Stochastic Integration via Error-Correcting Codes

Dimitris Achlioptas
Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064, USA

Pei Jiang
Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064, USA

Abstract

We consider the task of summing a non-negative function f over a discrete set Ω, e.g., to compute the partition function of a graphical model. Ermon et al. have shown that in a probabilistic approximate sense summation can be reduced to maximizing f over random subsets of Ω defined by parity (XOR) constraints. Unfortunately, XORs with many variables are computationally intractable, while XORs with few variables have poor statistical performance. We introduce two ideas to address this problem, both motivated by the theory of error-correcting codes. The first is to maximize f over explicitly generated random affine subspaces of Ω, which is equivalent to *unconstrained* maximization of f over an exponentially smaller domain. The second idea, closer in spirit to the original approach, is to use systems of linear equations defining Low Density Parity Check (LDPC) error-correcting codes. Even though the equations in such systems only contain $O(1)$ variables each, their sets of solutions (codewords) have excellent statistical properties. By combining these ideas we achieve dramatic speedup over the original approach and levels of accuracy that were completely unattainable.

1 INTRODUCTION

The partition function of a graphical model with unnormalized probability function f over a domain Ω is the integral (sum) of f over Ω. The partition function is a central object of Bayesian statistics. While some inference tasks, such as MAP or MLE, can be completed without it, knowledge of (an approximation of) the partition function is necessary for marginalization, prediction, sampling, and model comparison, as a proper distribution is required. In general, partition function estimation is intractable [1] and, in practice, becomes problematic rapidly as $|\Omega|$ increases. To overcome this problem approximation schemes, such as MCMC [7], or variational methods [5] are commonly used. However, variational methods, in general, do not provide accuracy certificates/guarantees, while the mixing time of MCMC is similarly unpenetratable in most applications. In recent years, Ermon, Gomes, Sabharwal, and Selman have pioneered an alternative approach [14, 13, 15]. The general idea is to reduce the counting problem into a collection of random optimization problems, the final estimate being a statistic over the optima found. Each random problem amounts to maximizing f over a random set $R \subseteq \Omega$, defined via a random system of parity constraints. Realizing this idea, the WISH algorithm [14] is shown to yield a constant-factor approximation for the partition function *given access to an optimization oracle*. Importantly, the approximation guarantee requires the induced optimization problems to be solved to optimality, an assumption that clearly does not hold in general. Nevertheless, empirically, the WISH algorithm achieves remarkable accuracy compared to other established algorithms [14, 13, 15], leveraging the practical advancements in optimization software.

The idea of adding parity constraints originates in the work of Sipser [16] as a reduction technique. Most famously, it was used by Valiant and Vazirani [19] to reduce SAT to Unique SAT. A variant of the technique plays an important role in the proof of Toda's theorem [18]. The idea has since been applied to various counting problems including $\#$SAT [11], $\#k$-SAT [17], and$\#$CSP [12]. WISH [14] can be regarded as a natural generalization to weighted CSPs (or, equivalently, Markov Random Fields (MRFs)).

To provably approximate the partition function of an MRF with n variables, the parity constraints added must each have $n/2$ variables on average. Unfortunately, the addition of such long constraints makes the MAP problem dramatically harder since each constraint (i) amounts to a clique involving half the variables of the MRF, and (ii) collapses the probability function whenever violated. In practice, this additional hardness can cause a MAP solver to submit dramatically suboptimal solutions under any reasonable time constraint, impairing the accuracy of estimation.

In all prior works the addition of random parity constraints is framed as *hashing* and the statistical properties of the resulting subsets of the domain is discussed in terms of independence properties of the corresponding families of hash functions. We break with this paradigm by taking a step back and asking: "how can we define subsets of the domain so that they are *computation-friendly* while having *good statistical properties*?" We answer the question twice, the two answers corresponding to two different notions of "friendliness" under the same notion of "goodness".

Regarding goodness we will see that the key statistical property is pairwise *negative correlation* of membership, i.e., that for any two distinct $\sigma, \sigma' \in \Omega$, it should be that $\Pr[\text{Both } \sigma, \sigma' \in R] \leq \Pr[\sigma \in R] \Pr[\sigma' \in R]$. (Long parity constraints achieve this with equality.) Equivalently, conditioning on $\sigma \in R$ should not make any $\sigma' \neq \sigma$ more likely to be in R (but can make it less). Visualizing this as $\sigma \in R$ exerting a repulsive force suggests an error-correcting code. Indeed, any linear error-correcting code $\mathcal{C} \subseteq \Omega = \{0,1\}^n$ with 2^{n-d} elements can be specified as $\mathcal{C} = \{\sigma \in \Omega : A\sigma = b\}$, where $A \in \{0,1\}^{d \times n}$ is any rank d matrix and operations are over GF(2), i.e., as the set of solutions to parity constraints.

Equipped with this idea, our first notion of computation-friendliness can be seen as "dimensionality reduction". That is, instead of operating over $\{0,1\}^n$ and maximizing f over R by setting $f(\sigma) = 0$ for $\sigma \in \Omega \setminus R$, we can operate over R directly by taking $G \in \{0,1\}^{n \times d}$ to be a *generator* of the subspace $A\sigma = b$ and maximizing $f(Gx + v)$ over $x \in \{0,1\}^d$. We thus get an *un*constrained optimization problem over a domain of size 2^d instead of 2^n. For any optimizer treating f as a black box, as is typical in MAP estimation, this makes optimizing f dramatically easier.

Our second notion of computation-friendliness can be seen as endowing $\Omega \setminus R$ with a "gradient" (pointing towards R), so that satisfying $A\sigma = b$ does not impose significant computational burden. Again drawing insights from the theory of error-correcting codes, the idea is to desire the number of violated equations of $A\sigma = b$ to be a function that has few local minima that are not global minima, i.e., not codewords, thus making its global minima easily accessible by some naive local method such as gradient descent. In other words, to make the optimizer's life easy, we would like $\mathcal{C} = R$ to be an "easily decodable" code. This is precisely what we will achieve by taking the random sets R to correspond to the codewords of LDPC codes constructed by the Progressive-Edge-Growth construction [3].

Finally we note that independently of how the optimization problems are constructed, the *number* of instances that need to be solved can be reduced significantly in practice by using branch-and-bound. Combined with the two ideas mentioned above, this gives a dramatic speedup over WISH and entirely new levels of accuracy.

2 BACKGROUND

For the benefit of clarity, as in previous works, we will restrict our exposition to $\Omega = \{0,1\}^n$ and only approximate the partition function, Z, within a fixed constant factor, e.g., 32 (recall that, typically, $Z \sim \exp(n)$). All ideas presented generalize readily to $\Omega = D^n$ for any finite D.

2.1 BINARY MARKOV RANDOM FIELD

Given $\Omega = \{0,1\}^n$ and a collection of non-negative functions, $\mathcal{F} = \{\psi_\alpha\}$, defined on subcubes of Ω, let

$$f(\sigma) = \prod_{\psi_\alpha \in \mathcal{F}} \psi_\alpha(\{\sigma\}_\alpha) \ ,$$

where $\{\sigma\}_\alpha$ is the subset of variables entailed by ψ_α. The *partition function* is the sum of f over all configurations:

$$Z = \sum_{\sigma \in \Omega} f(\sigma) \ .$$

2.2 ESTIMATION BY STRATIFICATION

We start by briskly revisiting (and, to some extent, reformulating) the groundbreaking work of Ermon et al. [14] connecting partition function estimation to optimization.

The first key idea is to stratify f over Ω into quantiles and estimate Z by bounding from above and below the contribution of each quantile. Specifically, and w.l.o.g., assume that the configurations are sorted in descending order according to f, i.e., $f(\sigma_1) \geq f(\sigma_2) \geq \ldots \geq f(\sigma_{2^n})$. Let $b_i = f(\sigma_{2^i})$. Now, define the *lower sum* as

$$L := b_0 + \sum_{i=0}^{n-1} b_{i+1} 2^i$$

and the *upper sum* as

$$U := b_0 + \sum_{i=0}^{n-1} b_i 2^i \ .$$

Trivially, $L \leq Z \leq U$. Moreover,

$$\begin{aligned} 2L &= b_0 + \left(b_0 + \sum_{i=0}^{n-1} b_{i+1} 2^{i+1} \right) \\ &= b_0 + \sum_{i=0}^{n} b_i 2^i \geq U \ . \end{aligned}$$

Hence, if we compute $b_0, b_1, \ldots, b_n$, taking any $\widehat{Z} \in [L, U]$ yields a 2-approximation of Z.

More generally, if for some integer $c \geq 0$ and all $i \in [n]$ an estimate $\hat{b}_i \in [b_{i+c}, b_{i-c}]$ is available, then letting $\widehat{U}$ and $\widehat{L}$ be the counterparts of U and L with b_i replaced by $\hat{b}_i$ we see that $\widehat{L} \leq L \leq Z \leq U \leq \widehat{U}$ and $\widehat{U}/\widehat{L} \leq 2^{2c+1}$. Thus, any $\widehat{Z} \in [\widehat{L}, \widehat{U}]$ is a 2^{2c+1}-approximation of Z, e.g., yielding a 32-approximation for $c = 2$.

2.3 STRATIFICATION BY THINNING

The second key idea is to estimate each $b_i = f(\sigma_{2^i})$ as the maximum of f over a random set $R_i \subseteq \Omega$ of (expected) size 2^{n-i}. As mentioned, the essential requirement for this approach to work is *pairwise negative correlation* of membership in R_i. Including each element of Ω in R_i independently with probability 2^{-i} achieves this trivially but at the cost of an exponentially large, and thus inoperable, representation of R_i. The foundation of this entire line of research has been that it is possible to achieve *pairwise* independence for membership in R in compact form via hashing. We introduce a somewhat more general, and ultimately more fruitful, point of view reflected in our definition of Thinning Sets below.

Thinning Sets. *A random variable R_i taking values in 2^{Ω} is an i-*thinner *if*

- $\forall \sigma$, $\Pr[\sigma \in R_i] = 2^{-i}$ *(Uniform)*
- $\forall \sigma \neq \sigma'$, $\Pr[\sigma \in R_i \wedge \sigma' \in R_i] \leq 2^{-2i}$ *(Universal)*

Given an i-thinner R_i and a solver capable of maximizing f over R_i, estimating b_i is entirely straightforward. Theorem 1 below is identical to the main result of [14], except for thinning sets replacing hash functions (for completeness we prove Theorem 1 in Section 5.)

Theorem 1 ([14])**.** *Let R_i be any i-thinner random variable. Let $\{m_j\}_{j=1}^{t}$ be i.i.d. random variables distributed as $m_j = \max_{\sigma \in R_i} f(\sigma)$. If $M = \text{median}(m_1, \ldots, m_t)$, then for every $c \geq 2$,*

$$\Pr[b_{i+c} \leq M \leq b_{i-c}] \geq 1 - 2\exp\left(-\frac{t}{2}(1 - 2^{-c+1})^2\right) .$$

One way to create an i-thinner is to take the solutions of a random system of linear equations over GF(2), i.e., modulo 2. Let $A \sim \text{Ber}(m \times n)$ denote that A is an $m \times n$ random matrix whose entries are independent random variables with $\Pr[a_{ij} = 1] = \Pr[a_{ij} = 0] = 1/2$, for all i, j.

Random Linear Code. *The random set*

$$R_i = \{\sigma \in \{0,1\}^n : A\sigma = b\} \qquad (1)$$

is an i-thinner if $A \sim \text{Ber}(i \times n)$ and $b \sim \text{Ber}(i \times 1)$.

In coding theory the set R_i in (1) is known as a random linear code, while the distribution $A \sim \text{Ber}(i \times n)$ is known as the Shannon ensemble. Note that the rows of A have, on average, $n/2$ non-zero entries. We will refer to it as the *dense parity* ensemble, to distinguish it from other distributions on $\{0,1\}^{i \times n}$ which we will encounter shortly. By Theorem 1, we can thus estimate b_i given an oracle $\mathcal{O}$ for

$$\max_{\substack{\sigma \in \{0,1\}^n \\ A\sigma = b}} f(\sigma) \ . \qquad (2)$$

The idea of adding long random parity constraints to achieve unweighted counting, e.g., to count the number of satisfying assignments of a CNF formula goes back to [11]. Ermon et al. in [14], after i-thinning Ω in the manner above, solved the optimization problem (2) with ToulBar2 [2], dedicated software for MAP estimation in graphical models (the parity constraints added as factors to f evaluating to 0 when violated). In later work [13], the authors translated (2) to an Integer Linear Program, thus bringing to bear CPLEX, a powerful commercial optimization software. Finally, for reasons to be discussed shortly, in [15], the dense parity ensemble was replaced by the *sparse parity ensemble* wherein the entries of A are i.i.d. Bernoulli random variables where $\Pr[a_{ij} = 1] = p < 1/2$.

3 OUR CONTRIBUTION

3.1 RANDOM AFFINE MAPS

Instead of starting with $\Omega = \{0,1\}^n$ and restricting it via i random parity equations to a subset R_i of (expected) size 2^{n-i}, we will *start* with $\{0,1\}^{n-i}$ and generate R_i as the image of $\{0,1\}^{n-i}$ under a random affine transformation $g : \{0,1\}^{n-i} \to \{0,1\}^n$, where $g(x) = Ax + b$. Thus, instead of solving the constrained optimization problem

$$\max_{\substack{\sigma \in \{0,1\}^n \\ A\sigma = b}} f(\sigma) \ ,$$

we will solve the *unconstrained* optimization problem

$$\max_{x \in \{0,1\}^{n-i}} (f \circ g)(x) \ ,$$

over the *exponentially* smaller set $\{0,1\}^{n-i}$. The benefit of such dimensionality reduction increases with i, i.e., smaller R_i, in contrast to thinning by parity constraints which typically has worsening behavior as i is increased.

3.2 LOW DENSITY PARITY CHECK CODES

In certain settings, such as when performing "light" thinning or when the function f can be optimized better than black box, operating directly on the restriction of Ω induced by parity constraints is preferable to operating through a random affine map. In these settings, instead of forming the constraint matrix A by having its entries be i.i.d. Bernoulli random variables (either sparse or dense) we will take A to be the parity check matrix of a Low Density Parity Check **(LDPC)** code. As we demonstrate experimentally, this has a stunning effect on the performance of CPLEX.

In the eyes of a solver operating on the variable representation of Ω (as opposed to a local search solver) a 3-XOR is greatly preferable to an $(n/2)$-XOR, even though both shrink the domain by half. This is because repairing a violated constraint of arity k represents a k-way choice, i.e., a branching factor of k. This motivated the introduction of sparse i.i.d. Bernoulli parity check matrices in [11] and later in [15]. While a step in the right direction, this does not go far enough. To cover the remaining distance, we exploit insights from the modern theory of LDPC codes.

Imagine a code $\mathcal{C} = \{\sigma \in \{0,1\}^n : A\sigma = b\}$, for some fixed matrix A and vector b. Imagine further that $\sigma \in \mathcal{C}$ is transmitted along a channel that *erases* a subset of the bits of σ, so that the recipient receives $\tau \in \{0, 1, *\}^n$. Clearly, equations (checks) with no $*$ variables offer no new information regarding σ. On the other hand, equations with two or more $*$ variables are ambiguous, as they can be satisfied in multiple ways. But any equation with exactly one $*$ variable is ideal: its erased bit can be recovered unambiguously; moreover, this recovery may cause other equations that had two $*$ variables to now only have one. Such a cascade of "safe steps" will recover σ unless it encounters a *stopping set*: a non-empty set V of erased bits, such that no equation entails exactly 1 element of V. The amazing performance of LDPC codes is, to first order, due to the absence of small stopping sets. Thus, if τ does not have too many erased bits, safe steps will suffice to recover σ.

To readers familiar with satisfiability algorithms the parallel between "safe" decoding and unit-clause propagation (UCP) will be immediate. The linear equations defining code $\mathcal{C}$ can be thought of as a formula F *very carefully designed* to have the following property: if one selects a random subset of variables and assigns them random values (subject only to no empty clause being created), then for the vast majority of random choices (corresponding to the unerased bits in the communication setting) the residual formula should be solvable by UCP alone. It is not hard to imagine that adding such a formula F to a formula F' will induce little "additional hardness" to any solver capable of recognizing the presence of "safe" choices: as soon as enough variables are instantiated to get within the "radius of attraction" of a solution to F, the solver devolves to a "safe choice decoder", setting variables at a rapid pace with minimal branching. We conjecture that this is precisely what causes the stunning improvement we observe in the performance of CPLEX when switching from dense/sparse parity ensembles to LDPC codes. Unfortunately, verifying this directly is non-trivial as CPLEX is commercial software. As implicit evidence we offer the observed complete insensitivity of stochastic local search to the structure of A (we use LocalSolver [9] in our experiments).

3.3 BRANCH AND BOUND

Recall that to form our estimate $\widehat{Z}$ we multiply each $\hat{b}_i = f(\sigma_{2^i})$ by 2^i. As a result, in most cases, $\widehat{Z}$ is dominated by the contribution of a set of quantiles $I \subseteq [n]$, where $|I| \ll n$. (Indeed, in physical terms, failure of this to be true is the signature of criticality.) With this observation in mind, rather than estimating all $\{\hat{b}_i\}_{i=1}^n$ in sequence, we can save computation by starting with $S = \{\hat{b}_0, \hat{b}_n\}$ and estimating more and more quantiles until sufficient accuracy is achieved. In particular, simply enlarging S by the unestimated quantile of greatest remaining potential contribution, gives a speedup ranging from 2x to 10x in our experiments, with 7x being the most common case.

4 RANDOM AFFINE MAPS

Throughout this section let $d := n - i$, where $i \in [n]$. Let $A \in \{0,1\}^{n \times d}$ be a matrix of rank d and let $b \in \{0,1\}^n$. If $g(x) = Ax + b$, then the image of $\{0,1\}^d$ under g is an affine subspace of $\{0,1\}^n$ containing 2^d distinct elements (since A has full rank). Let $\{0,1\}_d^{n \times d}$ denote the set of all full rank, i.e., rank d, matrices in $\{0,1\}^{n \times d}$.

Theorem 2. *Let A be uniform over $\{0,1\}_d^{n \times d}$ and let v be uniform over $\{0,1\}^n$. The image of $\{0,1\}^d$ under $x \mapsto Ax + v$ is an $(n-d)$-thinner.*

Theorem 1 immediately implies the following.

Corollary 1. *Let A be uniform over $\{0,1\}_d^{n \times d}$ and let v be uniform over $\{0,1\}^n$. Let $\{m_j\}_{j=1}^t$ be i.i.d. random variables distributed as*

$$\max_{x \in \{0,1\}^d} f(Ax + v) \ .$$

If $M = \mathrm{median}(m_1, \ldots, m_t)$, then for every $c \geq 2$,

$$\Pr[b_{i+c} \leq M \leq b_{i-c}] \geq 1 - 2\exp\left(-\frac{t}{2}(1 - 2^{-c+1})^2\right) \ .$$

In other words, replacing long parity constraints with random affine maps, retains all statistical guarantees while giving rise to optimization instances over $\{0,1\}^{n-i}$ instead of $\{0,1\}^n$. As in the estimation of the partition function, i ranges from 1 to n, at some point it becomes much more efficient to operate on $f \circ g$ in the reduced domain that to operate on f on Ω. Moreover, because $|R_i| = 2^{n-i}$ deterministically, rather than in expectation, the variance of each estimate m_i is smaller than for a random linear code.

To sample uniformly from $\{0,1\}_d^{n \times d}$ it is convenient and efficient to employ rejection sampling: trivially generate $A \sim \mathrm{Ber}(n \times d)$ and accept only if $\mathrm{rank}(A) = d$. Uniformity follows from the uniformity of $A \sim \mathrm{Ber}(n \times d)$ over $\{0,1\}^{n \times d}$. By Lemma 1, the number of trials needed is a geometric random variable with mean less than 4.

Lemma 1. $\left|\{0,1\}_d^{n \times d}\right| > 2^{dn-2}$.

4.1 EVALUATION

In Figure 1 we compare parity constraints vs. affine maps on the 10×10 Ising grid with $F = 0.1$ and $C = 1.0$, a noted hard problem in [14]. For the exact definition of

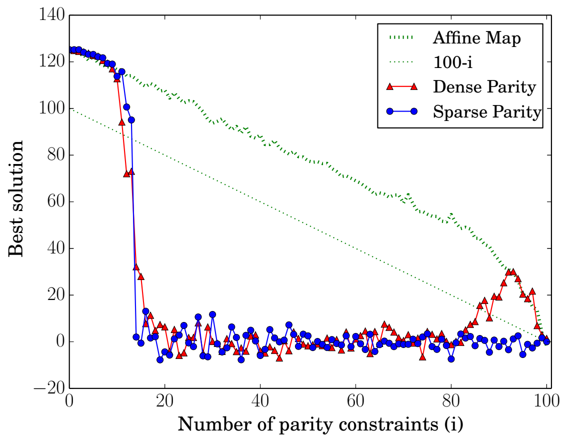

Figure 1: 10×10 Ising grid with $C = 1.0$ and $F = 0.1$. *Experiment:* for $i \in [0, 100]$ generate R_i via i random parity constraints, or as a random affine subspace of dimension $n - i$; seek $\max_{\sigma \in R_i} f(\sigma)$ for 30 seconds (see legend). *Plot:* for each i, repeat the experiment 10 times and report the binary logarithm of the median value found. We also plot the number $100 - i$ as a visual aid.

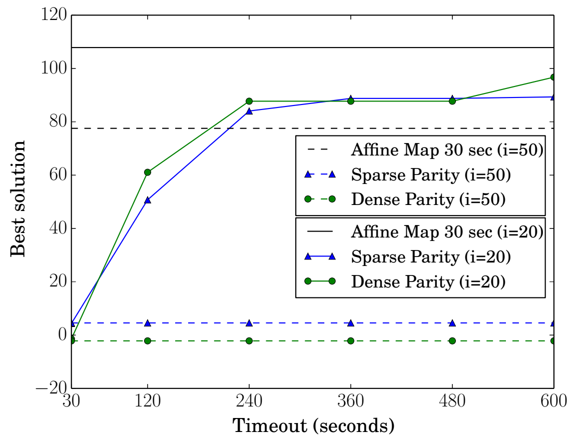

Figure 2: 10×10 Ising grid with $C = 1.0$ and $F = 0.1$. *Experiment:* for $i \in \{20, 50\}$ generate R_i via i random parity constraints; seek $\max_{\sigma \in R_i} f(\sigma)$ with a timeout of $t \in \{30, 120, 240, 360, 480, 600\}$ seconds (see legend). *Plot:* for each pair (i, t) repeat the experiment 10 times and report the binary logarithm of the values found.

f see (9) in Section 7. The parity constraints are generated from the dense parity ensemble and the sparse parity ensemble adopted in [15] and we use the ILP formulation proposed in [13], with CPLEX being the solver. To optimize $f \circ g$ for random affine maps we use LocalSolver [9], since CPLEX is an ILP solver and does not natively support affine transformations over GF(2). While not as strong as CPLEX on constrained optimization problems, LocalSolver is specialized in stochastic local search under black-box evaluation, hence a suitable choice for our setting.

As can be seen in Figure 1, when there are more than, roughly, 10 parity constraints, the performance of both random parity ensembles deteriorates rapidly, in sharp contrast to the robust performance of LocalSolver under affine maps. Note that since the y-axis is in $\log_2$-scale and each b_i contributes roughly $b_i 2^i$ to the partition function, the fact that the solutions found by LocalSolver are nearly parallel to the line $100-i$ imply that the under-performance of optimization under parity constraints is highly relevant and will have dramatic effect on the accuracy of estimation.

Moreover, as shown in Figure 2, the best solutions found under parity constraints in 10 minutes are still inferior to those found via random affine maps in 30 seconds. In particular, when there are 50 constraints, CPLEX cannot find a solution better than the initial one under either parity ensemble, suggesting that the hardness of optimization under parity constraints overwhelms the solver. Notably, the original MAP inference $\max_{\sigma \in \Omega} f(\sigma)$ can be solved to provable optimality in 0.1 second by CPLEX, highlighting that the hardness is due to the parity constraints.

4.2 PROOF OF THEOREM 2 AND LEMMA 1

We will refer to A as a *generator* matrix for the subspace, which we will represent by the pair (A, v).

Lemma 2. *If A is uniform over $\{0,1\}^{n \times d}_d$ and v is uniform over $\{0,1\}^n$, then (A, v) is uniform over all affine subspaces of dimension d. In particular, for any fixed v_0, the subspace (A, v_0) is uniform over all affine subspaces of dimension d that contain v_0.*

Proof. It suffices to prove the second statement as the independence of A and v implies the first.

We will prove that for any fixed v_0 and any affine subspace $\mathcal{A}$ of dimension d that contains v_0, the number of matrices $A \in \{0,1\}^{n \times d}_d$ such that $(A, v_0) = \mathcal{A}$ is independent of $\mathcal{A}$.

Clearly, $(A, v_0) = \mathcal{A}$ iff the columns of A are linearly independent elements of $\mathcal{A}$. For $k \in [d]$, let a_k be the k-th column of A and let $a_0 = \mathbf{0}$. Linear independence is equivalent to $a_k \notin \operatorname{span}(a_0, \ldots, a_{k-1})$ for all $k \in [d]$. Since $|\operatorname{span}(a_1, \ldots, a_k)| = 2^k$ if the first k columns are linearly independent, we see that for every $k \in [d]$ there are $2^d - 2^{k-1}$ valid choices for the k-th column. $\square$

Proof of Theorem 2. By the first part of Lemma 2, (A, v) is uniform over all affine subspaces of dimension d. Therefore, by symmetry, $\Pr[\sigma \in (A, v)]$ is the same for all $\sigma \in \{0,1\}^n$. Since (A, v) contains 2^d elements, uniformity follows. To prove universality we need to show that $\Pr[\sigma' \in (A, v) \mid \sigma \in (A, v)] \le \Pr[\sigma' \in (A, v)]$. For this we first observe that, by the second part of Lemma 2,

$$\Pr[\sigma' \in (A, v) \mid \sigma \in (A, v)] = \Pr[\sigma' \in (A, \sigma)] \ .$$

Since $\sigma \neq \sigma'$, this last probability equals the probability that $\tau = \sigma' - \sigma \neq \mathbf{0}$ belongs in $(A, 0)$. Let a_k be the k-th column of A and let A_k comprise the first k columns of A. If we generate A column by column we see that this last probability equals $1 - \prod_{k=1}^{d} \Pr[a_k \notin \operatorname{span}(A_{k-1} \cup \{\tau\})]$ which is the same for all $\tau \neq \mathbf{0}$. □

Proof of Lemma 1. If we construct A column by column then, as shown in Lemma 2, there are $2^n - 2^{k-1}$ choices for the k-th column that lead to A being full rank. Induction thus shows $\prod_{k=1}^{d}(2^n - 2^{k-1}) \geq \frac{1}{4}\left(2^{dn} + 2\right)$. □

5 THINNING AS ERROR-CORRECTION

Let us start by deriving the statistical desiderata of thinning sets from first principles. This will illuminate the suitability of error-correcting codes for thinning and offer insight. Recall that our goal is to estimate $b_i = f(\sigma_{2^i})$, for $i \in [n]$. We start by observing that for this it suffices to construct a random variable m_i such that for some (small) integer c and any $\epsilon > 0$,

$$\Pr[m_i \leq b_{i-c}] \quad \geq \quad 1/2 + \epsilon \tag{3}$$

$$\Pr[m_i \geq b_{i+c}] \quad \geq \quad 1/2 + \epsilon \ . \tag{4}$$

This is because if we take $\hat{b}_i$ to be the median of t independent realizations of m_i, by Hoeffding's inequality,

$$\Pr\left[b_{i+c} \leq \widehat{b}_i \leq b_{i-c}\right] \geq 1 - 2\exp(-2\epsilon^2 t) \ .$$

Thus, in order for $\Pr[m_i \in [b_{i+c}, b_{i-c}]] = 1 - \exp(-\Theta(s))$ it suffices to take $O(s/\epsilon^2)$ samples.

Achieving (3) is trivial. Let $\Omega_j = \{\sigma_1, \sigma_2, \ldots, \sigma_{2^j}\}$. If $R_i \subseteq \Omega$ is *any* random set such that $\Pr[\sigma \in R_i] = 2^{-i}$ for all $\sigma \in \Omega$ and $m_i = \max_{\sigma \in R_i} f(\sigma)$, then

$$\Pr[m_i > b_{i-c}] < |\Omega_{i-c}| 2^{-i} = \ 2^{-c}. \tag{5}$$

In other words, for m_i to be unlikely to be "too big" it suffices for R_i to have the right (expected) size, without any requirement of its geometry beyond uniformity. For example, R_i could even be a random subcube of Ω, an extremely computation-friendly constraint: pick i variables at random and assign them random values.

Achieving (4) is far more subtle. This is because in order for $\Pr[m_i \geq b_{i+c}]$ to not vanish the random variable $X_i = |\Omega_{i+c} \cap R_i|$ must be well-behaved. In particular, observe that $\mathbb{E}X_i = 2^c$ and we aim for c to be small, e.g., $c = 2$, so the expectation of X_i is modest. If X_i realizes its modest expectation via a lottery phenomenon, i.e., typically $X_i = 0$ but rarely X_i is very large we are in trouble. To control for this possibility we use the Paley-Zygmund inequality asserting that if X is any non-negative integer random variable, then $\Pr[X > 0] \geq (\mathbb{E}X)^2/\mathbb{E}X^2$. Taking R_i to be a random cube is thus exposed as a bad idea: if Ω_{i+c} is also a cube, their potential alignment implies $\mathbb{E}X_i^2 \gg (\mathbb{E}X_i)^2$.

To minimize $\mathbb{E}X_i^2$ we would like to find random sets R_i that behave like "mists", minimizing the probability of having an atypically large intersection with any fixed set. Error-correcting codes are ideal for this, with linear codes particularly so, as they are specified via linear equations. Motivated by these considerations, let

$$R_i = \{\sigma \in \{0,1\}^n : A\sigma = b\} \ ,$$

where the vector $b \in \{0,1\}^i$ is uniformly random, while $A \in \{0,1\}^{i \times n}$ is *arbitrary* (even deterministic), for now.

It is easy to see that the uniformity of b over $\{0,1\}^i$ alone suffices to make R_i uniform over $\{0,1\}^n$, i.e., for all σ,

$$\Pr[\sigma \in R_i] = 2^{-i} \ . \tag{6}$$

At the same time, for *any* $S \subseteq \Omega$, if $X_i = |S \cap R_i|$, then

$$\begin{aligned} \mathbb{E}X_i^2 &= \mathbb{E}\left(\sum_{\sigma \in S} \mathbf{1}_{\sigma \in R_i}\right)^2 \\ &= \sum_{\sigma, \sigma' \in S} \mathbb{E}(\mathbf{1}_{\sigma \in R_i} \mathbf{1}_{\sigma' \in R_i}) \\ &= \mathbb{E}X_i + \sum_{\substack{\sigma, \sigma' \in S \\ \sigma \neq \sigma'}} \Pr[A\sigma = b = A\sigma'] \ . \end{aligned}$$

To deal with the sum above note that, trivially,

$$\Pr[A\sigma = b = A\sigma'] = \Pr[A\sigma = b \wedge A(\sigma - \sigma') = \mathbf{0}] \ .$$

Fix any distinct pair σ, σ'. Choosing A first (to determine if $A(\sigma - \sigma') = \mathbf{0}$) and then choosing b (to determine if $A\sigma = b$) we see that $\Pr[A\sigma = b = A\sigma'] = 2^{-i} \Pr[A(\sigma - \sigma') = \mathbf{0}]$. Therefore, without *any* assumptions on either the set S or the distribution of A, we can conclude that

$$\mathbb{E}X_i^2 = \mathbb{E}X_i + 2^{-i} \sum_{\substack{\sigma, \sigma' \in S \\ \sigma \neq \sigma'}} \Pr[A(\sigma - \sigma') = \mathbf{0}] \ . \tag{7}$$

To move beyond this point we must make some assumptions about (the distribution of) A. One such assumption, of course, would be that for all $\sigma - \sigma' = \tau \neq \mathbf{0}$,

$$\Pr[A\tau = \mathbf{0}] \leq 2^{-i} \ . \tag{8}$$

It is not hard to see that if (8) holds then:

(a) R_i is an i-thinner.

(b) $\mathbb{E}X_i^2 \leq \mathbb{E}X_i + (\mathbb{E}X_i)^2$. Thus, by the Payley-Zigmund inequality, $\Pr[m_i \geq b_{i+c}] > 1 - 2^{-c}$.

(c) Taking $c = 2$ and recalling (5) we see that (3), (4) are satisfied with $\epsilon = 1/4$ (and we have proven Theorem 1).

The above viewpoint exposes how extraordinarily strict is the requirement of universality: it asks that an element $\tau \in \Omega$ that has a single 1 should be no more likely to solve $A\tau = \mathbf{0}$ than $\tau = \mathbf{1}$. Clearly, this can only be satisfied if the rows of A have very large expected mass. In [15] the universality requirement was dropped and the case where the entries of A are i.i.d. Bernoulli taking the value 1 with probability $p \leq 1/2$ was analyzed. The bound derived for the contribution of each $\sigma \in S$ to the sum in (7) under this scheme is dominated by its pairing with σ' forming a Hamming ball centered at σ. Note, though, that if R_i is an error-correcting code and $\sigma \in R_i$, then it is extremely unlikely that *any* σ' near σ will also be in R_i. Indeed, the most basic metric of the quality of an error-correcting code is its distance, i.e, the minimum distance of any two codewords. This "self-repulsive" property of error-correcting codes is our key insight in regards to their statistical properties.

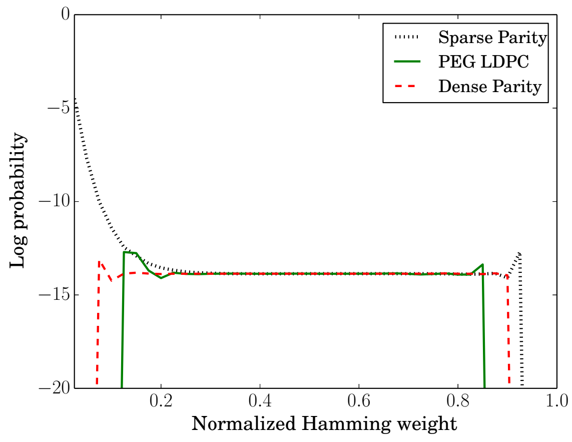

Figure 3: Empirical $\log P(w)$ for $w \in [1, 40]$ of dense parity ensemble (average 20 vars/equation), sparse parity ensemble (avg 8 vars/eq), and PEG LDPC (avg 8 vars/eq.)

5.1 LOW DENSITY PARITY CHECK CODES

In the basic LDPC construction, the (random) system of linear equations is represented as a bipartite graph with variables on the left and equations (checks) on the right, with adjacency denoting entailment, i.e., that the variable participates in the equation. To create a code with n variables, i.e., with codewords in $\{0, 1\}^n$, one first specifies the numbers $\{\lambda_j\}$ and $\{\phi_j\}$ of variables and equations, respectively, of each degree j. In the simplest case, $\lambda_j = \phi_k = 0$ for all but one value of j and k, respectively, i.e., the graph is (bi-)regular. In general, with the degree sequences thus fixed, a random bipartite graph is chosen uniformly at random subject to the degree constraints. This uniformity implies that $\Pr[A\tau = \mathbf{0}]$ depends only on the weight of τ, i.e., its number of 1s, for every $\tau \in \{0, 1\}^n$.

An even better construction of LDPC codes than the above is the Progressive Edge Growth (PEG) construction [3]. Its main feature, relative to the standard LDPC construction, is that it tries to maximize the length of the shortest cycle (girth) of the resulting bipartite graph, as short cycles contribute significantly to the formation of small stopping sets. Motivated by these considerations we replaced the dense and sparse parity ensembles with PEG constructed LDPC codes. If $N(w)$ is the number of codewords of weight w for a given code, then the probability in (8) is $P(w) = N(w)/\binom{n}{w}$. Ideally, $P(w)$ would be constant for all $w > 0$, i.e., (8) would be an equality, which is precisely what happens when $A \sim \mathrm{Ber}(i \times n)$. More generally, the flatter $P(w)$ is, the better the statistical properties of R_i.

To demonstrate the superiority of PEG LDPC over the sparse ensemble we plot in Figure 3 the empirical value of $\log P(w)$ for parity matrices of size 20×40, derived by exhaustively enumerating each code's, roughly, $2^{40/2} \approx 1$ million codewords (exhaustive enumeration was chosen because the number of codewords can have non-trivial fluctuations for small n.) For the dense and sparse parity ensembles we generated 100 matrices each and report the mean; the PEG construction for LDPC is deterministic. We only considered codes with $n = 40$ variables, as exhaustive enumeration rapidly becomes intractable with n. Already, though, for $n = 40$ the behavior is very stable and it is easy to prove that flatness increases as n grows.

As can be seen, for a wide range of w both the sparse parity ensemble and the LDPC ensemble match $P(w)$ perfectly (the fact that dense parity itself is not flat for $w \notin [3, 36]$ is a finite-size effect). Crucially, though, for small w there is a big difference, with the sparse parity ensemble containing many more codewords (note that the vertical axis is logarithmic). The over-representation of low-weight codewords causes nearby pairs in $S \subseteq \Omega$ to contribute disproportionately to the sum in (7), potentially causing the variance of X_i to blow up if S is clustered, e.g., if S is a cube. No such blowup occurs for the PEG codes even for such small n, witnessing their very good statistical properties. As the study of the codeword weight distribution function of LDPC codes greatly exceeds the scope of this work, we leave a formal proof that thinning by LDPC codes give rise to small-variance estimators as future work.

We claimed earlier that LDPC codes should be far preferable to random parity matrices. To that end we plot the performance of CPLEX on the Ising grid in Figure 4. The only difference between the three plots is in the parity matrix A used to define $R_i = A\sigma + b$. The collapse of CPLEX beyond a certain number of constraints was already pointed out in Figure 1. For LDPC PEG codes no such collapse occurs and CPLEX remains competitive with LocalSolver and random affine maps until $i \approx 40$ even with a timeout at small as 30 seconds. In contrast, as demonstrated in Figure 5, LocalSolver, unaware of the variable/product

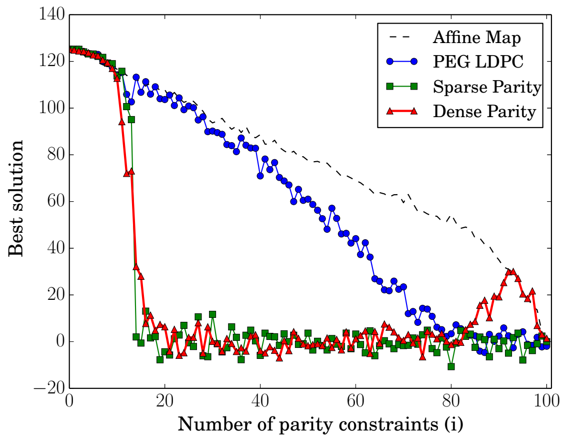

Figure 4: 10×10 Ising grid with $C = 1.0$ and $F = 0.1$. *Experiment:* for $i \in [0, 100]$, generate R_i via an $i \times 100$ parity matrix chosen from 3 different ensembles (see legend); seek $\max_{\sigma \in R_i} f(\sigma)$ for 30 seconds using CPLEX. *Plot:* for each i and each ensemble repeat the experiment 10 times and report the binary logarithm of the median value found. (As a yardstick, we also plot the values found by LocalSolver when R_i is a random affine subspace.)

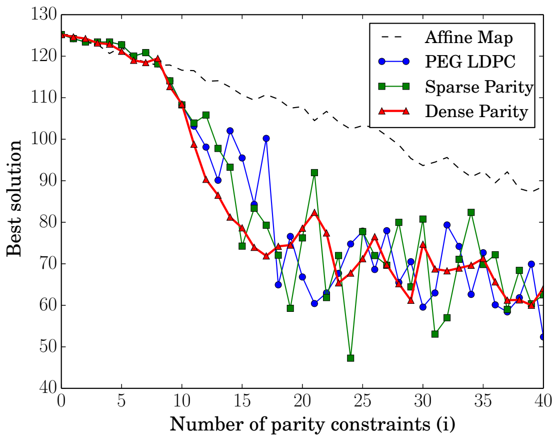

Figure 5: 10×10 Ising grid with $C = 1.0$ and $F = 0.1$. *Experiment:* for $i \in [0, 40]$, generate R_i via an $i \times 100$ parity matrix chosen from 3 different ensembles, or as a random affine subspace (see legend); seek $\max_{\sigma \in R_i} f(\sigma)$ for 30 seconds using LocalSolver. *Plot:* for each i and method repeat the experiment 10 times and report the binary logarithm of the median value found.

structure of Ω (and the factorization of f over $\{\psi_\alpha\}$) does not "feel" the difference between different parity check matrices, consistent with our hypothesis that it is the presence (and exploitation) of "safe" decoding moves that causes the dramatic improvement in the performance of CPLEX.

6 BRANCH-AND-BOUND

Our last observation is that not all $\hat{b}_i$ are equally important (or even necessary) for an accurate estimate $\widehat{Z}$. For instance, if $f(\sigma) \in \{0, 1\}$, it suffices to find the boundary k such that $b_i = 1$ for $i \leq k$ and $b_i = 0$ for $i > k$. Using binary search we can do this by solving only $\log n$, instead of n, optimization problems as in (2).

To generalize let $I = \{i_0, i_1, i_2, \ldots, i_s\}$ be the set of quantiles estimated so far, where $0 = i_0 < i_1 < \ldots < i_s = n$. Now define

$$U_I = b_0 + \sum_{j=0}^{s-1} b_{i_j} \left(\sum_{i=i_j}^{i_{j+1}-1} 2^i \right)$$

$$L_I = b_0 + \sum_{j=0}^{s-1} b_{i_j+1} \left(\sum_{i=i_j}^{i_{j+1}-1} 2^i \right) .$$

By construction, $U_I \geq U \geq Z \geq L \geq L_I$. Let $\widehat{U}_I$ and $\widehat{L}_I$ be the estimated counterparts, of U_I, L_I, respectively, with $\hat{b}_i$ in place of b_i. To identify the next quantile to estimate we consider the successive pairs $(i_\ell, i_{\ell+1})$ in I and for each such pair $(i_\ell, i_{\ell+1}) = (i, j)$ we define $\text{Gap}(i, j) = \left(\hat{b}_i - \hat{b}_j\right) \sum_{q=i}^{j} 2^q$. At each iteration, we chose the pair (i, j) with maximum gap and estimate b_k, where $k = \lfloor \frac{i+j}{2} \rfloor$. Once the ratio $\widehat{U}_I / \widehat{L}_I$ drops below the desired accuracy threshold, the process can stop early (see Table 6 for some indicative results).

Algorithm 1 Branch-and-Bound

```
 1: $\hat{b}_0 \leftarrow$ ESTIMATE$(b_0)$
 2: $\hat{b}_n \leftarrow$ ESTIMATE$(b_n)$
 3: $I \leftarrow \{0, n\}$
 4: $\widehat{U}_I \leftarrow \hat{b}_0 2^n$
 5: $\widehat{L}_I \leftarrow \hat{b}_n 2^n$
 6: while ($\widehat{U}_I > \widehat{L}_I \cdot$ Tolerance) do
 7:     Find successive $i, j \in I$ maximizing Gap$(i, j)$
 8:     $k \leftarrow \lfloor (i + j)/2 \rfloor$
 9:     $\hat{b}_k \leftarrow$ Estimate$(b_k)$
10:     $I \leftarrow I \cup \{k\}$
11:     Compute $\widehat{U}_I$, $\widehat{L}_I$
12: end while
13: Return $(\widehat{U}_I + \widehat{L}_I)/2$
```

The benefit of branch and bound is universal, i.e., independent of the $\{b_i\}$ estimation method. For example, over 24 problems on Ising grids it yielded a 7x average speedup.

$C \quad (F = 0.1)$	0.25	0.5	1.0	1.5	2	2.5	3
Speedup (x)	11	5	3	7	9	11	12

Table 1: Speedup by Branch and Bound

7 EXPERIMENTS

The ferromagnetic Ising grid is a canonical spin glass model. Binary variables (spins) $x_i \in \{\pm 1\}$ are placed on the vertices of a $\sqrt{n} \times \sqrt{n}$ grid (V, E) and have nearest-neighbor interactions and a local (to each spin) field. Thus,

$$f(x) = \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j) \ , \qquad (9)$$

with $\psi_i(x_i) = \exp(F_i x_i)$ and $\psi_{ij}(x_i) = \exp(C_{ij} x_i x_j)$, where the local fields F_i are i.i.d. $U[-F, F]$ and the coupling strengths C_{ij} are i.i.d. $U[0, C]$. (As $C_{ij} \geq 0$, configurations where neighboring spins align are favored.)

The model has been widely used as a test case for partition function estimation [4][8][14][15] due to its flexibility: as C is increased, the dominant contribution to the partition function shifts from configurations with many unaligned neighbors to configurations with few unaligned neighbors. We focus on the particularly challenging setting $C \approx 1$. A side benefit of this choice (not unrelated to hardness) is that a wide range of quantiles contribute significantly to Z, thus exercising each method for a wide range of thinning.

7.1 THE ALGORITHMS

Given the complementary nature of thinning via parity matrices and thinning via affine subspaces, it is natural to combine our two ideas into one algorithm that uses LDPC parity check matrices for small i ("light" thinning) and random affine subspaces for large i ("heavy" thinning). To reduce the confounding factors we simply used parity matrices for $i \in [1, 33]$ and affine subspaces for $i \in [34, 100]$.

Besides the junction-tree algorithm used to compute Z exactly, we also evaluated the Mean Field approximation, and Tree-Reweighted Belief Propagation (TRWBP) [6], providing a lower and an upper bound for the partition function, respectively. We use the libDAI [10] implementations of all three algorithms. The WISH algorithm is the CPLEX implementation by the authors of [14].

7.2 THE RESULTS

WISH is not competitive with our algorithm in certain settings, as indicated by Figure 6 (note that the vertical axis is logarithmic). For example, our algorithm achieves better accuracy with a 10-second timeout than WISH achieves with 360 seconds. When this difference in accuracy is combined with the Branch and Bound speedup, our algorithm is over 100x faster than WISH.

In Figure 7 we compare all four algorithms for $F = 0.1$ and various values of $C \in [0.25, 3.0]$. Rather than comparing run times, which in order to be fair would require adapting the timeout of each algorithm to the difficulty it experiences, we have chosen the more transparent experiment of running each algorithm with the same timeout of 360 seconds across all instances and all $i \in [n]$ and comparing the accuracy achieved in the final estimate of $\widehat{Z}$.

The case of high coupling strengths is easy for both algorithms as the dominant contribution to Z comes from few configurations of high probability and, thus, only light thinning is performed. For $C \in \{0.5, 0.75, 1.0\}$, though, our algorithm outperforms WISH by a significant margin (the vertical axis is logarithmic).

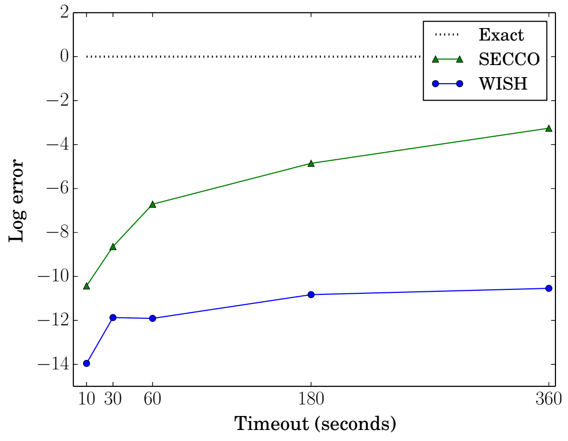

Figure 6: 10×10 Ising grid with $C = 1.0$ and $F = 0.1$. *Experiment:* run WISH and our algorithm with a timeout of $t \in \{10, 30, 60, 180, 360\}$ seconds per instance to get an estimate $\widehat{Z}$ (see legend). *Plot:* report $\log_2(\widehat{Z}/Z)$.

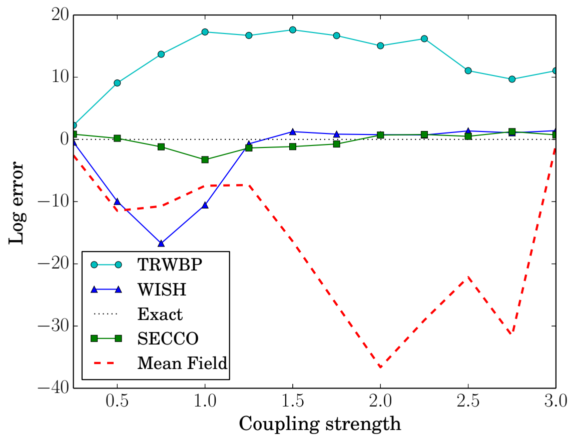

Figure 7: 10×10 Ising grid with $C \in [0.25, 3.0]$, $F = 0.1$. *Experiment:* for $C \in \{0.25, 0.50, \ldots, 3.0\}$, determine Z and run four algorithms to get an estimate $\widehat{Z}$ (see legend). Mean Field and TRWBP are run to termination. WISH and our algorithm have a 360 second timeout for each instance. *Plot:* For $C \in \{0.25, 0.50, \ldots, 3.0\}$ report $\log_2(\widehat{Z}/Z)$.

References

[1] A. Bulatov and M. Grohe. The complexity of partition functions. *Theoretical Computer Science*, 348(2-3):148–186, 2005.

[2] D. Allouche, S. de Givry, and T. Schiex. Toulbar2, an open source exact cost function network solver. Technical report, Technical report, INRIA, 2010.

[3] X. Hu, E. Eleftheriou, and D. Arnold. Regular and irregular progressive edge-growth tanner graphs. *IEEE Transactions on Information Theory*, 51(1):386–398, 2005.

[4] T. Hazan and T. Jaakkola. On the partition function and random maximum a-posteriori perturbations. In *ICML*. icml.cc / Omnipress, 2012.

[5] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[6] M. Wainwright, T. Jaakkola, and A. Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In *Workshop on Artificial Intelligence and Statistics*, volume 21, 2003.

[7] M. Jerrum and A. Sinclair. *Approximation Algorithms for NP-hard Problems*, chapter The Markov Chain Monte Carlo Method: An Approach to Approximate Counting and Integration, pages 482–520. PWS Publishing Co., Boston, MA, USA, 1997.

[8] T. Hazan, S. Maji, and T. Jaakkola. On sampling from the gibbs distribution with random maximum a-posteriori perturbations. In *NIPS*, pages 1268–1276, 2013.

[9] T. Benoist, B. Estellon, F. Gardi, R. Megel, and K. Nouioua. Localsolver 1.x: a black-box local-search solver for 0-1 programming. *4OR*, 9(3):299–316, 2011.

[10] J. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, August 2010.

[11] C. Gomes, A. Sabharwal, and B. Selman. Model counting: A new strategy for obtaining good bounds. In *AAAI*, pages 54–61. AAAI Press, 2006.

[12] C. Gomes, W. Hoeve, A. Sabharwal, and B. Selman. Counting CSP solutions using generalized XOR constraints. In *AAAI*, pages 204–209. AAAI Press, 2007.

[13] S. Ermon, C. Gomes, A. Sabharwal, and B. Selman. Optimization with parity constraints: From binary codes to discrete integration. In *UAI*. AUAI Press, Corvallis, Oregon, 2013.

[14] S. Ermon, C. Gomes, A. Sabharwal, and B. Selman. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *ICML*, volume 28 of *JMLR Proceedings*, pages 334–342. JMLR.org, 2013.

[15] S. Ermon, C. Gomes, A. Sabharwal, and B. Selman. Low-density parity constraints for hashing-based discrete integration. In *ICML*, volume 32 of *JMLR Proceedings*, pages 271–279. JMLR.org, 2014.

[16] M. Sipser. A complexity theoretic approach to randomness. In *STOC*, pages 330–335. ACM, 1983.

[17] M. Thurley. An approximation algorithm for #k-sat. In *STACS*, volume 14 of *LIPIcs*, pages 78–87. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[18] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[19] L. Valiant and V. Vazirani. Np is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(3):85–93, 1986.