Low-Density Parity Constraints for Hashing-Based Discrete Integration

Stefano Ermon ERMONSTE@CS.CORNELL.EDU
Carla P. Gomes 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

In recent years, a number of probabilistic inference and counting techniques have been proposed that exploit pairwise independent hash functions to infer properties of succinctly defined high-dimensional sets. While providing desirable statistical guarantees, typical constructions of such hash functions are themselves not amenable to efficient inference. Inspired by the success of LDPC codes, we propose the use of low-density parity constraints to make inference more tractable in practice. While not strongly universal, we show that such sparse constraints belong to a new class of hash functions that we call Average Universal. These weaker hash functions retain the desirable statistical guarantees needed by most such probabilistic inference methods. Thus, they continue to provide provable accuracy guarantees while at the same time making a number of algorithms significantly more scalable in practice. Using this technique, we provide new, tighter bounds for challenging discrete integration and model counting problems.

1. Introduction

Many fundamental problems in machine learning and statistics, such as evaluating expectations or partition functions, can be stated as computing integrals in very high dimensional spaces. Since exact integration is generally intractable due to the curse of dimensionality (Bellman, 1961), a number of approximation techniques have been introduced. Sampling (Jerrum & Sinclair, 1997; Madras, 2002) and variational methods (Wainwright & Jordan, 2008; Jordan et al., 1999) are perhaps the most popular, but rarely provide formal guarantees in practice. Perturbation based methods (Papandreou & Yuille, 2011; Hazan et al., 2013) exploiting extreme value statistics provide a novel way of using MAP queries to recover the exact partition function and samples. However, their correctness relies heavily on fully independent Gumbel perturbations applied to exponentially many configurations.

A new family of provably approximately correct probabilistic inference and discrete integration algorithms seeks to overcome these limitations (Chakraborty et al., 2013a;b; Ermon et al., 2013a;b; Gomes et al., 2006b; 2007b). These techniques are all based on universal hash functions and require access to an optimization oracle to, e.g., answer a small (low degree polynomial) number of MAP inference queries. Since these oracle queries are NP-easy optimization problems, this strategy is desirable because discrete integration is a #P-hard problem (Valiant, 1979), a complexity class believed to be even harder than NP.

This reduction based on universal hashing has been well studied in theoretical computer science (Valiant & Vazirani, 1986; Bellare et al., 2000; Sipser, 1983). The focus is often on constructions that are optimal in terms of the number of random bits used. Typically, constructions exploit modular arithmetic and can also be interpreted as parity or XOR constraints. In practice, one implements the optimization oracle with a combinatorial reasoning tool, such as a Boolean satisfiability (SAT) (Gomes et al., 2006b) or Integer Programming (Ermon et al., 2013a) solver. Although requiring exponential time in the worst-case, real-

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

world instances are often solved quickly, and the implementation of the hash function heavily affects the runtime. One must thus design hash functions that not only have good statistical properties but are also easy to reason with in practice. This is closely related to coding theory, where one seeks codes that have good properties assuming NP-hard optimal decoding is possible, but which also lead to practical, efficient suboptimal decoding algorithms (MacKay, 1999).

We introduce a new class of hash functions, termed Average Universal, that are statistically weaker than traditional ones but strong enough to be used for discrete integration and counting. Specifically, they retain the desirable property that for large enough sets, the size of each hash "bucket" is sufficiently concentrated around its mean. The main advantage is that they can be implemented with sparse low-density parity constraints, which are empirically much easier to handle. Thus, our construction provides a tradeoff between statistical properties and computational efficiency of combinatorial reasoning with such constraints. This idea applies to a wide range of probabilistic inference and counting techniques that are based on universal hashing (Ermon et al., 2013b; Chakraborty et al., 2013a; Gomes et al., 2006b; 2007b), which would all benefit from our results.

To illustrate the gains, we empirically demonstrate that the WISH algorithm for discrete integration (Ermon et al., 2013b) benefits enormously from sparse parity constraints when an Integer Programming solver is used as optimization oracle (Ermon et al., 2013a). Specifically, given the same computational power, we obtain much better estimates on the partition function of graphical models thanks to the sparser parity constraints. The improvement over WISH comes at no cost, as we maintain the same theoretical properties of the original algorithm. Further, we show that model counting techniques based on modern SAT solvers (Gomes et al., 2006b) also greatly benefit from our new hash function construction.

2. Preliminaries

We start with definitions of standard universal hash functions (cf. Vadhan, 2011; Goldreich, 2011).

Definition 1. A family of functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is ϵ-SU (Strongly Universal) if the following two conditions hold when H is 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) $\forall x_1, x_2 \in \{0,1\}^n$ $x_1 \neq x_2$, $\forall y_1, y_2 \in \{0,1\}^m$, it holds that $P[H(x_1) = y_1, H(x_2) = y_2] \leq \epsilon/2^m$.

It can be verified that $\epsilon \geq 1/2^m$, and the case $\epsilon = 1/2^m$ corresponds to pairwise independent hash functions.

Definition 2. 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 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.

Statistically optimal functions can be constructed by considering the family $\mathcal{H}$ of all possible functions from $\{0,1\}^n$ to $\{0,1\}^m$. It is easy to verify that this is a family of *fully* independent functions. However, functions from this family require $m2^n$ bits to be specified, making this construction not very useful for large n. On the other hand, *pairwise independent* hash functions can be specified compactly. They are generally based on modular arithmetic constraints of the form $Ax = b \mod 2$, referred to as parity or XOR constraints.

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.*

2.1. Probabilistic Inference by Hashing

Recently there has been a range of probabilistic inference, discrete integration and counting, and sampling methods relying heavily on universal hash functions, such as WISH (Ermon et al., 2013b;a), ApproxMC (Chakraborty et al., 2013b), MBound and Hybrid-MBound (Gomes et al., 2006b), and XORSample (Gomes et al., 2006a). These methods all build upon the original theoretical ideas by Valiant & Vazirani (1986); Bellare et al. (2000); Sipser (1983).

The key idea is that one can reliably estimate properties of a very large (high-dimensional) set S (such as the integral of a function) by randomly dividing it into cells using a hash function h and looking at properties of a randomly chosen, lower-dimensional cell $h^{-1}(y) \cap S$. Remarkably, although $h^{-1}(y)$ is exponentially large, by the properties of the hash functions used it is possible to specify and represent this set in a compact way, without having to enumerate each individual element. For example, applying the hash function to all possible variable assignments for some probabilistic model (e.g., a discrete graphical model) is achieved by adding to the model a set of randomly generated parity constraints (e.g., extra factors), that can be specified in a compact way.

To work with high probability, this family of algorithms relies on good statistical properties of the hash functions. Specifically, they need to behave in a uniform and concentrated way, so that it is possible to predict $|h^{-1}(y) \cap S|$ (as a function of unknown $|S|$) with high probability, no mat-

ter what the structure of S is. Full independence would be ideal, as it would make the structure of S irrelevant. However, fully independent hash functions are computational intractable. If there is high correlation between $\{h(x)\}_{x\in S}$, things might not work. In the extreme case, even if $\mathcal{H}$ is uniform (property 1), it is possible that all the random variables $h(x), x \in S$ are identical, i.e., the division into cells does not break up S in a nice way — S is contained in a single cell in this extreme case. It turns out that pairwise independence suffices and is also computationally tractable. All schemes mentioned earlier rely on pairwise independence.

2.2. (k, δ)-Concentration and Counting

Formally, let $S \subseteq \{0,1\}^n$ and $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a family of hash functions. Let h be a hash function chosen uniformly from $\mathcal{H}$. Then, for $y \in \{0,1\}^m$, define the following random variable:

$$X(h, S, y) = |h^{-1}(y) \cap S| \tag{1}$$

Let $\mu(h, S, y) = \mathrm{E}[X(h, S, y)]$ denote its expected value and $\sigma(h, X, y)$ the standard deviation. Note that if $\mathcal{H}$ is universal, then $\mu(h, S, y) = |S|/2^m$. For brevity, we will sometimes use X and μ when h, S, and y are implicit in the context. Our main interest is in understanding how concentrated X is around μ, under various choices of $\mathcal{H}$. We introduce a general notion of concentration that will come handy:

Definition 3. Let $k \geq 0$ and $\delta > 2$. Let X be a random variable with $\mu = \mathrm{E}[X]$. Then X is *strongly* (k, δ)*-concentrated* if $\Pr[|X - \mu| \geq \sqrt{k}] \leq 1/\delta$ and *weakly* (k, δ)*-concentrated* if both $\Pr[X \leq \mu - \sqrt{k}] \leq 1/\delta$ and $\Pr[X \geq \mu + \sqrt{k}] \leq 1/\delta$.

For a given δ, smaller k corresponds to higher concentration. Clearly, strong (k, δ)-concentration implies weak (k, δ)-concentration and, for $k' > k$, (k, δ)-concentration implies (k', δ)-concentration. Further, by union bound, weak (k, δ)-concentration implies strong $(k, \delta/2)$-concentration.

As an illustrative example of how (k, δ)-concentration can be used for estimating sizes of high-dimensional sets, consider the following simple randomized algorithm $\mathcal{A}$ for approximately computing $|S|$ with high probability. Let $\mathcal{H}^i = \{h : \{0,1\}^n \to \{0,1\}^i\}$ for $i = 1, 2, \ldots, n$ be universal families of hash functions. For i increasing from 1 to m, compute "*is the set* $h_t^{-1}(0) \cap S$ *empty*" T times, each time with a different hash function h_t chosen uniformly from $\mathcal{H}^i$. If the answer is "yes" for a majority of the T times, stop increasing i and return 2^{i-1} as the estimate of $|S|$.

Proposition 2. *Let* $S \subseteq \{0,1\}^n$. *If* $\mathcal{H}^i = \{h : \{0,1\}^n \to \{0,1\}^i\}$, $i \in \{1, 2, \ldots, n\}$ *are universal families of hash functions such that* $X(h, S, y)$ *is weakly* $(\mu^2, 4)$*-concentrated, then* $\mathcal{A}$ *using* $\mathcal{H}^i$ *and* $T \geq 8\ln(8n)$ *correctly computes* $|S|$ *within a factor of* 4 *with probability at least* $3/4$.

3. Concentration and Hash Families

We discuss how the statistical properties of various hash families $\mathcal{H}$ influence the strength of (k, δ)-concentration of $X = |h^{-1}(y) \cap S|$. Proofs may be found in the appendix. Without any assumptions on the nature of $\mathcal{H}$, Chebychev's inequality and Cantelli's one-sided inequalities yield the following general observation for strong and weak concentration, resp.:

Proposition 3. *Let* $\delta > 2$ *and* $\mathcal{H}$ *be a family of hash functions. For any* $S \subseteq \{0,1\}^n$ *and* $y \in \{0,1\}^m$, *and for* $h \in_R \mathcal{H}$, $X(h, S, y)$ *is strongly* $(\delta\sigma^2, \delta)$*-concentrated and weakly* $((\delta - 1)\sigma^2, \delta)$*-concentrated.*

Ideally, one would like to choose a family of fully independent hash functions, which results in very strong concentration guarantees from Chernoff's bounds:

Proposition 4. *Let* $\delta > 2$, $c > 3$ *and* $\mathcal{H}$ *be a family of fully independent hash functions. For any* $S \subseteq \{0,1\}^n$ *and* $y \in \{0,1\}^m$, *and for* $h \in_R \mathcal{H}$, $X(h, S, y)$ *is strongly* $((c\ln\delta)\mu, \delta)$*-concentrated and weakly* $((3\ln\delta)\mu, \delta)$*-concentrated.*

However, as discussed earlier, it is often impossible to construct such a family. One commonly uses a family of only pairwise independent hash functions, which have compact constructions involving objects such as parity or XOR constraints. The concentration guarantees we get are much weaker but still very powerful:

Proposition 5. *Let* $\delta > 2$ *and* $\mathcal{H}$ *be a family of pairwise independent hash functions. For any* $S \subseteq \{0,1\}^n$ *and* $y \in \{0,1\}^m$, *and for* $h \in_R \mathcal{H}$, $X(h, S, y)$ *is strongly* $(\delta\mu, \delta)$*-concentrated and weakly* $((\delta - 1)\mu, \delta)$*-concentrated.*

This follows from observing that pairwise independence implies $\sigma^2 = |S|/2^m(1 - 1/2^m) < \mu$ and then applying Prop. 3.

Although one can compactly represent pairwise independent hash functions as m parity constraints (cf. Prop. 1), these constraints have average length $n/2$. Such long parity constraints are often particularly hard to reason about using standard inference methods (see Experimental section below). To start addressing this issue, we observe that in many applications, $O(\mu)$-concentration is unnecessary and it suffices to have only $O(\mu^2)$-concentration. An example of this is Proposition 2. We next discuss how one can exploit ϵ-SU hash functions to explore this wide spectrum of possible concentrations by varying ϵ.

Theorem 1. *Let $\delta > 2$, $\epsilon \geq 1/2^m$, and $\mathcal{H}$ be a family of ϵ-SU hash functions. For any $S \subseteq \{0,1\}^n$ and $y \in \{0,1\}^m$, and for $h \in_R \mathcal{H}$, $X(h,S,y)$ is strongly $(\delta\mu(1+\epsilon(|S|-1)-\mu),\delta)$-concentrated and weakly $((\delta-1)\mu(1+\epsilon(|S|-1)-\mu),\delta)$-concentrated.*

Proof. The variance of X can be computed as follows.

$$\begin{aligned}\mathrm{E}[X(h,S,y)^2] &= \sum_{s,s' \in S} \mathrm{E}[1_{h(s)=y,h(s')=y}] \\ &= \sum_{s \in S} \mathrm{E}[1_{h(s)=y}] + \sum_{s \neq s'} \mathrm{E}[1_{h(s)=y,h(s')=y}] \\ &\leq \mu + |S|(|S|-1)\epsilon/2^m \quad \text{(from SU)} \\ &= \mu(1+\epsilon(|S|-1))\end{aligned}$$

Therefore, $\sigma^2 = \mathrm{E}[X^2] - \mu^2 \leq \mu(1+\epsilon(|S|-1)-\mu)$. The result now follows from Prop. 3. □

Corollary 1. *Let $\delta > 2$, $\epsilon \geq 1/2^m$, and $\mathcal{H}$ be a family of ϵ-SU hash functions. For any $S \subseteq \{0,1\}^n$ and $y \in \{0,1\}^m$, and for $h \in_R \mathcal{H}$, $X(h,S,y)$ is strongly (μ^2,δ)-concentrated whenever $\epsilon \leq (\frac{\mu}{\delta} + \mu - 1)/(|S|-1)$ and weakly (μ^2,δ)-concentrated whenever $\epsilon \leq (\frac{\mu}{\delta-1} + \mu - 1)/(|S|-1)$.*

Thus, by increasing ϵ, we can achieve lower (but still acceptable) levels of concentration of X. In practice, however, it is not easy to construct ϵ-SU families that allow efficient inference. Simply using sparser parity constraints (i.e., with fewer than $n/2$ variables on average), for instance, does not lead to SU hash functions because if $s, s' \in S$ are close in Hamming distance, sparser parity-based hash functions will act on them in a very correlated way. The next section provides a way around this.

4. Average Universal Hashing

We now define a new family of hash functions that have the same statistical concentration properties as ϵ-SU (namely, the guarantees in Theorem 1) but are computationally much more tractable for inference methods.

Definition 4. A family of functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is (ϵ,i)-AU (Average Universal) if the following two conditions hold when H is a function chosen uniformly at random from $\mathcal{H}$.

- $\forall x \in \{0,1\}^n$, the random variable $H(x)$ is uniformly distributed in $\{0,1\}^m$.
- $\forall S \subseteq \{0,1\}^n$, $|S| = i$, $\forall y_1, y_2 \in \{0,1\}^m$, the following property holds $\sum_{x_1,x_2 \in S; x_1 \neq x_2} \Pr[H(x_1) = y_1, H(x_2) = y_2] \leq |S|(|S|-1)\epsilon/2^m$.

In other words, we allow pairs of random variables $H(x_1), H(x_2)$ to be potentially heavily correlated, for instance $\Pr[H(x_1) = y_1, H(x_2) = y_2]$ could be much larger than $\epsilon/2^m$. However, it needs to balance out so that the average correlation among configuration pairs on (large enough sets) S is smaller than $\epsilon/2^m$.

Proposition 6. *Let $\mathcal{H}$ be a family of hash functions. (a) If $\mathcal{H}$ is $(\epsilon,2)$-AU, then $\mathcal{H}$ is also ϵ-SU. (b) If $\mathcal{H}$ is ϵ-SU, then $\mathcal{H}$ is also (ϵ,i)-AU for all $i \geq 2$. (c) If $\mathcal{H}$ is (ϵ,i)-AU, then $\mathcal{H}$ is also $(\epsilon,i+1)$-AU.*

Theorem 2. *Theorem 1 and Cor. 1 also hold when $\mathcal{H}$ is a family of (ϵ,i)-AU hash functions and $|S| \geq i$.*

Proof. The proof is close to that of Theorem 1. The key observation is that whenever $|S| \geq i$, under an (ϵ,i)-AU hash family we obtain the same bound on the variance, σ^2, as in the ϵ-SU hash family case. □

5. Hashing with Low-Density (Sparse) Parity Constraints

Our main technical contribution is the following:

Theorem 3. *Let $A \in \{0,1\}^{m \times n}$ be a random matrix whose entries are Bernoulli i.i.d. random variables of parameter $f \leq 1/2$, i.e., $\Pr[A_{ij} = 1] = f$. Let $b \in \{0,1\}^m$ be chosen uniformly at random, independently from A. Let $w^* = \min\left\{w \mid \sum_{j=1}^{w} \binom{n}{j} \geq q\right\}$ and*

$$\epsilon(n,m,q,f) = \frac{1}{|S|-1} \sum_{w=1}^{w^*} \binom{n}{w} \left(\frac{1}{2} + \frac{1}{2}(1-2f)^w\right)^m$$

Then the family $\mathcal{H}^f = \{h_{A,b}(x) : \{0,1\}^n \to \{0,1\}^m\}$, where $h_{A,b}(x) = Ax + b \mod 2$ and $H \in \mathcal{H}^f$ is chosen randomly according to this process, is a family of $(\epsilon(n,m,q,f),q)$-AU hash functions.

Proof. Let $S \subseteq \{0,1\}^n$ and $y_1, y_2 \in \{0,1\}^m$. Then,

$$\begin{aligned}&\sum_{\substack{x_1,x_2 \in S \\ x_1 \neq x_2}} \Pr[H(x_1) = y_1, H(x_2) = y_2] \\ &= \sum_{\substack{x_1,x_2 \in S \\ x_1 \neq x_2}} \sum_{v \in \{0,1\}^m} \Pr\left[Ax_1 + v = y_1, Ax_2 + v = y_2\right] \Pr[b = v] \\ &= 2^{-m} \sum_{\substack{x_1,x_2 \in S \\ x_1 \neq x_2}} \sum_{v \in \{0,1\}^m} \Pr\left[Ax_1 + v = y_1, Ax_2 + v = y_2\right] \\ &= 2^{-m} \sum_{\substack{x_1,x_2 \in S \\ x_1 \neq x_2}} \Pr\left[A(x_1 - x_2) = y_1 - y_2\right]\end{aligned}$$

For brevity, let $\delta = y_1 - y_2$. The probability $\Pr\left[A(x_1 - x_2) = \delta\right]$ depends on the Hamming weight w

of $x_1 - x_2$, and is precisely the probability that the w columns of the (sparse) matrix A corresponding to the bits in which x_1 and x_2 differ sum to δ (mod 2).

In order to compute this probability, we use an analysis similar to MacKay (1999), based on treating the m random entries in each of the w columns of A as defining w steps of a *biased random walk* in each of the m dimensions of the m-dimensional Boolean hypercube. The probability that $A(x_1 - x_2)$ equals δ, when viewed this way, is nothing but the probability that starting from the origin and taking these w steps brings us to δ. Note that this is a function of only w, f, and δ; the exact columns in which x_1 and x_2 differ do not matter. Let us denote this probability $r^{(w,f)}(0,\delta)$.

Unlike MacKay (1999), each row of our matrix A is sampled independently. So we can model the random walk with m independent Markov Chains (one for each of the m dimensions) with two states $\{0,1\}$ and with transition probabilities

$$p_{0\to0} = 1-\alpha,\ p_{0\to1} = \alpha,\ p_{1\to1} = 1-\beta,\ p_{1\to0} = \beta.$$

Observing that the eigenvalues are 1 and $1-\alpha-\beta$, it is easy to verify that

$$\Pr[X_n = 0 \mid X_0 = 0] = \frac{\beta}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta}(1-\alpha-\beta)^n \quad (2)$$

and $\Pr[X_n = 1 \mid X_0 = 0] = 1 - \Pr[X_n = 0 \mid X_0 = 0]$. Setting $\alpha = \beta = f$ and $\delta = y_1 - y_2$ we get

$$\begin{aligned} r^{(w,f)}(0,\delta) &= \prod_{j=1}^{m}\left(\frac{1}{2} + (1-2\delta_j)\frac{1}{2}(1-2f)^w\right) \\ &\le \left(\frac{1}{2} + \frac{1}{2}(1-2f)^w\right)^m = r^{(w,f)}(0,0) \end{aligned}$$

because $f \le 1/2$. Plugging into the previous expression we get

$$\begin{aligned} &\sum_{\substack{x_1,x_2\in S\\ x_1\neq x_2}} \Pr[H(x_1)=y_1, H(x_2)=y_2] \\ &\qquad \le 2^{-m}\sum_{x_1\in S} h(w|x_1) r^{(w,f)}(0,0) \end{aligned}$$

where $h(w|x)$ is defined as the number of vectors in S that are at Hamming distance w from x. Clearly, $h(w|x) \le \binom{n}{w}$. Since $r^{(w,f)}(0,0)$ is monotonically decreasing in w, we can derive a worst case bound on the above expression by assuming all $\binom{n}{w}$ vectors at distance w from x are actually present in S for small w. Recalling the definition of w^* from the statement of the theorem, for all $|S| \le q$ this gives:

$$\begin{aligned} &\sum_{\substack{x_1,x_2\in S\\ x_1\neq x_2}} \Pr[H(x_1)=y_1, H(x_2)=y_2] \\ &\le 2^{-m}\sum_{x_1\in S}\sum_{w=1}^{w^*}\binom{n}{w} r^{(w,f)}(0,0) \\ &= 2^{-m}|S|\sum_{w=1}^{w^*}\binom{n}{w} r^{(w,f)}(0,0) = |S|(|S|-1)\frac{\epsilon(n,m,q,f)}{2^m} \end{aligned}$$

which proves that $\mathcal{H}$ is $(\epsilon(n,m,q,f),q)$-AU. □

The significance of this result is that given n, m, and q, we have a family of hash functions parameterized by f for which we can control the "average correlation" across all pairs of points in *any* set of size at least q. These range from rather dense but fully pairwise independent families when $f = 0.5$, to statistically useful but much sparser and hence much more tractable families when $f < 0.5$. Algorithms for probabilistic inference that use universal hashing often do not need full or even pairwise independence but only (μ^2,δ)-concentration for success with high probability. Section 4 thus prescribes a value of $\epsilon > \frac{1}{2^m}$ that suffices. Using the above theorem, we can therefore look for the smallest value of f that is compatible with the requirement. For example, using Thm. 3 and Cor. 1 applied to AU hash families as Thm. 2, we can look for the smallest value f^* such that the resulting family $\mathcal{H}^f = \{h_{A,b}(x) : \{0,1\}^n \to \{0,1\}^m\}$ guarantees (μ^2,δ)-concentration for sets of size at least 2^{m+2} and for $\delta = 9/4$.

In the top panel of Figure 1 we plot the corresponding $f^*(n)$ as a function of the number of variables n, for $m = n/2$ constraints. In the bottom panel, we plot $f^*(m)$ as a function of m, for $n = 100$. We see that we can obtain hash functions with provable concentration guarantees using constraints with average length scaling empirically as $n^{1-0.72}$ as opposed to $n/2$ for the pairwise independent construction. We also plot the smallest value of f that guarantees concentration when the set S in Definition 3 is not an arbitrary set of size $q = 2^{m+2}$ but is instead restricted to be an $(m+2)$-dimensional hypercube, for which we have an exact expression for $h(w|x)$. This is clearly a lower-bound for f^* (which is guaranteed to work for *any* set, hypercube included). The hypercube is intuitively close to a worst-case distribution for $h(w|x)$ because points have small average distance and hence high correlation. The comparison with the hypercube case highlights that our bounds are fairly tight.

5.1. Alternative Construction

We consider a related construction where we generate each row independently by fixing exactly t randomly chosen el-

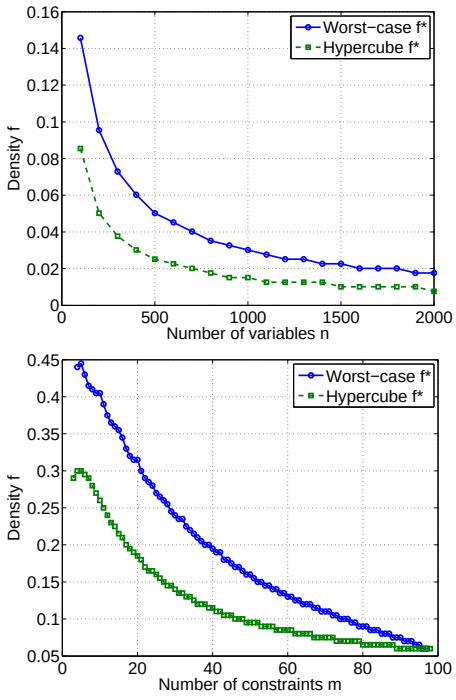

Figure 1. Numerical evaluation of the bound on the constraint density. Top: $m = n/2$, varying n. Bottom: $n = 100$, varying m.

ements to 1. In contrast, the previous construction has on average nf non-zero elements per row, but the number can vary. We can use an analysis similar to the one for Theorem 3, the main difference being that we need to substitute $r^{(w,f)}(0,0)$ with

$$z^{(w,f)}(0,0) = \left(\sum_{\text{even } \ell}^{\min(w,t)} \frac{\binom{w}{\ell}\binom{L-w}{t-\ell}}{\binom{L}{t}} \right)^m .$$

Then we can obtain a closed form expression for ϵ by again looking at the worst-case distribution of the neighbors in terms of Hamming distance w.

6. WISH With Sparse Parity Constraints

Our technique and analysis based on Average Universal hash functions applies to a range of probabilistic inference and counting techniques such as WISH (Ermon et al., 2013b;a), ApproxMC (Chakraborty et al., 2013b), and MBound (Gomes et al., 2006b). While preserving all their theoretical properties in terms of approximation guarantees, by substituting pairwise independent hash functions with *sparse* Average Universal ones we obtain significant improvements in terms of runtime (see experimental section below). For concreteness and brevity, we discuss its application to the recent WISH algorithm for discrete integration.

Algorithm 1 SPARSE-WISH $(w, n = \log_2 |\mathcal{X}|, \Delta, \alpha)$

$T \leftarrow \left\lceil \frac{\ln(1/\Delta)}{\alpha} \ln n \right\rceil$
for $i = 0, \cdots, n$ **do**
  for $t = 1, \cdots, T$ **do**
    $f^* = \min\{f | \epsilon(n, i, 2^{i+2}, f) < \frac{31}{5(2^{i+2}-1)}\}$
    Sample hash function $h^i_{A,b}$ from $\mathcal{H}^{f*}$
    i.e. sample sparse $A \in \{0,1\}^{i \times n}$, $b \in \{0,1\}^i$
    $w^t_i \leftarrow \max_\sigma w(\sigma)$ subject to $h^i_{A,b}(\sigma) = \mathbf{0}$
  end for
  $M_i \leftarrow \text{Median}(w^1_i, \cdots, w^T_i)$
end for
Return $M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i$

WISH is an algorithm used to estimate the partition function $Z = \sum_{\sigma \in \{0,1\}^n} w(\sigma) = \sum_{\sigma \in \mathcal{X}} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha)$ of a graphical model by solving a small number of MAP queries on the original model augmented with randomly generated parity constraints. WISH uses a universal hash function to partition the space of all possible variable assignments into 2^i cells, and then searches for the most likely variable assignment within a single cell. By varying the number of cells used to partition the state space, it estimates the tail of the weight distribution, finding its quantiles that can then be used to obtain a constant factor approximation of the intractable partition function within any desired degree of accuracy, with high probability and using only a polynomial number of MAP queries.

The original WISH (Ermon et al., 2013b) is based on pairwise independent hash functions, constructed using random parity constraints of average length $n/2$ for a problem with n binary variables. Our previous analysis allows us replace them with sparser constraints as in Theorem 3, obtaining an extension that we call SPARSE-WISH of which we provide the pseudocode as Algorithm 1. The key property is that Lemma 1 from Ermon et al. (2013b) still holds.

Lemma 1. *Fix an ordering $\sigma_i, 1 \le i \le 2^n$, of the configurations in $\{0,1\}^n$ such that for $1 \le j < 2^n$, $w(\sigma_j) \ge w(\sigma_{j+1})$. For $i \in \{0, 1, \cdots, n\}$, define $b_i \triangleq w(\sigma_{2^i})$. Let $M_i = \text{Median}(w^1_i, \cdots, w^T_i)$ be defined as in Algorithm 1. Then, for $0 < \alpha \le 0.0042$,*

$$\Pr\left[M_i \in [b_{\min\{i+2,n\}}, b_{\max\{i-2,0\}}]\right] \ge 1 - \exp(-\alpha T)$$

The proof is based on a variance argument. Intuitively, the hash functions used at iteration i are by Thm. 3 $(\epsilon, 2^{i+2})$-AU with ϵ chosen such that by Cor. 1 and Thm. 2 they guarantee weak $(\mu^2, 9/4)$-concentration for sets of size at least 2^{i+2}. In particular, this guarantees that the hash functions will behave nicely on the set formed by the 2^{i+2} heaviest configurations (no matter what is the structure of the set), and M_i will not underestimate b_i by too much. See

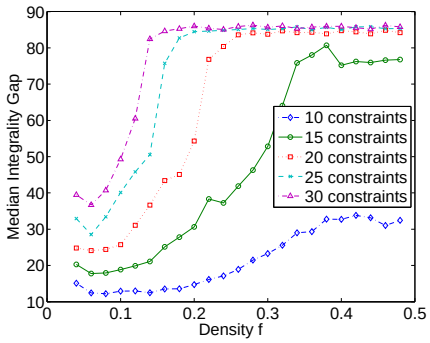

Figure 2. Integrality gap, $w = 2.5$, $M = 10$..

Appendix for a formal proof. We then have the following result analogous to Theorem 1 of Ermon et al. (2013b):

Theorem 4. *For any $\Delta > 0$, positive constant $\alpha \leq 0.0042$, and the hash families $\mathcal{H}^f$ given by Proposition 3, SPARSE-WISH makes $\Theta(n \ln n \ln 1/\delta)$ MAP queries and, with probability at least $(1 - \Delta)$, outputs a 16-approximation of $Z = \sum_{\sigma \in \{0,1\}^n} w(\sigma)$.*

This means that if we carefully choose the density f^* as in the pseudocode (which is a function of the number of constraints i, and in general much smaller than 0.5; cf. bottom panel of Fig 1), we maintain the same accuracy guarantees but using much sparser constraints. In contrast, the analysis of short XORs in Ermon et al. (2013a) only guaranteed that the output is an approximate lower bound for Z, not a constant factor approximation.

7. Experimental Evaluation

We evaluate SPARSE-WISH using the Integer Linear Programming (ILP) formulation from Ermon et al. (2013a) to solve the MAP inference instances in the inner loop of the algorithm. We use the Integer Programming solver CPLEX with a timeout of 10 minutes on Intel Xeon 5670 3GHz machines with 48GB RAM, obtaining at the end a lower bound and, by solving a sequence of LP relaxations, an upper bound on the optimization instances. These translate into bounds for the generally intractable partition function Z [1] (Ermon et al., 2013a) . We evaluate these bounds on $M \times M$ grid Ising models for $M \in \{10, 15\}$. In an Ising model, there are M^2 binary variables, with unary potentials $\psi_i(x_i) = \exp(f_i x_i)$ and (mixed) binary interactions $\psi_{ij}(x_i, x_j) = \exp(w_{ij} x_i x_j)$, where $w_{ij} \in_R [-w, w]$ and $f_i \in_R [-f, f]$. The external field is $f \in \{0.1, 1.0\}$.

In Figure 2 we show the median integrality gap[2] (over 500 runs) for the ILP formulation of the MAP inference instances, for $i \in \{10, 15, 20, 25, 30\}$ random parity constraints generated at various density levels f. We see that problems with short XORs (generated with small f) typically have smaller integrality gaps, which confirms the fact that short XORs are easier to reason about. This is not surprising, because the optimizations involved are analogous to max likelihood decoding problems, and sparse codes are known to be easier to decode empirically (MacKay, 1999).

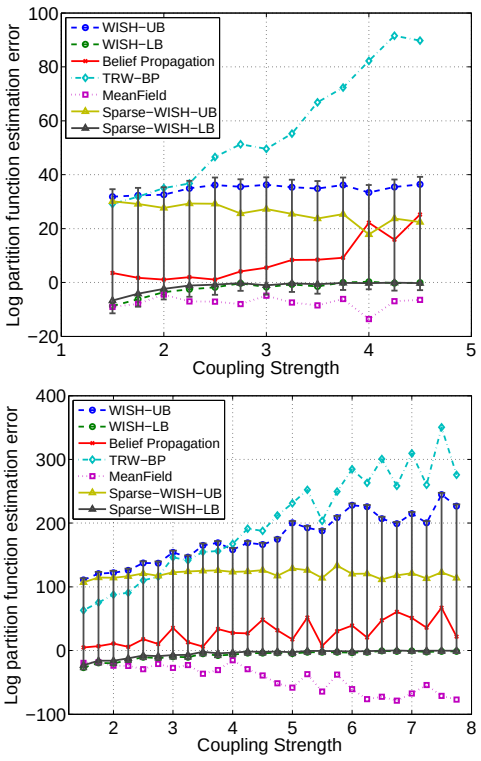

Figure 3. Results on Ising grids with mixed interactions with field 0.1. Top: 10×10 grid. Bottom: 15×15 grid.

We compare SPARSE-WISH with WISH (based on the same ILP formulation, but with denser $f = 0.5$ constraints), with Loopy BP (Murphy et al., 1999) which estimates Z without providing any accuracy guarantee, Tree Reweighted BP (Wainwright, 2003) which gives a provable upper bound, and Mean Field (Wainwright & Jordan, 2008) which gives a provable lower bound. We use the implementations in the LibDAI library (Mooij, 2010), allowing 1000 random restarts for Mean Field. Figure 3 shows the error in the resulting estimates, where ground truth is from Junction Trees (Lauritzen & Spiegelhalter, 1988).

We see that SPARSE-WISH provides significantly better bounds compared to the original WISH algorithm. Since they are both run for the same amount of time using the same combinatorial optimization suite, the improvement is to be attributed entirely to the sparser constraints employed

[1] When all the ILPs are solved to optimality, upper and lower bounds match and the value is guaranteed to be a constant factor approximation for Z.

[2] At the root node of the ILP solver search tree.

by SPARSE-WISH, which are easier to reason about. Intuitively, as seen in Figure 2 the LP relaxation obtained using shorter XORs is much tighter, hence improving the quality of the bounds, and yielding overall the best provable upper and lower bounds among all algorithms we considered. Notice the improvement in terms of lower bound is smaller because in both cases the bounds are quite tight (with an error close to 0). Remarkably, SPARSE-WISH is the only method that does not deteriorate as the coupling strength is increased. We emphasize that the improvement over WISH comes at no cost, because thanks to our carefully chosen density thresholds f^*, we maintain the same theoretical properties without trading off accuracy for speed.

7.1. Model counting for SAT

Long parity constraints are difficult to reason about not only for Integer Programming solvers but also for SAT solvers. In fact, SAT solvers can be substantially faster on sparser parity constraints than those of length $n/2$ (Gomes et al., 2006b).

The use of short parity constraints for model counting (i.e., count the number of solutions of a SAT instance) was investigated by Gomes et al. (2007a), where it was empirically shown that short XORs perform well on a wide variety of instances a number of problem domains. Our analysis provides the first theoretical basis for this observed empirical phenomenon, while also providing a principled way to estimate a priori a suitable length of parity constraints to use.

Table 1 reports the bounds obtained with our analysis on the benchmark used by Gomes et al. (2007a). The best previously known theoretical bound on the length was $n/2$, based on the pairwise independent construction (Prop. 1). The best previously known empirical bound was computed by finding the smallest XOR length such that the variance of the resulting model count estimate is the same as what one would obtain with pairwise independent functions, which can be easily computed analytically. The new provable bound is computed by looking for the shortest XOR length that gives (μ^2, δ)-concentration and therefore provides a "correct" answer more than half the time (as in Prop. 2). Specifically, by Theorem 2, we look for the minimum XOR length satisfying the weak $(\mu^2, 9/4)$-concentration condition given by Corollary 1, where the number of variables (n), the log of the set size ($\log_2 |S|$), and the number of XORs (m) are taken from Gomes et al. (2007a) and reported in the first three columns of Table 1. The new empirical bound is also based on the shortest XOR length yielding weak $(\mu^2, 9/4)$-concentration, but using Prop. 3 for general hash families and taking the sample variance as a proxy for the true variance, σ^2.

On this diverse benchmark spanning a variety of domains (Latin square completion, logistic planning, hardware verification, random, and synthetic), the new theoretical bound on the minimum XOR length is significantly smaller than $n/2$. The empirical bound we achieve (last column, often in single digits and thus extremely efficient for SAT solvers) is also much smaller than the previously reported empirical bound. The gap between our new provable and empirical bounds is due to the intricate structure of the set S of solutions. If the solutions in S are far away on average (hence less correlated when using sparse parity constraints), we obtain an empirical variance that is much tighter than our provable worst-case bound. Overall, our new bounds significantly improve upon the previous best known bounds in all cases considered.

Table 1. Minimum size of XOR constraints suitable for model counting for SAT

Instance				Provable Bounds		Empirical Bounds	
Name	Num Vars	$\log_2$ Solns	Num XORs	Old	New	Old	New
ls7R34med	119	10	7	59	46	6	3
ls7R35med	136	12	9	68	53	7	3
ls7R36med	149	14	11	74	56	7	3
log.c.red	352	19	10	176	112	98	28
2bitmax_6	252	97	87	126	26	–	8
wff-3-100-330	100	32	25	50	21	17	7
wff-3-100-380	100	22	15	50	27	26	8
wff-3-100-396	100	18	11	50	29	38	10
string-50-30	50	30	20	25	8	11	4
string-50-40	50	40	30	25	5	>10	4
string-50-49	50	49	39	25	3	6	3
blk-50-3-10-20	50	23	13	25	10	>15	5
blk-50-6-5-20	50	26	16	25	9	15	4
blk-50-10-3-20	50	30	20	25	8	15	3

8. Conclusion

We introduced a new class of hash functions, called Average Universal, that can be constructed using low-density (sparse) parity constraints. While statistically weaker than traditional hash functions, these are still powerful enough to be used in hashing-based randomized counting and discrete integration schemes. Sparse parity constraints are empirically much easier to do inference with, a well-known fact in the context of low-density parity check codes. By substituting dense parity constraints with sparser ones, we obtain variations of inference and counting techniques that have the same provable guarantees but are empirically much more tractable. We show that this leads to significant improvements in the bounds obtained by the recent WISH algorithm and in model counting applications.

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

References

Bellare, M., Goldreich, O., and Petrank, E. Uniform generation of NP-witnesses using an NP-oracle. *Information and Computation*, 163(2):510–526, 2000.

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

Chakraborty, S., Meel, K., and Vardi, M. A scalable and nearly uniform generator of SAT witnesses. In *CAV*, 2013a.

Chakraborty, S., Meel, K. S., and Vardi, M. Y. A scalable approximate model counter. In *Principles and Practice of Constraint Programming*, pp. 200–216. Springer, 2013b.

Ermon, S., Gomes, C., Sabharwal, A., and Selman, B. Optimization with parity constraints: From binary codes to discrete integration. In *UAI*, 2013a.

Ermon, S., Gomes, C., Sabharwal, A., and Selman, B. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *ICML*, 2013b.

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

Gomes, C. P., Hoffmann, J., Sabharwal, A., and Selman, B. Short XORs for model counting; from theory to practice. In *10th SAT*, volume 4501 of *LNCS*, pp. 100–106, Lisbon, Portugal, May 2007a.

Gomes, C. P., van Hoeve, W. J., Sabharwal, A., and Selman, B. Counting CSP solutions using generalized XOR constraints. In *AAAI*, 2007b.

Gomes, C., Sabharwal, A., and Selman, B. Near-uniform sampling of combinatorial spaces using XOR constraints. *Advances In Neural Information Processing Systems*, 19:481–488, 2006a.

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

Hazan, T., Maji, S., and Jaakkola, T. On sampling from the Gibbs distribution with random maximum a-posteriori perturbations. In *NIPS*, pp. 1268–1276, 2013.

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

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

MacKay, D. J. Good error-correcting codes based on very sparse matrices. *Information Theory, IEEE Transactions on*, 45(2): 399–431, 1999.

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

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

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

Papandreou, G. and Yuille, A. L. Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In *ICCV*, pp. 193–200, 2011.

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

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

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

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

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

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