Low-Density Parity Constraints for Hashing-Based Discrete Integration

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

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

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

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-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 trade-off between statistical properties and computational efficiency of combinatorial reasoning with such constraints. This idea applies to a wide range of probabilistic inference methods all building upon the original theoretical ideas by Valiant & Vazirani (1986); Bellare et al. (2000); Sipser (1983).

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 \( \epsilon \)-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) \( \forall x_1, x_2 \in \{0,1\}^n \), \( x_1 \neq x_2 \), 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. Preliminaries

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

**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.
ever, 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. \((k, \delta)\)-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:

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

Let \( \mu(h, S, y) = E[X(h, S, y)] \) 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 \( \mu \) when \( h, S, \) and \( y \) are implicit in the context. Our main interest is in understanding how concentrated \( X \) is around \( \mu \), under various choices of \( \mathcal{H} \). We introduce a general notion of concentration that will come handy:

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

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

As an illustrative example of how \((k, \delta)\)-concentration can be used for estimating sizes of high-dimensional sets, consider the following simple randomized algorithm \( A \) for approximately computing \( |S| \) with high probability. Let \( \mathcal{H}^i = \{h : \{0,1\}^n \to \{0,1\}^1\} \) for \( i = 1, 2, \ldots, n \) be universal families of hash functions. For \( i \) increasing from 1 to \( n \), compute “is the set \( h_i^{-1}(0) \cap S \) empty” \( T \) times, each time with a different hash function \( h_i \) 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 \( \mathcal{H}^i = \{h : \{0,1\}^n \to \{0,1\}^1\}, i \in \{1, 2, \ldots, n\} \) are universal families of hash functions such that \( X(h, S, y) \) is weakly \((\mu^2, 4)\)-concentrated, then \( A \) 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 \((k, \delta)\)-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 \( \delta > 0 \) and \( \mathcal{H} \) be a family of hash functions. For any \( S \subseteq \{0,1\}^n \) and \( y \in \{0,1\}^m \), and for \( h \in \mathcal{H} \), \( X(h, S, y) \) is strongly \((\delta \sigma^2, \delta)\)-concentrated and weakly \((\delta^2 - \delta \sigma^2, \delta)\)-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 \( h \in \mathcal{H} \), \( X(h, S, y) \) 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 \( h \in \mathcal{H} \), \( X(h, S, y) \) is strongly \((\delta \mu, \delta)\)-concentrated and weakly \((\delta - 1)\mu, \delta)\)-concentrated.

This follows from observing that pairwise independence implies \( \sigma^2 = |S|/2^m(1 - 1/2^n) < \mu \) 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 \( \epsilon \)-SU hash functions to explore this wide spectrum of possible concentrations by varying \( \epsilon \).

**Theorem 1.** Let \( \delta > 2, \epsilon \geq 1/2^m \), and \( \mathcal{H} \) be a family of \( \epsilon \)-SU hash functions. For any \( S \subseteq \{0,1\}^n \) and \( y \in \{0,1\}^m \),

\[
\Pr[|X(h, S, y) - \mu| > \epsilon \mu] < \frac{1}{\epsilon^2}.
\]
and for \( h \in \mathcal{H} \), \( X(h, S, y) \) 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.

\[
E[X(h, S, y)^2] = \sum_{s, s' \in S} E[1_{h(s) = y, h(s') = y}]
\]

\[
= \sum_{s \in S} E[1_{h(s) = y}] + \sum_{s \neq s'} E[1_{h(s) = y, h(s') = y}]
\]

\[
\leq \mu + |S|(1/2 - 1)\epsilon - |S| \leq \mu + |S|(1/2 - 1)\epsilon/2^m
\]

(from SU)

\[
= \mu(1 + \epsilon(|S| - 1)).
\]

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

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

Thus, by increasing \( \epsilon \), we can achieve lower (but still acceptable) levels of concentration of \( X \). In practice, however, it is not easy to construct \( \epsilon\)-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 \( \epsilon\)-SU (namely, the guarantees in Theorem 1) but are computationally much more tractable for inference methods.

Definition 4. A family of hash functions \( \mathcal{H} = \{h : \{0, 1\}^n \rightarrow \{0, 1\}^m\} \) is \((\epsilon, i)\)-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 \( \epsilon\)-SU. (b) If \( \mathcal{H} \) is \( \epsilon\)-SU, then \( \mathcal{H} \) is also \((\epsilon, i)\)-AU for all \( i \geq 2 \). (c) If \( \mathcal{H} \) is \((\epsilon, i)\)-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 \((\epsilon, i)\)-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 \((\epsilon, i)\)-AU hash family we obtain the same bound on the variance, \( \sigma^2 \), as in the \( \epsilon\)-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 \( 0 \leq q \leq 2^m \), \( w^* = \max \left\{ w \mid \sum_{i=1}^w \binom{n}{w} \left( \frac{1}{2} + \frac{1}{2} \left( 1 - 2f \right)^w \right)^m \right\} \) and

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

Then the family \( \mathcal{H}^f = \{h_{A,b}(x) : \{0, 1\}^n \rightarrow \{0, 1\}^m\} \), 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,

\[
\sum_{x_1, x_2 \in S \atop x_1 \neq x_2} \Pr[H(x_1) = y_1, H(x_2) = y_2]
\]

\[
= \sum_{x_1, x_2 \in S \atop x_1 \neq x_2} \sum_{v \in \{0, 1\}^m} \Pr[Ax_1 + v = y_1, Ax_2 + v = y_2] \Pr[b = v]
\]

\[
= 2^{-m} \sum_{x_1, x_2 \in S \atop x_1 \neq x_2} \sum_{v \in \{0, 1\}^m} \Pr[Ax_1 + v = y_1, Ax_2 + v = y_2]
\]

\[
= 2^{-m} \sum_{x_1, x_2 \in S \atop x_1 \neq x_2} \Pr[A(x_1 - x_2) = y_1 - y_2]
\]

For brevity, let \( \delta = y_1 - y_2 \). The probability \( \Pr[A(x_1 - x_2) = \delta] \) 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 \( \delta \) (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 \( A(x_1 - x_2) = \delta \), when viewed this way, is nothing but the probability that starting from the origin and taking these \( w \) steps brings us to \( \delta \). Note that this is a function of only \( w \), \( f \), and \( \delta \); 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

\[
p_{0 \to 0} = 1 - \alpha, \quad p_{0 \to 1} = \alpha, \quad p_{1 \to 1} = 1 - \beta, \quad p_{1 \to 0} = \beta.
\]

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

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

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

\[
r^{(w,f)}(0, \delta) = \prod_{j=1}^{m} \left( \frac{1}{2} + (1 - 2f) \right)^{-\frac{n}{w}}
\]

\[
\leq \left( \frac{1}{2} + (1 - 2f) \right)^{m} = r^{(w,f)}(0, 0)
\]

because \( f \leq 1/2 \). Plugging into the previous expression we get

\[
\sum_{x_1, x_2 \in S} Pr[H(x_1) = y_1, H(x_2) = y_2]
\]

\[
\leq 2^{-m} \sum_{x_1 \in S} \sum_{w=1}^{n} h(w|x_1)r^{(w,f)}(0, 0)
\]

where \( h(w|x) \) is defined as the number of vectors in \( S \) that are at Hamming distance \( w \) from \( x \). Clearly, \( h(w|x) \leq \binom{n}{w} \). 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 and letting \( q^* = q - 1 - \sum_{w=1}^{w^*} \binom{n}{w} \), for all \( |S| \leq q \) this gives:

\[
\sum_{x_1, x_2 \in S} Pr[H(x_1) = y_1, H(x_2) = y_2]
\]

\[
\leq 2^{-m} \sum_{x_1 \in S} \sum_{w=1}^{w^*} h(w|x_1)r^{(w,f)}(0, 0)
\]

\[
= 2^{-m}|S| \binom{n}{w^*}r^{(w^*,f)}(0, 0) + q^*r^{(w^*+1,f)}(0, 0)
\]

which proves that \( H \) 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 \((\mu^2, \delta)\)-concentration for success with high probability. Section 4 thus prescribes a value of \( \epsilon \geq \frac{1}{2m} \) 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 \( \mathcal{H} = \{ h_{A,b}(x) : \{0, 1\}^n \rightarrow \{0, 1\}^m \} \) guarantees \((\mu^2, \delta)\)-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 elements to 1. In contrast, the previous construction has on average \( n/f \) 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 \( f(w,f)(0,0) \) with

\[
(\sum_{\ell}^{w} \frac{\ell^{(L-w)}}{L^{\ell}})^{m}
\]

Then we can obtain a closed form expression for \( \epsilon \) 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.

WISH is an algorithm used to estimate the partition function \( Z = \sum_{\sigma \in \{0,1\}^n} w(\sigma) = \sum_{\sigma \in \mathcal{X}} \prod_{x \in \mathcal{T}} \psi_x(\{x\},\alpha) \) 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 \( 2^l \) 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 \( \sigma_i, 1 \leq i \leq 2^n \), of the configurations in \( \{0,1\}^n \) such that for \( 1 \leq j < 2^n \), \( w(\sigma_j) \geq w(\sigma_{j+1}) \). For \( i \in \{0,1,\ldots,n\} \), define \( b_i \triangleq w(\sigma_{2^i}) \). Let \( M_i = \text{Median}(w_1^i, \ldots, w_{2^i}) \) be defined as in Algorithm 1. Then, for \( 0 < \alpha \leq 0.0042 \),

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

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 \( \epsilon \) 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 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 $\mathcal{H}^i$ 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 $Z$ \footnote{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 $Z.$} (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 $w_{ij} \in \mathbb{R} [-w, w]$ and $f_i \in \mathbb{R} [-f, f]$. The external field is $f \in \{0, 1, 1.0\}$.

In Figure 2 we show the median integrality gap\footnote{At the root node of the ILP solver search tree.} (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).

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 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 \( n/2 \). 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 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.

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.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Num Vars</th>
<th>Num Solns</th>
<th>Provable Bounds</th>
<th>Empirical Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>ls/R34med</td>
<td>119</td>
<td>10</td>
<td>7</td>
<td>59</td>
</tr>
<tr>
<td>ls/R35med</td>
<td>136</td>
<td>12</td>
<td>9</td>
<td>68</td>
</tr>
<tr>
<td>ls/R36med</td>
<td>149</td>
<td>14</td>
<td>11</td>
<td>74</td>
</tr>
<tr>
<td>log.c.red</td>
<td>352</td>
<td>19</td>
<td>10</td>
<td>176</td>
</tr>
<tr>
<td>2bitmax,6</td>
<td>252</td>
<td>97</td>
<td>87</td>
<td>126</td>
</tr>
<tr>
<td>wff-3-100-330</td>
<td>100</td>
<td>32</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>wff-3-100-380</td>
<td>100</td>
<td>22</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>wff-3-100-396</td>
<td>100</td>
<td>18</td>
<td>11</td>
<td>50</td>
</tr>
<tr>
<td>string-50-30</td>
<td>50</td>
<td>30</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>string-50-40</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>string-50-49</td>
<td>50</td>
<td>49</td>
<td>39</td>
<td>25</td>
</tr>
<tr>
<td>blk-50-3-10-20</td>
<td>50</td>
<td>23</td>
<td>13</td>
<td>25</td>
</tr>
<tr>
<td>blk-50-6-5-20</td>
<td>50</td>
<td>26</td>
<td>16</td>
<td>25</td>
</tr>
<tr>
<td>blk-50-10-3-20</td>
<td>50</td>
<td>30</td>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 1. Minimum size of XOR constraints suitable for model counting for SAT.
Acknowledgments: Research supported by NSF grants #0832782 and #1059284.

References


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


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


A. Appendix

A.1. Proofs

**Proof of Proposition 2.** Let \( i^* = \log_2 |S| \) and \( 2^{i^*-1} \) be the output of \( A \). We will show that \( i \) is within \([i^*, i^* + 2]\) with probability at least 3/4.

Fix any \( i \leq i^* \). Then \( \mathbb{E} [|h^{-1}(0) \cap S|] = |S|/2^i \geq 1 \). Weak \((\mu^2, 4)\)-concentration implies that \( \Pr[|h^{-1}(0) \cap S| = 0] \leq 1/4 \). Chernoff bound applied to the \( T \) underlying independent 0-1 indicator random variables then implies that a majority of the \( T \) sets will be empty with probability at most \( \exp(-T/8) \). It follows that with probability at least \((1 - \exp(-T/8))^{T} \geq (1 - \exp(-T/8))^n \geq 1 - n \exp(-T/8) \), the majority of the \( T \) sets for all \( i \leq i^* \) will simultaneously be non-empty. Thus, for \( T \geq 8 \ln(8n) \), we have that with probability at least 7/8, all \( i \leq i^* \) will behave correctly.

Fix any \( i \geq i^* + 2 \). Here we can simply use Markov’s inequality to infer that \( \Pr[|h^{-1}(0) \cap S|] \geq 1 \leq 1/4 \). From the same Chernoff bound based argument as above, it follows that for \( T \geq 8 \ln(8n) \), with probability at least 7/8, all \( i \geq i^* + 2 \) will behave correctly.

By union bound, it follows that the output \( 2^{i^*-1} \) of \( A \) will be in the range \([2^{i^*-1} - 1, 2^{i^*-1} + 1]\) with probability at least \( 1 - 1/8 - 1/8 = 3/4 \). \( \square \)

**Proof of Proposition 3.** From Chebychev’s inequality, \( \Pr[X - \mu] \geq \sqrt{\delta \sigma} \leq \delta \), which implies the desired weak correlation. For showing the desired weak correlation, we use Cantelli’s one-sided inequalities. For the first case, \( \Pr[X \leq \mu - \sqrt{\delta - 1} \sigma] \leq 1/(1 + (\delta - 1)) = 1/\delta \). The second case works similarly. \( \square \)

**Proof of Proposition 4.** From Chernoff’s bound, \( \Pr[X \leq \mu + \sqrt{k}] = \Pr[X \leq (1 + \sqrt{k})\mu] \leq \exp(-k/2^m) \). Thus, \( k \geq (3 \ln \delta) \mu \) suffices to bound this probability by \( 1/\delta \). The other side, \( \Pr[X \leq \mu - \sqrt{k}] \), similarly leads to \( k \geq (2 \ln \delta) \mu \) as the condition to bound the probability by \( 1/\delta \). Combining the two, we get the desired result for weak concentration. The result for strong concentration follows by using the union bound to obtain \( \exp(-k/2^m) + \exp(-k/2^m) \), which is less than \( \exp(-k/4^m) \) for any \( c > 3 \). \( \square \)

**Proof of Proposition 5.** This follows from observing that pairwise independence implies \( \sigma^2 = |S|/2^m (1 - 1/2^m) < \mu \) and then applying Prop. 3. \( \square \)

**Proof of Proposition 6.** The first two observations are straightforward. For the third, let \( S \) and \( T \) be sets with \( |T| = |S| + 1 \). Given \( y_1, y_2 \in \{0, 1\}^m \), let \( f(x_1, x_2) \) de-
A.2. Additional Experiments

We report additional experimental results for mixed interaction Ising grids in Figure 4 with the same setup as in Section 7 but with external field 1.0 rather than 0.1.

Figure 4. Results on Ising grids with mixed interactions. Top: Mixed 10 × 10. Field 1.0. Bottom: Mixed 15 × 15. Field 1.0.