# Taming the Curse of Dimensionality: Discrete Integration by Hashing and Optimization

**Stefano Ermon, Carla P. Gomes** {ERMONSTE,GOMES}@CS.CORNELL.EDU
Dept. of Computer Science, Cornell University, Ithaca NY 14853, U.S.A.

**Ashish Sabharwal** ASHISH.SABHARWAL@US.IBM.COM
IBM Watson Research Center, Yorktown Heights, NY 10598, U.S.A.

**Bart Selman** SELMAN@CS.CORNELL.EDU
Dept. of Computer Science, Cornell University, Ithaca NY 14853, U.S.A.

## Abstract

Integration is affected by the curse of dimensionality and quickly becomes intractable as the dimensionality of the problem grows. We propose a randomized algorithm that, with high probability, gives a constant-factor approximation of a general discrete integral defined over an exponentially large set. This algorithm relies on solving only a small number of instances of a discrete combinatorial optimization problem subject to randomly generated parity constraints used as a hash function. As an application, we demonstrate that with a small number of MAP queries we can efficiently approximate the partition function of discrete graphical models, which can in turn be used, for instance, for marginal computation or model selection.

## 1. Introduction

Computing integrals in very high dimensional spaces is a fundamental and largely unsolved problem of scientific computation (Dyer et al., 1991; Simonovits, 2003; Cai & Chen, 2010), with numerous applications ranging from machine learning and statistics to biology and physics. As the volume grows exponentially in the dimensionality, the problem quickly becomes computationally intractable, a phenomenon traditionally known as the *curse of dimensionality* (Bellman, 1961).

We revisit the problem of approximately computing discrete integrals, namely weighted sums over (extremely large) sets of items. This problem encompasses several important probabilistic inference tasks, such as computing marginals or normalization constants (partition function) in graphical models, which are in turn cornerstones for parameter and structure learning (Wainwright & Jordan, 2008).

There are two common approaches to approximate such large discrete sums: variational methods and sampling. Variational methods (Wainwright & Jordan, 2008; Jordan et al., 1999), often inspired by statistical physics, are very fast but do not provide quality guarantees. Since sampling and counting can be reduced to each other (Jerrum & Sinclair, 1997), approximate techniques based on sampling are quite popular, but they suffer from similar issues because the number of samples required to obtain a statistically reliable estimate often grows exponentially in the problem size. Importance sampling based techniques such as `SampleSearch` (Gogate & Dechter, 2011) provide lower bounds but without a tightness guarantee. Markov Chain Monte Carlo (MCMC) methods for sampling are asymptotically accurate, but guarantees for practical applications exist only in a limited number of cases (fast mixing chains) (Jerrum & Sinclair, 1997; Madras, 2002). They are therefore often used in a heuristic manner. In practice, their performance crucially depends on the choice of the proposal distributions, which often must be domain-specific and expert-designed (Girolami & Calderhead, 2011).

We introduce a randomized scheme that computes with high probability ($1 - \delta$ for any desired $\delta > 0$) an approximately correct estimate (within a factor of $1 + \epsilon$ of the true value for any desired $\epsilon > 0$) for general weighted sums defined over exponentially large

*Proceedings of the 30th International Conference on Machine Learning*, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

sets of items, such as the set of all possible variable assignments in a discrete probabilistic graphical model. From a computational complexity perspective, the counting problem we consider is complete for the #P complexity class (Valiant, 1979), a set of problems encapsulating the entire Polynomial Hierarchy and believed to be significantly harder than NP.

The key idea is to reduce this #P problem to a small number (polynomial in the dimensionality) of instances of a (NP-hard) combinatorial optimization problem defined on the same space and subject to randomly generated "parity" constraints. The rationale behind this approach is that although combinatorial optimization is intractable in the worst case, it has witnessed great success in the past 50 years in fields such as Mixed Integer Programming (MIP) and propositional Satisfiability Testing (SAT). Problems such as computing a Maximum a Posteriori (MAP) assignment, although NP-hard, can in practice often be approximated or solved exactly fairly efficiently (Park, 2002; Sontag et al., 2008; Riedel, 2008). In fact, modern solvers can exploit structure in real-world problems and prune large portions of the search space, often dramatically reducing the runtime. In contrast, in a #P counting problem such as computing a marginal probability, one needs to consider contributions of an exponentially large number of items.

Our algorithm, called **W**eighted-**I**ntegrals-And-**S**ums-By-**H**ashing (`WISH`), relies on randomized hashing techniques to probabilistically "evenly cut" a high dimensional space. Such hashing was introduced by Valiant & Vazirani (1986) to study the relationship between the number of solutions and the hardness of a combinatorial search. These techniques were also applied by Gomes et al. (2006a) and Chakraborty et al. (2013) to uniformly sample solutions for the SAT problem and to obtain bounds on their number (Gomes et al., 2006b). Our work is more general in that it can handle general weighted sums, such as the ones arising in probabilistic inference for graphical models. Our work is also closely related to recent work by Hazan & Jaakkola (2012), who obtain bounds on the partition function by taking suitable expectations of a combination of MAP queries over randomly perturbed models. We improve upon this in two crucial aspects, namely, our estimate is a constant factor approximation of the true partition function (while their bounds have no tightness guarantee), and we provide a concentration result showing that our bounds hold not just in expectation but with high probability with a polynomial number of MAP queries. Note that this is consistent with known complexity results regarding #P and $\mathrm{BPP}^{\mathrm{NP}}$; see Remark 1 below.

We demonstrate the practical efficacy of the `WISH` algorithm in the context of computing the partition function of random Clique-structured Ising models, Grid Ising models with known ground truth, and a challenging combinatorial application (Sudoku puzzle) completely out of reach of techniques such as Mean Field and Belief Propagation. We also consider the Model Selection problem in graphical models, specifically in the context of hand-written digit recognition. We show that our "anytime" and highly parallelizable algorithm can handle these problems at a level of accuracy and scale well beyond the current state of the art.

## 2. Problem Statement and Assumptions

Let $\Sigma$ be a (large) set of items. Let $w : \Sigma \to \mathbb{R}^+$ be a non-negative function that assigns a weight to each element of $\Sigma$. We wish to (approximately) compute the total weight of the set, defined as the following discrete integral or "partition function"

$$W = \sum_{\sigma \in \Sigma} w(\sigma) \tag{1}$$

We assume $w$ is given as input and that it can be compactly represented, for instance in a factored form as the product of conditional probabilities tables. Note however that our results are more general and do not rely on a factored representation.

**Assumption:** We assume that we have access to an *optimization oracle* that can solve the following constrained optimization problem

$$\max_{\sigma \in \Sigma} \quad w(\sigma) 1_{\{\mathcal{C}\}}(\sigma) \tag{2}$$

where $1_{\{\mathcal{C}\}} : \Sigma \to \{0, 1\}$ is an indicator function for a compactly represented subset $\mathcal{C} \subseteq \Sigma$, i.e., $1_{\{\mathcal{C}\}}(\sigma) = 1$ iff $\sigma \in \mathcal{C}$. For concreteness, we discuss our setup and assumptions in the context of probabilistic graphical models, which is our motivating application.

### 2.1. Inference in Graphical Models

We consider a graphical model specified as a factor graph with $N = |V|$ discrete random variables $x_i, i \in V$ where $x_i \in \mathcal{X}_i$. The global random vector $x = \{x_s, s \in V\}$ takes value in the cartesian product $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_N$. We consider a probability distribution over $x \in \mathcal{X}$ (called **configurations**) $p(x) = \frac{1}{Z} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha)$ that factors into potentials or factors $\psi_\alpha : \{x\}_\alpha \mapsto \mathbb{R}^+$, where $\mathcal{I}$ is an index set and $\{x\}_\alpha \subseteq V$ a subset of variables the factor $\psi_\alpha$ depends on, and $Z$ is a normalization constant known as the **partition function**.

Given a graphical model, we let $\Sigma = \mathcal{X}$ be the set of all possible configurations (variable assignments). Define a weight function $w : \mathcal{X} \to \mathbb{R}^+$ that assigns to each configuration a score proportional to its probability: $w(x) = \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha)$. $Z$ may then be rewritten as

$$Z = \sum_{x \in \mathcal{X}} w(x) = \sum_{x \in \mathcal{X}} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha) \tag{3}$$

Computing $Z$ is typically intractable because it involves a sum over an exponential number of configurations, and is often the most challenging inference task for many families of graphical models. Computing $Z$ is however needed for many inference and learning tasks, such as evaluating the likelihood of data for a given model, computing marginal probabilities, and parameter estimation (Wainwright & Jordan, 2008).

In the context of graphical models inference, we assume to have access to an optimization oracle that can answer Maximum a Posteriori (MAP) queries, namely, solve the following constrained optimization problem

$$\arg\max_{x \in \mathcal{X}} \quad p(x \mid \mathcal{C}) \tag{4}$$

that is, we can find the most likely state (and its weight) given some evidence $\mathcal{C}$. This is a strong assumption because MAP inference is known to be an NP-hard problem in general. Notice however that computing $Z$ is a #P-complete problem, a complexity class believed to be even harder than NP.

## 3. Preliminaries

We review some results on the construction and properties of universal hash functions (cf. Vadhan, 2011; Goldreich, 2011). A reader already familiar with these results may skip to the next section.

**Definition 1.** A family of functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is pairwise independent if the following two conditions hold when $H$ is a function chosen uniformly at random from $\mathcal{H}$. 1) $\forall x \in \{0,1\}^n$, the random variable $H(x)$ is uniformly distributed in $\{0,1\}^m$. 2) $\forall x_1, x_2 \in \{0,1\}^n$ $x_1 \neq x_2$, the random variables $H(x_1)$ and $H(x_2)$ are independent.

A simple way to construct such a function is to think about the family $\mathcal{H}$ of all possible functions $\{0,1\}^n \to \{0,1\}^m$. This is a family of not only pairwise independent but *fully* independent functions. However, each function requires $m2^n$ bits to be represented, and is thus impractical in the typical case where $n$ is large. On the other hand, *pairwise independent* hash functions can be constructed and represented in a much more compact way as follows; see Appendix for a proof.

**Proposition 1.** *Let* $A \in \{0,1\}^{m \times n}$, $b \in \{0,1\}^m$. *The family* $\mathcal{H} = \{h_{A,b}(x) : \{0,1\}^n \to \{0,1\}^m\}$ *where* $h_{A,b}(x) = Ax + b \mod 2$ *is a family of pairwise independent hash functions.*

The space $\mathcal{C} = \{x : h_{A,b}(x) = p\}$ has a nice geometric interpretation as the translated nullspace of the random matrix $A$, which is a finite dimensional vector space, with operations defined on the field $\mathbb{F}(2)$ (arithmetic modulo 2). We will refer to constraints of the form $Ax = b \mod 2$ as **parity constraints**, as they can be rewritten in terms of logical XOR operations as $A_{i1}x_1 \oplus A_{i2}x_2 \oplus \cdots \oplus A_{in}x_n = b_i$.

## 4. The `WISH` Algorithm

We start with the intuition behind our algorithm to approximate the value of $W$ called **W**eighted-**I**ntegrals-And-**S**ums-By-**H**ashing (`WISH`).

Computing $W$ as defined in Equation (1) is challenging because the sum is defined over an exponentially large number of items, i.e., $|\Sigma| = 2^n$ when there are $n$ binary variables. Let us define the **tail distribution** of weights as $G(u) \triangleq |\{\sigma \mid w(\sigma) \geq u\}|$. Note that $G$ is a non-increasing step function, changing values at no more than $2^n$ points. Then $W$ may be rewritten as $\int_{\mathbb{R}^+} G(u) \mathrm{d}u$, i.e., the total *area* $A$ under the $G(u)$ vs. $u$ curve. One way to approximate $W$ is to (implicitly) divide this area $A$ into either *horizontal* or *vertical* slices (see Figure 2), approximate the area in each slice, and sum up.

Suppose we had an efficient procedure to estimate $G(u)$ given any $u$. Then it is not hard to see that one could create enough slices by dividing up the x-axis, estimate $G(u)$ at these points, and estimate the area $A$ using quadrature. However, the natural way of doing this to any degree of accuracy would require a number of slices that grows at least logarithmically with the weight range on the x-axis, which is undesirable.

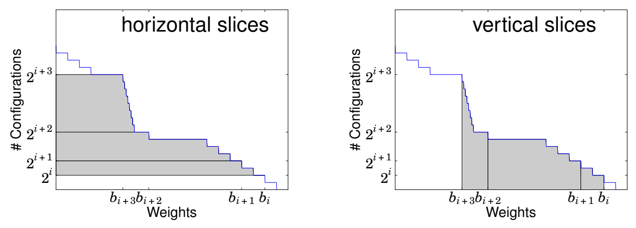


*Figure 2.* Horizontal vs. vertical slices for integration.

Alternatively, one could split the y-axis, i.e., the $G(u)$ value range $[0, 2^n]$, at geometrically growing values $1, 2, 4, \cdots, 2^n$, i.e., into bins of sizes $1, 1, 2, 4, \cdots, 2^{n-1}$. Let $b_0 \geq b_1 \geq \cdots \geq b_n$ be the weights of the configu-

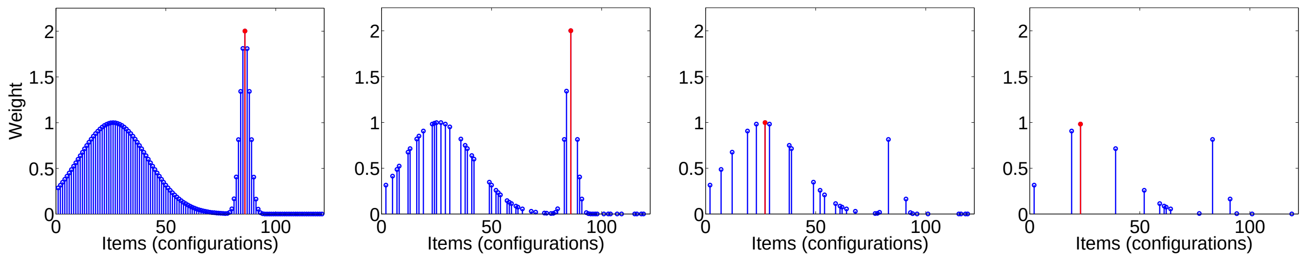


*Figure 1.* Visualization of the "thinning" effect of random parity constraints, after adding 0, 1, 2, and 3 parity constraints. Leftmost plot shows the original function to integrate. Constrained optimal solution in red.

rations at the split points. In other words, $b_i$ is the $2^i$-th quantile of the weight distribution. Unfortunately, despite the monotonicity of $G(u)$, the area in the horizontal slice defined by each bin is difficult to bound, as $b_i$ and $b_{i+1}$ could be arbitrarily far from each other. However, the area in the *vertical* slice defined by $b_i$ and $b_{i+1}$ must be bounded between $2^i(b_i - b_{i+1})$ and $2^{i+1}(b_i - b_{i+1})$, i.e., within a factor of 2. Thus, summing over the lower bound for all such slices and the left-most slice, the total area $A$ must be within a factor of 2 of $\sum_{i=0}^{n-1} 2^i(b_i - b_{i+1}) + 2^n b_n = b_0 + \sum_{i=1}^{n} 2^{i-1} b_i$. Of course, we don't know $b_i$. But if we could approximate each $b_i$ within a factor of $p$, we would get a $2p$-approximation to the area $A$, i.e., to $W$.

WISH provides an efficient way to realize this strategy, using a combination of randomized hash functions and an optimization oracle to approximate the $b_i$ values with high probability. Note that this method allows us to compute the partition function $W$ (or the area $A$) by estimating weights $b_i$ at $n+1$ carefully chosen points, which is "only" an optimization problem.

The key insight to compute the $b_i$ values is as follows. Suppose we apply to configurations in $\Sigma$ a randomly sampled pairwise independent hash function with $2^m$ buckets and use an optimization oracle to compute the weight $w_m$ of a *heaviest* configuration in a fixed (arbitrary) bucket. If we repeat this process $T$ times and consistently find that $w_m \geq w^*$, then we can infer by the properties of hashing that at least $2^m$ configurations (globally) are likely to have weight at least $w^*$. By the same token, if there were in fact at least $2^{m+c}$ configurations of a heavier weight $\hat{w} > w^*$ for some $c > 0$, there is a good chance that the optimization oracle will find $w_m \geq \hat{w}$ and we would not underestimate the weight of the $2^m$-th heaviest configuration. As we will see shortly, this process, using pairwise independent hash functions to keep variance low, allows us to estimate $b_i$ accurately with only $T = \mathrm{O}(\ln n)$ samples.

The pseudocode of WISH is shown as Algorithm 1. It is parameterized by the weight function $w$, the dimensionality $n$, a correctness parameter $\delta > 0$, and a constant $\alpha > 0$. Notice that the algorithm requires solving only $\Theta(n \ln n \ln 1/\delta)$ optimization instances (MAP inference) to compute a sum defined over $2^n$ items. In the following section, we formally prove that the output is a constant factor approximation of $W$ with probability at least $1-\delta$ (probability over the choice of hash functions). Figure 1 shows the working of the algorithm. As more and more random parity constraints are added in the outer loop of the algorithm ("levels" increasing from 1 to $n$), the configuration space is (pairwise-uniformly) thinned out and the optimization oracle selects the heaviest (in red) of the surviving configurations. The final output is a weighted sum over the median of $T$ such modes obtained at each level.

**Algorithm 1** WISH $(w : \Sigma \to \mathbb{R}^+, n = \log_2 |\Sigma|, \delta, \alpha)$

```
$T \leftarrow \left\lceil \frac{\ln(1/\delta)}{\alpha} \ln n \right\rceil$
for $i = 0, \cdots, n$ do
    for $t = 1, \cdots, T$ do
        Sample hash function $h^i_{A,b} : \Sigma \to \{0,1\}^i$, i.e.
            sample uniformly $A \in \{0,1\}^{i \times n}$, $b \in \{0,1\}^i$
        $w_i^t \leftarrow \max_\sigma w(\sigma)$ subject to $A\sigma = b \mod 2$
    end for
    $M_i \leftarrow \mathrm{Median}(w_i^1, \cdots, w_i^T)$
end for
Return $M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i$
```

**Remark 1.** The parity constraints $A\sigma = b \bmod 2$ do not change the worst-case complexity of an NP-hard optimization problem. Our result is thus consistent with the fact that #P can be approximated in $\mathrm{BPP}^{\mathrm{NP}}$, that is, one can approximately count the number of solutions with a randomized algorithm and a polynomial number of queries to an NP oracle (Goldreich, 2011).

**Remark 2.** Although the parity constraints we impose are simple linear equations over a field, they can make the optimization harder. For instance, finding a configuration with the smallest Hamming weight satisfying a set of parity constraints is known to be NP-hard, i.e. equivalent to computing the minimum distance of a parity code (Berlekamp et al., 1978; Vardy, 1997). On the other hand, most low density parity check codes can be solved extremely fast in practice using heuristic methods such as message passing.

**Remark 3.** Each of the optimization instances can be solved independently, allowing natural massive **parallelization**. We will also discuss how the algorithm can be used in an **anytime** fashion, and the implications of obtaining suboptimal solutions.

## 5. Analysis

Since many configurations can have identical weight, it will help for the purposes of the analysis to fix, w.l.o.g., a weight-based ordering of the configurations, and a natural partition of the $|\Sigma| = 2^n$ configurations into $n+1$ bins that the ordering induces.

**Definition 2.** Fix an ordering $\sigma_i, 1 \leq i \leq 2^n$, of the configurations in $\Sigma$ such that for $1 \leq j < 2^n$, $w(\sigma_j) \geq w(\sigma_{j+1})$. For $i \in \{0, 1, \cdots, n\}$, define $b_i \triangleq w(\sigma_{2^i})$. Define a special *bin* $B \triangleq \{\sigma_1\}$ and, for $i \in \{0, 1, \cdots, n-1\}$, define *bin* $B_i \triangleq \{\sigma_{2^i+1}, \sigma_{2^i+2}, \cdots, \sigma_{2^{i+1}}\}$.

Note that bin $B_i$ has precisely $2^i$ configurations. Further, for all $\sigma \in B_i$, it follows from the definition of the ordering that $w(\sigma) \in [b_{i+1}, b_i]$. This allows us to bound the sum of the weights of configurations in $B_i$ (the "horizontal" slices) between $2^i b_{i+1}$ and $2^i b_i$.

### 5.1. Estimating the Total Weight

Our main theorem, whose proof relies on the two lemmas below, is that Algorithm 1 provides a constant factor approximation to the partition function. The complete proof of the theorem and all lemmas may be found in the Appendix.

**Lemma 1.** *Let* $M_i = \text{Median}(w_i^1, \cdots, w_i^T)$ *be defined as in Algorithm 1 and* $b_i$ *as in Definition 2. Then, for any* $c \geq 2$*, there exists* $\alpha^*(c) > 0$ *such that for* $0 < \alpha \leq \alpha^*(c)$*,*

$$\Pr\left[M_i \in [b_{\min\{i+c,n\}}, b_{\max\{i-c,0\}}]\right] \geq 1 - \exp(-\alpha T)$$

**Lemma 2.** *Let* $L' \triangleq b_0 + \sum_{i=0}^{n-1} b_{\min\{i+c+1,n\}} 2^i$ *and* $U' \triangleq b_0 + \sum_{i=0}^{n-1} b_{\max\{i+1-c,0\}} 2^i$*. Then* $U' \leq 2^{2c} L'$*.*

**Theorem 1.** *For any* $\delta > 0$ *and positive constant* $\alpha \leq 0.0042$*, Algorithm 1 makes* $\Theta(n \ln n \ln 1/\delta)$ *MAP queries and, with probability at least* $(1-\delta)$*, outputs a 16-approximation of* $W = \sum_{\sigma \in \Sigma} w(\sigma)$*.*

*Proof Sketch.* It is clear from the pseudocode that it makes $\Theta(n \ln n \ln 1/\delta)$ MAP queries. For accuracy analysis, we can write $W$ as:

$$W \triangleq \sum_{j=1}^{2^n} w(\sigma_j) = w(\sigma_1) + \sum_{i=0}^{n-1} \sum_{\sigma \in B_i} w(\sigma)$$
$$\in \left[b_0 + \sum_{i=0}^{n-1} b_{i+1} 2^i, b_0 + \sum_{i=0}^{n-1} b_i 2^i\right] \triangleq [L, U]$$

Note that $U \leq 2L$ because $2L = 2b_0 + \sum_{i=0}^{n-1} b_{i+1} 2^{i+1} = b_0 + \sum_{\ell=0}^{n} b_\ell 2^\ell \geq U$. Hence, if we had access to the true values of all $b_i$, we could obtain a 2-approximation to $W$. We do not know true $b_i$ values, but Lemma 1 shows that the $M_i$ values computed by Algorithm 1 are sufficiently close to $b_i$ with high probability. Specifically, applying Lemma 1 with $T = \frac{\log(1/\delta)}{\alpha} \log n$, we can show that with probability at least $(1-\delta)$, the output of Algorithm 1 lies in $[L', U']$ as defined in Lemma 2. Observing that $[L, U]$ is contained in $[L', U']$ and applying Lemma 2, we have a $2^{2c}$-approximation of $W$. Fixing $c = 2$ and noting that $\alpha^*(2) \geq 0.0042$ finishes the proof. $\square$

### 5.2. Estimating the Tail Distribution

We can also estimate the entire tail distribution of the weights, defined as $G(u) \triangleq |\{\sigma \mid w(\sigma) \geq u\}|$.

**Theorem 2.** *Let* $M_i$ *be defined as in Algorithm 1,* $u \in \mathbb{R}^+$*, and* $q(u)$ *be the maximum* $i$ *such that* $\forall j \in \{0, \cdots, i\}, M_j \geq u$*. Then, for any* $\delta > 0$*, with probability* $\geq (1-\delta)$*,* $2^{q(u)}$ *is an 8-approximation of* $G(u)$ *computed using* $\text{O}(n \ln n \ln 1/\delta)$ *MAP queries.*

While this is an interesting result in its own right, if the goal is to estimate the total weight $W$, then the scheme in Section 5.1, requiring a total of only $\Theta(n \ln n \ln 1/\delta)$ MAP queries, is more efficient than first estimating the tail distribution for several values of $u$.

### 5.3. Improving the Approximation Factor

Given a $\kappa$-approximation algorithm such as Algorithm 1 and any $\epsilon > 0$, we can design a $(1+\epsilon)$-approximation algorithm with the following construction. Let $\ell = \log_{1+\epsilon} \kappa$. Define a new set of configurations $\Sigma^\ell = \Sigma \times \Sigma \times \cdots \times \Sigma$, and a new weight function $w' : \Sigma^\ell \to \mathbb{R}$ as $w'(\sigma_1, \cdots, \sigma_\ell) = w(\sigma_1) w(\sigma_2) \cdots w(\sigma_\ell)$.

**Proposition 2.** *Let* $\widehat{W}$ *be a* $\kappa$*-approximation of* $\sum_{\sigma' \in \Sigma^\ell} w'(\sigma')$*. Then* $\widehat{W}^{1/\ell}$ *is a* $\kappa^{1/\ell}$*-approximation of* $\sum_{\sigma \in \Sigma} w(\sigma)$*.*

To see why this holds, observe that $W' = \sum_{\sigma' \in \Sigma^\ell} w'(\sigma') = \left(\sum_{\sigma \in \Sigma} w(\sigma)\right)^\ell = W^\ell$. Since $\frac{1}{\kappa} W' \leq \widehat{W} \leq \kappa W'$, we obtain that $\widehat{W}^{1/\ell}$ must be a $\kappa^{1/\ell} = 1+\epsilon$ approximation of $W$.

Note that this construction requires running Algorithm 1 on an enlarged problem with $\ell$ times more variables. Although the number of optimization queries grows polynomially with $\ell$, increasing the number of variables might significantly increase the runtime.

### 5.4. Further Approximations

When the instances defined in the inner loop are not solved to optimality, Algorithm 1 still provides approximate *lower bounds* on $W$ with high probability.

**Theorem 3.** *Let $\widetilde{w}_i^t$ be suboptimal solutions for the optimization problems in Algorithm 1, i.e., $\widetilde{w}_i^t \leq w_i^t$. Let $\widetilde{W}$ be the output of Algorithm 1 with these suboptimal solutions. Then, for any $\delta > 0$, with probability at least $1-\delta$, $\frac{\widetilde{W}}{16} \leq W$.*

*Further, if $\widetilde{w}_i^t \geq \frac{1}{L} w_i^t$ for some $L > 0$, then with probability at least $1-\delta$, $\widetilde{W}$ is a $16L$-approximation to $W$.*

The output is always an approximate lower bound, even if the optimization is stopped early. The lower bound is monotonically non-decreasing over time, and is guaranteed to eventually reach within a constant factor of $W$. We thus have an **anytime** algorithm.

## 6. Experimental Evaluation

We implemented WISH using the open source solver ToulBar2 (Allouche et al., 2010) to solve the MAP inference problem. ToulBar2 is a complete solver (i.e., given enough time, it will find an optimal solution and provide an optimality certificate), and it was one of the winning algorithms in the UAI-2010 inference competition. We augmented ToulBar2 with the IBM ILOG CPLEX CP Optimizer 12.3 based techniques borrowed from Gomes et al. (2007) to efficiently handle the random parity constraints. Specifically, the set of equations $Ax = b \mod 2$ are linear equations over the field $\mathbb{F}(2)$ and thus allow for efficient propagation and domain filtering using Gaussian Elimination.

For our experiments, we run WISH in parallel using a compute cluster with 642 cores. We assign each optimization instance in the inner loop to one core, and finally process the results when all optimization instances have been solved or have reached a timeout.

For comparison, we consider Tree Reweighted Belief Propagation (Wainwright, 2003) which provides an upper bound on $Z$, Mean Field (Wainwright & Jordan, 2008) which provides a lower bound, and Loopy Belief Propagation (Murphy et al., 1999) which provides an estimate with no guarantees. We use the implementations available in the LibDAI library (Mooij, 2010).

### 6.1. Provably Accurate Approximations

For our first experiment, we consider the problem of computing the partition function, $Z$ (cf. Eqn. (3)), of random Clique-structured Ising models on $n$ binary variables $x_i \in \{0,1\}$ for $i \in \{1,\cdots,n\}$. The interaction between $x_i$ and $x_j$ is defined as $\psi_{ij}(x_i,x_j) = \exp(-w_{ij})$ when $x_i \neq x_j$, and 1 otherwise, where $w_{ij}$ is uniformly sampled from $[0, w\sqrt{|i-j|}]$ and $w$ is a parameter set to 0.2. We further inject some structure by introducing a closed chain of strong repulsive interactions uniformly sampled from $[-10w, 0]$. We consider models with $n$ ranging from 10 to 60. These models have treewidth $n$ and can be solved exactly (by brute force) only up to about $n = 25$ variables.

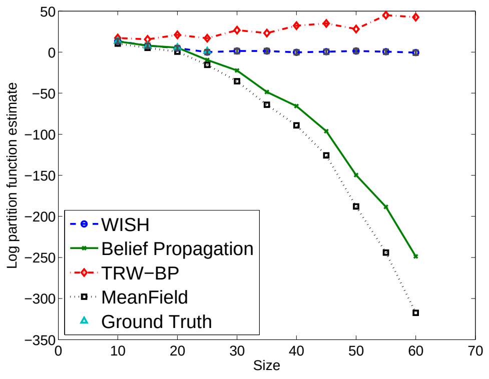
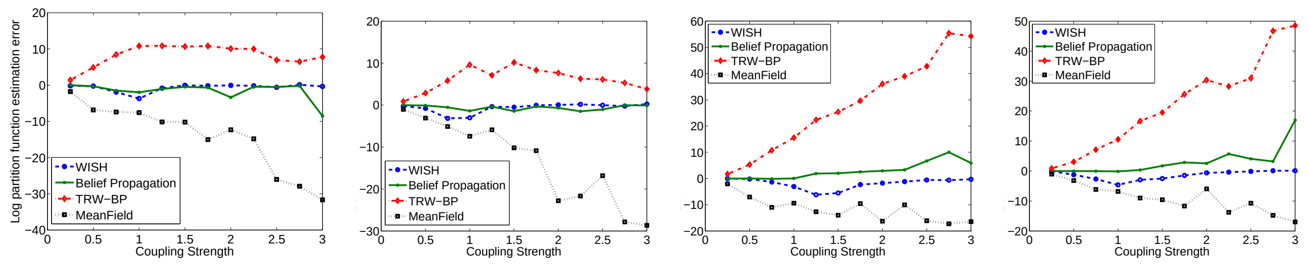


*Figure 3.* Log parition function for cliques.

Figure 3 shows the results using various methods for varying problem size. We also computed ground truth for $n \leq 25$ by brute force enumeration. While other methods start to diverge from the ground truth at around $n = 25$, our estimate, as predicted by Theorem 1, remains very accurate, visually overlapping in the plot. The actual estimation error is much smaller than the worst-case factor of 16 guaranteed by Theorem 1, as in practice over- and under-estimation errors tend to cancel out. For $n > 25$ we don't have ground truth, but other methods fall *well outside* the provable interval provided by WISH, reported as an error bar that is very small compared to the magnitude of errors made by the other methods.

All optimization instances generated by WISH for $n \leq 60$ were solved (in parallel) to optimality within a timeout of 8 hours, resulting in high confidence tight approximations of the partition function. We are not aware of any other practical method that can provide such guarantees for counting problems of this size, i.e., a weighted sum defined over $2^{60}$ items.



(a) Attractive. Field 0.1. (b) Attractive. Field 1.0. (c) Mixed. Field 0.1. (d) Mixed. Field 1.0.

*Figure 4.* Estimation errors for the log-partition function on $10 \times 10$ randomly generated Ising Grids.

## 6.2. Anytime Usage with Suboptimal Solutions

Next, we investigate the quality of our results when not all of the optimization instances can be solved to optimality because of timeouts, so that the strong theoretical guarantees of Theorem 1 do not apply (although Theorem 3 still applies). We consider $10 \times 10$ binary Grid Ising models, for which ground truth can be computed using the junction tree method (Lauritzen & Spiegelhalter, 1988). We use the same experimental setup as Hazan & Jaakkola (2012), who also use random MAP queries to derive bounds (without a tightness guarantee) on the partition function. Specifically, we have $n = 100$ binary variables $x_i \in \{-1, 1\}$ with interaction $\psi_{ij}(x_i, x_j) = \exp(w_{ij} x_i x_j)$. For the attractive case, we draw $w_{ij}$ from $[0, w]$; for the mixed case, from $[-w, w]$. The "local field" is $\psi_i(x_i) = \exp(f_i x_i)$ where $f_i$ is sampled uniformly from $[-f, f]$, where $f$ is a parameter with value 0.1 or 1.0.

Figure 4 reports the estimation *error* for the log-partition function, when using a timeout of 15 minutes. We see that WISH provides accurate estimates for a wide range of weights, often improving over all other methods. The slight performance drop of WISH for coupling strengths $w \approx 1$ appears to occur because in that weight range the terms corresponding to $i \approx n/2$ parity constraints are the most significant in the output sum $M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i$. Empirically, optimization instances with roughly $n/2$ parity constraints are often the hardest to solve, resulting in possibly a significant underestimation of the value of $W = Z$ when a timeout occurs. We do not directly compare with the work of Hazan & Jaakkola (2012) as we did not have access to their code. However, a visual look at their plots suggests that WISH would provide an improvement in accuracy, although with longer runtime.

## 6.3. Hard Combinatorial Structures

An interesting and combinatorially challenging graphical model arises from Sudoku, which is a popular number-placement puzzle where the goal is to fill a $9 \times 9$ grid (see Figure 5) with digits from $\{1, \cdots, 9\}$ so that the entries in each row, column, and $3 \times 3$ block composing the grid, are all distinct. The puzzle can be encoded as a graphical model with 81 discrete variables with domain $\{1, \cdots, 9\}$, with potentials $\psi_\alpha(\{x\}_\alpha) = 1$ if and only if all variables in $\{x\}_\alpha$ are different, and $\alpha \in \mathcal{I}$ where $\mathcal{I}$ is an index set containing the subsets of variables in each row, column, and block. This defines a uniform probability distribution over all valid complete Sudoku grids (a non-valid grid has probability zero), and the normalization constant $Z_s$ equals the total number of valid grids. It is known that $Z_s \approx 6.671 \times 10^{21}$. This number was computed exactly with a combination of computer enumeration and clever exploitation of symmetry properties (Felgenhauer & Jarvis, 2005). Here, we attempt to approximately compute this number using the general-purpose scheme WISH.

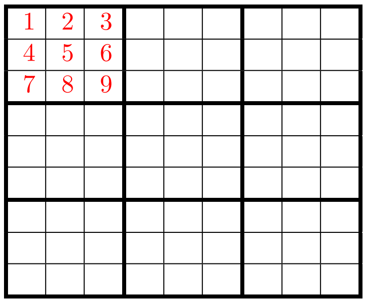


*Figure 5.* Partially completed Sudoku puzzle.

First, following Felgenhauer & Jarvis (2005), we simplify the problem by fixing the first block as in Figure 5, obtaining a new problem over 72 variables whose normalization constant is $Z' = Z_s/9! \approx 2^{54}$. Next, since we are dealing with a feasibility rather than optimization problem, we replace ToulBar2 with CryptoMiniSAT (Soos et al., 2009), a SAT solver designed for unweighted cryptographic problems and which natively supports parity constraints. We observed that

WISH can consistently find solutions (60% of the times) after adding 52 random parity constraints, while for 53 constraints the success rate drops below 0.5, at 45%. Therefore $M_i = 1$ in Algorithm 1 for $i \leq 52$ and there should thus be at least $2^{52} \cdot 9! \approx 1.634 \times 10^{21}$ solutions to the Sudoku puzzle. Although Theorem 1 cannot be applied due to timeouts for larger values of $i$, this estimate is clearly very close to the known true count. In contrast, the simple "local reasoning" done by variational methods is not powerful enough to find even a single solution. Mean Field and Belief Propagation report an estimated solution count of $\exp(-237.921)$ and $\exp(-119.307)$, resp., on a relaxed problem where violating a constraint gives a penalty $\exp(-10)$ (similar results are obtained using a wide range of weights to model hard constraints). A sophisticated adapative MCMC approach tailored for (weighted) SAT instances (Ermon et al., 2011) reports $5.6822 \times 10^{21}$ solutions, with a runtime of about 45 minutes.

## 6.4. Model Selection

Many inference and learning tasks require computing the normalization constant of graphical models. For instance, it is needed to evaluate the likelihood of observed data for a given model. This is necessary for Model Selection, i.e., to rank candidate models, or to trigger early stopping during training when the likelihood of a validation set starts to decrease, in order to avoid overfitting (Desjardins et al., 2011).

We train Restricted Boltzmann Machines (RBM) (Hinton et al., 2006) using Contrastive Divergence (CD) (Welling & Hinton, 2002; Carreira-Perpinan & Hinton, 2005) on MNIST hand-written digits dataset. In an RBM there is a layer of $n_h$ hidden binary variables $h = h_1, \cdots, h_{n_h}$ and a layer of $n_v$ binary visible units $v = v_1, \cdots, v_{n_v}$. The joint probability distribution is given by $P(h, v) = \frac{1}{Z}\exp(b'v + c'h + h'Wv)$. We use $n_h = 50$ hidden units and $n_v = 196$ visible units. We learn the parameters $b, c, W$ using CD-$k$ for $k \in \{1, 10, 15\}$, where $k$ denotes the number of Gibbs sampling steps used in the inference phase, with 15 training epochs and minibatches of size 20.

Figure 6 depicts confabulations (samples generated with Gibbs sampling) from the three learned models. To evaluate the loglikelihood of the data and determine which model is the best, one needs to compute $Z$. We use WISH to estimate this quantity, with a timeout of 10 minutes, and then rank the models according to the average loglikelihood of the data. The scores we obtain are $-41.70, -40.35, -40.01$ for $k = 1, 10, 15$, respectively (larger scores means higher likelihood). In this case ToulBar2 was not able to prove optimality for all instances, so only Theorem 3 applies to these results. Although we do not have ground truth, it can be seen that the ranking of the models is consistent with what visually appears closer to a large collection of hand-written digits in Figure 6. Note that $k = 1$ is clearly not a good representative, because of the highly uneven distribution of digit occurrences. The ranking of WISH is also consistent with the fact that using more Gibbs sampling steps in the inference phase should provide better gradient estimates and therefore a better learned model. In contrast, Mean Field results in scores $-35.47, -36.08, -36.84$, resp., and would thus rank the models in reverse order of what is visually the most representative order.

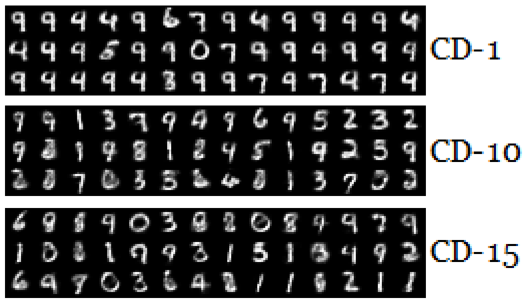


*Figure 6.* Model selection for hand-written digits: confabulations from RBM models trained with CD-k for $k \in \{1, 10, 15\}$.

## 7. Conclusion

We introduced WISH, a randomized algorithm that, with high probability, gives a constant-factor approximation of a general discrete integral defined over an exponentially large set. WISH reduces the intractable counting problem to a small number of instances of a combinatorial optimization problem subject to parity constraints used as a hash function. In the context of graphical models, we showed how to approximately compute the normalization constant, or partition function, using a small number of MAP queries. Using state-of-the-art combinatorial optimization tools, we are thus able to provide discrete integral or partition function estimates with approximation guarantees at a scale that could till now be handled only heuristically. One advantage of our method is that it is massively parallelizable, allowing it to easily benefit from the increasing availability of large compute clusters. Finally, it is an anytime algorithm which can also be stopped early to obtain empirically accurate estimates that provide lower bounds with a high probability.

**Acknowledgments:** Research supported by NSF grants #0832782 and #1059284.

## References

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

Bellman, R.E. *Adaptive control processes: A guided tour.* Princeton University Press (Princeton, NJ), 1961.

Berlekamp, E., McEliece, R., and Van Tilborg, H. On the inherent intractability of certain coding problems. *Information Theory, IEEE Transactions on*, 24(3):384–386, 1978.

Cai, J.Y. and Chen, X. A decidable dichotomy theorem on directed graph homomorphisms with non-negative weights. In *FOCS*, 2010.

Carreira-Perpinan, M.A. and Hinton, G.E. On contrastive divergence learning. In *Artificial Intelligence and Statistics*, volume 2005, pp. 17, 2005.

Chakraborty, S., Meel, K., and Vardi, M. A scalable and nearly uniform generator of SAT witnesses, 2013. To appear.

Desjardins, G., Courville, A., and Bengio, Y. On tracking the partition function. In *NIPS-2011*, pp. 2501–2509, 2011.

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

Ermon, S., Gomes, C., Sabharwal, A., and Selman, B. Accelerated Adaptive Markov Chain for Partition Function Computation. In *NIPS-2011*, 2011.

Felgenhauer, B. and Jarvis, F. Enumerating possible Sudoku grids. *Mathematical Spectrum*, 2005.

Girolami, M. and Calderhead, B. Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods. *J. of the Royal Statistical Society*, 73(2):123–214, 2011.

Gogate, V. and Dechter, R. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence*, 175(2):694–729, 2011.

Goldreich, O. Randomized methods in computation. *Lecture Notes*, 2011.

Gomes, Carla P., van Hoeve, Willem Jan, Sabharwal, Ashish, and Selman, Bart. Counting CSP solutions using generalized XOR constraints. In *AAAI*, 2007.

Gomes, C.P., Sabharwal, A., and Selman, B. Near-uniform sampling of combinatorial spaces using XOR constraints. In *NIPS-2006*, pp. 481–488, 2006a.

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

Hazan, T. and Jaakkola, T. On the partition function and Random Maximum A-Posteriori perturbations. In *ICML*, 2012.

Hinton, G.E., Osindero, S., and Teh, Y.W. A fast learning algorithm for Deep Belief Nets. *Neural Computation*, 18 (7):1527–1554, 2006.

Jerrum, M. and Sinclair, A. The Markov chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pp. 482–520, 1997.

Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., and Saul, L.K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Lauritzen, Steffen L and Spiegelhalter, David J. Local computations with probabilities on graphical structures and their application to expert systems. *J. of the Royal Statistical Society, B (Methodological)*, pp. 157–224, 1988.

Madras, N.N. *Lectures on Monte Carlo Methods.* American Mathematical Society, 2002. ISBN 0821829785.

Mooij, J.M. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *JMLR*, 11:2169–2173, 2010.

Murphy, K.P., Weiss, Y., and Jordan, M.I. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, 1999.

Park, J.D. Using weighted MAX-SAT engines to solve MPE. In *AAAI-2002*, pp. 682–687, 2002.

Riedel, Sebastian. Improving the accuracy and efficiency of MAP inference for Markov Logic. In *UAI-2008*, pp. 468–475, 2008.

Simonovits, M. How to compute the volume in high dimension? *Math. programming*, 97(1):337–374, 2003.

Sontag, David, Meltzer, Talya, Globerson, Amir, Jaakkola, Tommi, and Weiss, Yair. Tightening LP relaxations for MAP using Message Passing. In *UAI*, pp. 503–510, 2008.

Soos, M., Nohl, K., and Castelluccia, C. Extending SAT solvers to cryptographic problems. *SAT*, 2009.

Vadhan, S. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 2011.

Valiant, L.G. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3): 410–421, 1979.

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

Vardy, Alexander. Algorithmic complexity in coding theory and the minimum distance problem. In *STOC*, 1997.

Wainwright, M.J. Tree-reweighted belief propagation algorithms and approximate ML estimation via pseudo-moment matching. In *AISTATS*, 2003.

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

Welling, M. and Hinton, G. A new learning algorithm for mean field Boltzmann Machines. *Artificial Neural Networks: ICANN 2002*, pp. 82–82, 2002.