas `MB-WISH` gives values closer to the belief propagation results.

Since for cycle-free graphs, BP can provide exact result, it gives an opportunity to compare `MB-WISH` with the single-bin version (i.e., Ermon et al.'s original algorithm) for moderate values of n and q. We perform the next experiment on a path-graph, which is an undirected graph where there are exactly two nodes of degree 1 and every other node has degree 2. We perform the experiment with the number of nodes $n \equiv |V|$ varying in $20, \dots, 50$ on a path-graph such that the number of states $q = 31$ and the external parameters $J = 0.1$, $H = 0.5$ and $\zeta = -5$. For two different values of r, respectively 1 (single-bin) and 15 (multi-bin) we compute the estimates of the partition function. We have plotted the ratio of the estimates with the corresponding ones computed by BP (which is exact), in Figure 2. It is clear from the figure and the table that the `Unconstrained MB-WISH` performs much better than its single-bin counterpart. The timeout for each call to the oracle is chosen to be $n/10$ where n is the number of nodes in the graph.

Real-world constraint satisfaction problem (CSPs). Many instances of real-world graphical models are available in `http://www.cs.huji.ac.il/project/PASCAL/showExample.php`. Notably, some of them (e.g., image alignment, protein folding) are defined on non-Boolean domains, which justify the use of `MB-WISH`. We have computed the partition functions for several of them.

The dataset `Network.uai` is a Markov network with 120 nodes each having a binary value. A configuration here is a binary sequence of length 120. To calculate the partition function, we need to find the sum of weights for 2^{120} different configurations. In order to use `Unconstrained MB-WISH`, we view each configura-

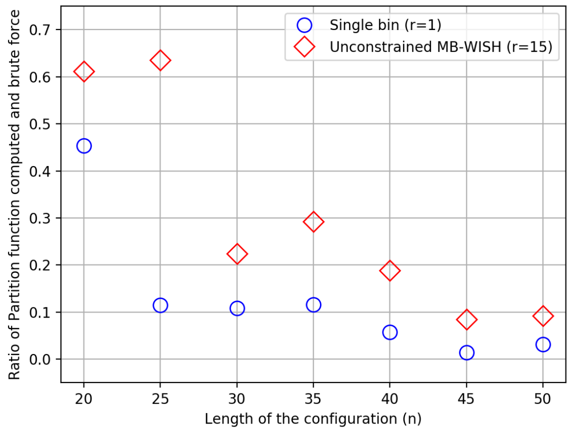

Figure 2: Comparison of approximation ratios obtained by using `Unconstrained MB-WISH` (red) and single-bin (Ermon et al.'s method) (blue). A ratio closer to 1 is better.

tion as a 16-ary string of length 30. Our results for the log-partition came out to be 156.00 with one hour time out for each call to the MAX-oracle. The benchmark for the log-partition function is provided to be 163.204.

The `Object detection` dataset comprised of 60 nodes each having a 11-ary value and by `Unconstrained MB-WISH` we found the log-partition function to be −38.9334. The `CSP` dataset is a Markov network with 30 node having a ternary value: we found the log partition function to be −39.9933. For these datasets there were no baselines available for comparison. The purpose of these experiments were to establish the scalability of `MB-WISH`.

6 CONCLUSION

Large scale counting problems (or discrete integrations of nonnegative weight functions) are often computationally intractable, but come up frequently in variety of inference tasks, most prominently as evaluations of partition functions. In this paper we extend a recent technique of hashing and optimization due to Ermon et al. for discrete integration over hypercube $\{0,1\}^n$ to that over hypergrids $\{0,1,\dots,q-1\}^n$. The trivial generalization results in an approximation factor that rapidly becomes worse as q increases. We remedy the situation by providing constant factor approximation algorithms for all q.

The main drawback of this approach of discrete integration is the delegation of a hard combinatorial optimization to an oracle. In this line of work, an open problem is to come up with hash functions that maintain the essential properties (such as pairwise independence), but make the oracle optimization amenable. While in general this is not possible, for certain classes of weight functions this may be a plausible task and requires further exploration.

Acknowledgements: This research is supported in parts by NSF awards CCF 1642658, CCF 1618512, CCF 1909046 and CCF 1934846.

References

PGMPY documentation. `http://pgmpy.org/`. Accessed: 2018-06-28.

Achlioptas, D. and Jiang, P. (2015). Stochastic integration via error-correcting codes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 22–31.

Birnbaum, E. and Lozinskii, E. L. (1999). The good old davis-putnam procedure helps counting models. *Journal of Artificial Intelligence Research*, 10:457–477.

Chakraborty, S., Fremont, D. J., Meel, K. S., Seshia, S. A., and Vardi, M. Y. (2014). Distribution-aware sampling and weighted model counting for SAT. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1722–1730.

Chakraborty, S., Meel, K. S., and Vardi, M. Y. (2016). Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3569–3576.

Devroye, L. and Lugosi, G. (2012). *Combinatorial methods in density estimation*. Springer Science & Business Media.

Ermon, S., Gomes, C., Sabharwal, A., and Selman, B. (2013a). Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 334–342.

Ermon, S., Gomes, C., Sabharwal, A., and Selman, B. (2014). Low-density parity constraints for hashing-based discrete integration. In *International Conference on Machine Learning*, pages 271–279.

Ermon, S., Gomes, C. P., Sabharwal, A., and Selman, B. (2013b). Optimization with parity constraints: From binary codes to discrete integration. In *Uncertainty in Artificial Intelligence*, page 202.

Goldberg, L. A. and Jerrum, M. (2012). Approximating the partition function of the ferromagnetic potts model. *Journal of the ACM (JACM)*, 59(5):25.

Gomes, C. P., Sabharwal, A., and Selman, B. (2006). Model counting: A new strategy for obtaining good bounds. In *AAAI*, pages 54–61.

Gomes, C. P., van Hoeve, W. J., Sabharwal, A., and Selman, B. (2007). Counting csp solutions using generalized xor constraints. In *AAAI*, pages 204–209.

Jerrum, M. and Sinclair, A. (1996). The Markov chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520.

Jerrum, M., Sinclair, A., and Vigoda, E. (2004). A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4):671–697.

Jr., R. J. B. and Pehoushek, J. D. (2000). Counting models using connected components. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA.*, pages 157–162.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Kroese, D., Taimre, T., and Botev, Z. (2011). *Handbook of Monte Carlo Methods*. John Willey & Sons Inc., Hoboken, New Jersey.

Pesant, G. (2005). Counting solutions of csps: A structural approach. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 260–265.

Potts, R. B. (1952). Some generalized order-disorder transformations. In *Mathematical proceedings of the cambridge philosophical society*, volume 48, pages 106–109. Cambridge University Press.

Sason, I. and Verdú, S. (2016). f-divergence inequalities. *IEEE Transactions on Information Theory*, 62(11):5973–6006.

Sipser, M. (1983). A complexity theoretic approach to randomness. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 330–335. ACM.

Stinson, D. R. (1996). On the connections between universal hashing, combinatorial designs and error-correcting codes. *Congressus Numerantium*, pages 7–28.

Stockmeyer, L. (1985). On approximation algorithms for# p. *SIAM Journal on Computing*, 14(4):849–861.

Thurley, M. (2011). An approximation algorithm for# k-sat. *arXiv preprint arXiv:1107.2001*.

Valiant, L. G. (1979). The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201.

Valiant, L. G. and Vazirani, V. V. (1986). NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93.

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

Wei, W. and Selman, B. (2005). A new approach to model counting. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 324–339. Springer.