

# Closing the Gap Between Short and Long XORs for Model Counting

**Shengjia Zhao**

Computer Science Dept.  
Tsinghua University  
zhaosj12@mails.tsinghua.edu.cn

**Sorathan Chaturapruek**

Computer Science Dept.  
Stanford University  
sorathan@cs.stanford.edu

**Ashish Sabharwal**

Allen Institute for AI  
Seattle, WA  
ashishs@allenai.org

**Stefano Ermon**

Computer Science Dept.  
Stanford University  
ermon@cs.stanford.edu

## Abstract

Many recent algorithms for approximate model counting are based on a reduction to combinatorial searches over random subsets of the space defined by parity or XOR constraints. Long parity constraints (involving many variables) provide strong theoretical guarantees but are computationally difficult. Short parity constraints are easier to solve but have weaker statistical properties. It is currently not known how long these parity constraints need to be. We close the gap by providing matching necessary and sufficient conditions on the required asymptotic length of the parity constraints. Further, we provide a new family of lower bounds and the first non-trivial upper bounds on the model count that are valid for arbitrarily short XORs. We empirically demonstrate the effectiveness of these bounds on model counting benchmarks and in a Satisfiability Modulo Theory (SMT) application motivated by the analysis of contingency tables in statistics.

## Introduction

Model counting is the problem of computing the number of distinct solutions of a given Boolean formula. It is a classical problem that has received considerable attention from a theoretical point of view (Valiant 1979b; Stockmeyer 1985), as well as from a practical perspective (Sang et al. 2004; Gogate and Dechter 2007). Numerous probabilistic inference and decision making tasks, in fact, can be directly translated to (weighted) model counting problems (Richardson and Domingos 2006; Gogate and Domingos 2011). As a generalization of satisfiability testing, the problem is clearly intractable in the worst case. Nevertheless, there has been considerable success in both exact and approximate model counting algorithms, motivated by a number of applications (Sang, Beame, and Kautz 2005).

Recently, approximate model counting techniques based on randomized hashing have emerged as one of the leading approaches (Gomes, Sabharwal, and Selman 2006; Chakraborty, Meel, and Vardi 2013a; Ermon et al. 2014; Ivrii et al. 2015; Achlioptas and Jiang 2015; Belle, Van den Broeck, and Passerini 2015). While approximate, these techniques provide strong guarantees on the accuracy of the results in a probabilistic sense. Further, these methods all reduce model counting to a small number of combinatorial

searches on a randomly projected version of the original formula, obtained by augmenting it with randomly generated parity or XOR constraints. This approach allows one to leverage decades of research and engineering in combinatorial reasoning technology, such as fast satisfiability (SAT) and SMT solvers (Biere et al. 2009).

While modern solvers have witnessed tremendous progress over the past 25 years, model counting techniques based on hashing tend to produce instances that are difficult to solve. In order to achieve strong (probabilistic) accuracy guarantees, existing techniques require each randomly generated parity constraint to be relatively long, involving roughly half of the variables in the original problem. Such constraints, while easily solved in isolation using Gaussian Elimination, are notoriously difficult to handle when conjoined with the original formula (Gomes et al. 2007; Ermon et al. 2014; Ivrii et al. 2015; Achlioptas and Jiang 2015). Shorter parity constraints, i.e., those involving a relative few variables, are friendlier to SAT solvers, but their statistical properties are not well understood.

Ermon et al. (2014) showed that long parity constraints are not strictly necessary, and that one can obtain the *same accuracy guarantees* using shorter XORs, which are computationally much more friendly. They provided a closed form expression, allowing an easy computation of an XOR length that suffices, given various parameters such as the number of problem variables, the number of constraints being added, and the size of the solution space under consideration. It is, however, currently not known how tight their sufficiency condition is, how it scales with various parameters, or whether it is in fact a necessary condition.

*We resolve these open questions by providing an analysis of the optimal asymptotic constraint length* required for obtaining high-confidence approximations to the model count. Surprisingly, for formulas with  $n$  variables, we find that when  $\Theta(n)$  constraints are added, a constraint length of  $\Theta(\log n)$  is both necessary and sufficient. This is a significant improvement over standard long XORs, which have length  $\Theta(n)$ . Constraints of logarithmic length can, for instance, be encoded efficiently with a polynomial number of clauses. We also study how the sufficient constraint length evolves from  $O(\log n)$  to  $O(n^\gamma \log^2 n)$  to  $n/2$  across various regimes of the number of parity constraints.

As a byproduct of our analysis, we obtain a *new fam-*

ily of probabilistic upper and lower bounds that are valid regardless of the constraint length used. These upper and lower bounds on the model count reach within a constant factor of each other as the constraint density approaches the aforementioned optimal value. The bounds gracefully degrade as we reduce the constraint length and the corresponding computational budget. While lower bounds for arbitrary XOR lengths were previously known (Gomes et al. 2007; Ermon et al. 2013a), this is the first non-trivial upper bound in this setting. Remarkably, even though we rely on random projections and therefore only look at subsets of the entire space (a *local* view, akin to traditional sampling), we are able to say something about the *global* nature of the space, i.e., a probabilistic upper bound on the number of solutions.

We evaluate these new bounds on standard model counting benchmarks and on a new counting application arising from the analysis of contingency tables is statistics. These data sets are common in many scientific domains, from sociological studies to ecology (Sheldon and Dietterich 2011). We provide a new approach based on SMT solvers and a bit-vector arithmetic encoding. Our approach scales very well and produces accurate results on a wide range of benchmarks. It can also handle additional constraints on the tables, which are very common in scientific data analysis problems, where prior domain knowledge translates into constraints on the tables (e.g., certain entry must be zero because the corresponding event is known to be impossible). We demonstrate the capability to handle structural zeroes (Chen 2007) in real experimental data.

## Preliminaries: Counting by Hashing

Let  $x_1, \dots, x_n$  be  $n$  Boolean variables. Let  $S \subseteq \{0, 1\}^n$  be a large, high-dimensional set<sup>1</sup>. We are interested in computing  $|S|$ , the number of elements in  $S$ , when  $S$  is defined succinctly through conditions or constraints that the elements of  $S$  satisfy and membership in  $S$  can be tested using an NP oracle. For example, when  $S$  is the set of solutions of a Boolean formula over  $n$  binary variables, the problem of computing  $|S|$  is known as model counting, which is the canonical  $\#\text{-P}$  complete problem (Valiant 1979b).

In the past few years, there has been growing interest in approximate probabilistic algorithms for model counting. It has been shown (Gomes, Sabharwal, and Selman 2006; Ermon et al. 2013b; Chakraborty, Meel, and Vardi 2013a; Achlioptas and Jiang 2015; Belle, Van den Broeck, and Passerini 2015) that one can reliably estimate  $|S|$ , both in theory and in practice, by repeating the following simple process: randomly partition  $S$  into  $2^m$  cells and select one of these lower-dimensional cells, and compute whether  $S$  has at least 1 element in this cell (this can be accomplished with a query to an NP oracle, e.g., invoking a SAT solver). Somewhat surprisingly, repeating this procedure a small number of times provides a constant factor approximation to  $|S|$  with high probability, even though counting problems (in  $\#\text{-P}$ ) are believed to be significantly harder than decision problems (in NP).

<sup>1</sup>We restrict ourselves to the binary case for the ease of exposition. Our work can be naturally extended to categorical variables.

The correctness of the approach relies crucially on how the space is randomly partitioned into cells. All existing approaches partition the space into cells using parity or XOR constraints. A parity constraint defined on a subset of variables checks whether an odd or even number of the variables take the value 1. Specifically,  $m$  parity (or XOR) constraints are generated, and  $S$  is partitioned into  $2^m$  equivalence classes based on which parity constraints are satisfied.

The way in which these constraints are generated affects the quality of the approximation of  $|S|$  (the model count) obtained. Most methods *randomly* generate  $m$  parity constraints by adding each variable to each constraint with probability  $f \leq 1/2$ . This construction can also be interpreted as defining a hash function, mapping the space  $\{0, 1\}^n$  into  $2^m$  hash bins (cells). Formally,

**Definition 1.** 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  $A$ . Then,  $\mathcal{H}_{m \times n}^f = \{h_{A,b} : \{0, 1\}^n \rightarrow \{0, 1\}^m\}$ , where  $h_{A,b}(x) = Ax + b \pmod 2$  and  $h_{A,b} \in_R \mathcal{H}_{m \times n}^f$  is chosen randomly according to this process, is a family of  $f$ -sparse hash functions.

The idea to estimate  $|S|$  is to define progressively smaller cells (by increasing  $m$ , the number of parity constraints used to define  $h$ ), until the cells become so small that no element of  $S$  can be found inside a (randomly) chosen cell. The intuition is that the larger  $|S|$  is, the smaller the cells will have to be, and we can use this information to estimate  $|S|$ .

Based on this intuition, we give a hashing-based counting procedure (Algorithm 1, SPARSE-COUNT), which relies on an NP oracle  $\mathcal{O}_S$  to check whether  $S$  has an element in the cell. It is adapted from the SPARSE-WISH algorithm of Ermon et al. (2014). The algorithm takes as input  $n$  families of  $f$ -sparse hash functions  $\{\mathcal{H}_{i \times n}^{f_i}\}_{i=0}^n$ , used to partition the space into cells. In practice, line 7 is implemented using a SAT solver as an NP-oracle. In a model counting application, this is accomplished by adding to the original formula  $i$  parity constraints generated as in Definition 1 and checking the satisfiability of the augmented formula.

Typically,  $\{\mathcal{H}_{i \times n}^{1/2}\}$  is used, corresponding to XORs where each variable is added with probability  $1/2$  (hence with average length  $n/2$ ). We call these **long parity constraints**. In this case, it can be shown that SPARSE-COUNT will output a factor 16 approximation of  $|S|$  with probability at least  $1 - \Delta$  (Ermon et al. 2014). Unfortunately, checking satisfiability (i.e.,  $S(h_{A,b}^i) \geq 1$ , line 7) has been observed to be very difficult when many long parity constraints are added to a formula (Gomes et al. 2007; Ermon et al. 2014; Ivrii et al. 2015; Achlioptas and Jiang 2015). Note, for instance, that while a parity constraint of length one simply freezes a variable right away, a parity constraint of length  $k$  can be propagated only after  $k - 1$  variables have been set. From a theoretical perspective, parity constraints are known to be fundamentally difficult for the resolution proof system underlying SAT solvers (cf. exponential scaling of Tseitin tautologies (Tseitin 1968)). A natural question, therefore, is

---

**Algorithm 1** SPARSE-COUNT ( $\mathcal{O}_S, \Delta, \alpha, \{\mathcal{H}_{i \times n}^{f_i}\}_{i=0}^n$ )

---

```

1:  $T \leftarrow \left\lceil \frac{\ln(1/\Delta)}{\alpha} \ln n \right\rceil$ 
2:  $i = 0$ 
3: while  $i \leq n$  do
4:   for  $t = 1, \dots, T$  do
5:      $h_{A,b}^i \leftarrow$  hash function sampled from  $\mathcal{H}_{i \times n}^{f_i}$ 
6:     Let  $S(h_{A,b}^i) = |\{x \in S \mid h_{A,b}^i(x) = 0\}|$ 
7:      $w_i^t \leftarrow \mathbb{I}[S(h_{A,b}^i) \geq 1]$ , invoking  $\mathcal{O}_S$ 
8:   end for
9:   if Median( $w_i^1, \dots, w_i^T$ )  $< 1$  then
10:     break
11:   end if
12:    $i = i + 1$ 
13: end while
14: Return  $\lfloor 2^{i-1} \rfloor$ 

```

---

whether **short parity constraints** can be used in SPARSE-COUNT and provide reliable bounds for  $|S|$ .

Intuitively, for the method to work we want the hash functions  $\{\mathcal{H}_{i \times n}^{f_i}\}$  to have a small collision probability. In other words, we want to ensure that when we partition the space into cells, configurations from  $S$  are divided into cells evenly. This gives a direct relationship between the original number of solutions  $|S|$  and the (random) number of solutions in one (randomly chosen) cell,  $S(h)$ . More precisely, we say that the hash family *shatters*  $S$  if the following holds:

**Definition 2.** For  $\epsilon > 0$ , a family of hash functions  $\mathcal{H}_{i \times n}^f$   $\epsilon$ -shatters a set  $S$  if  $\Pr[S(h) \geq 1] \geq 1/2 + \epsilon$  when  $h \in_R \mathcal{H}_{i \times n}^f$ , where  $S(h) = |\{x \in S \mid h(x) = 0\}|$ .

The crucial property we need to obtain reliable estimates is that the hash functions (equivalently, parity constraints) are able to shatter sets  $S$  with arbitrary “shape”. This property is both sufficient and necessary for SPARSE-COUNT to provide accurate model counts with high probability:

**Theorem 1.** (Informal statement) *A necessary and sufficient condition for SPARSE-COUNT to provide a constant factor approximation to  $|S|$  is that each family  $\mathcal{H}_{i \times n}^{f_i}$   $\epsilon$ -shatters all sets  $S'$  of size  $|S'| = 2^{i+c}$  for some  $c \geq 2$ .*

A formal statement, along with all proofs, is provided in a companion technical report (Zhao et al. 2015).

Long parity constraints, i.e.,  $1/2$ -sparse hash functions, are capable of shattering sets of arbitrary shape. When  $h \in_R \mathcal{H}_{i \times n}^{1/2}$ , it can be shown that configurations  $x \in \{0, 1\}^n$  are placed into hash bins (cells) pairwise independently, and this guarantees shattering of sufficiently large sets of arbitrary shape. Recently, Ermon et al. (2014) showed that sparser hash functions can be used for approximate counting as well. In particular,  $f^*$ -sparse hash functions, for sufficiently large  $f^* \lesssim 1/2$ , were shown to have good enough shattering capabilities. It is currently not known whether  $f^*$  is the optimal constraint density.

## Asymptotically Optimal Constraint Density

We analyze the asymptotic behavior of the minimum constraint density  $f$  needed for SPARSE-COUNT to produce correct bounds with high confidence. As noted earlier, the bottleneck lies in ensuring that  $f$  is large enough for a randomly chosen hash bin to receive at least one element of the set  $S$  under consideration, i.e., the hash family shatters  $S$ .

**Definition 3.** Let  $n, m \in \mathbb{N}, n \geq m$ . For any fixed  $\epsilon > 0$ , the *minimum constraint density*  $f = \tilde{f}_\epsilon(m, n)$  is defined as the pointwise smallest function such that for any constant  $c \geq 2$ ,  $\mathcal{H}_{m \times n}^f$   $\epsilon$ -shatters all sets  $S \in \{0, 1\}^n$  of size  $2^{m+c}$ .

We will show (Theorem 2) that for any  $\epsilon > 0$ ,  $\tilde{f}_\epsilon(m, n) = \Omega(\frac{\log m}{m})$ , and this is asymptotically tight when  $\epsilon$  is small enough and  $m = \Theta(n)$ , which in practice is often the computationally hardest regime of  $m$ . Further, for the regime of  $m = \Theta(n^\beta)$  for  $\beta < 1$ , we show that  $\tilde{f}_\epsilon(m, n) = O(\frac{\log^2 m}{m})$ . Combined with the observation that  $\tilde{f}_\epsilon(m, n) = \Theta(1)$  when  $m = \Theta(1)$ , our results thus reveal how the minimum constraint density evolves from a constant to  $\Theta(\frac{\log m}{m})$  as  $m$  increases from a constant to being linearly related to  $n$ .

The *minimum average constraint length*,  $n \cdot \tilde{f}_\epsilon(m, n)$ , correspondingly decreases from  $n/2$  to  $O(n^{1-\beta} \log^2 n)$  to  $\Theta(\log n)$ , showing that in the computationally hardest regime of  $m = \Theta(n)$ , the parity constraints can in fact be represented using only  $2^{\Theta(\log n)}$ , i.e., a polynomial number of SAT clauses.

**Theorem 2.** *Let  $n, m \in \mathbb{N}, n \geq m$ , and  $\kappa > 1$ . The minimum constraint density,  $\tilde{f}_\epsilon(m, n)$ , behaves as follows:*

1. Let  $\epsilon > 0$ . There exists  $M_\kappa$  such that for all  $m \geq M_\kappa$ :

$$\tilde{f}_\epsilon(m, n) > \frac{\log m}{\kappa m}$$

2. Let  $\epsilon \in (0, \frac{3}{10})$ ,  $\alpha \in (0, 1)$ , and  $m = \alpha n$ . There exists  $N$  such that for all  $n \geq N$ :

$$\tilde{f}_\epsilon(m, n) \leq \left(3.6 - \frac{5}{4} \log_2 \alpha\right) \frac{\log m}{m}$$

3. Let  $\epsilon \in (0, \frac{3}{10})$ ,  $\alpha, \beta \in (0, 1)$ , and  $m = \alpha n^\beta$ . There exists  $N_\kappa$  such that for all  $n \geq N_\kappa$ :

$$\tilde{f}_\epsilon(m, n) \leq \frac{\kappa(1-\beta) \log^2 m}{2\beta m}$$

The lower bound in Theorem 2 follows from analyzing the shattering probability of an  $m+c$  dimensional hypercube  $S_c = \{x \mid x_j = 0 \ \forall j > m+c\}$ . Intuitively, random parity constraints of density smaller than  $\frac{\log m}{m}$  do not even touch the  $m+c$  relevant (i.e., non-fixed) dimensions of  $S_c$  with a high enough probability, and thus cannot shatter  $S_c$  (because all elements of  $S_c$  would be mapped to the same hash bin).

For the upper bounds, we exploit the fact that  $\tilde{f}(m, n)$  is at most the  $f^*$  function introduced by Ermon et al. (2014) and provide an upper bound on the latter. Intuitively,  $f^*$  was defined as the minimum function such that the variance of  $S(h)$  is relatively small. The variance was upper bounded by

considering the worst case “shape” of  $|S|$ : points packed together unrealistically tightly, all fitting together within Hamming distance  $w^*$  of a point. For the case of  $m = \alpha n$ , we observe that  $w^*$  must grow as  $\Theta(n)$ , and divide the expression bounding the variance into two parts: terms corresponding to points that are relatively close (within distance  $\lambda n$  for a particular  $\lambda$ ) are shown to contribute a vanishingly small amount to the variance, while terms corresponding to points that are farther apart are shown to behave as if they contribute to  $S(h)$  in a pairwise independent fashion. The  $\frac{\log m}{m}$  bound is somewhat natural and also arises in the analysis of the rank of sparse random matrices and random sparse linear systems (Kolchin 1999). For example, this threshold governs the asymptotic probability that a matrix  $A$  generated as in Definition 1 has full rank (Cooper 2000). The connection arises because, in our setting, the rank of the matrix  $A$  affects the quality of hashing. For example, an all-zero matrix  $A$  (of rank 0) would map all points to the same hash bucket.

### Improved Bounds on the Model Count

In the previous sections, we established the optimal (smallest) constraint density that provides a constant factor approximation on the model count  $|S|$ . However, depending on the size and structure of  $S$ , even adding constraints of density  $f^* \ll 0.5$  can lead to instances that cannot be solved by modern SAT solvers (see Table 1 below).

In this section we show that for  $f < f^*$  we can still obtain probabilistic upper and lower bounds. The bounds constitute a trade off between small  $f$  for easily solved NP queries and  $f$  close to  $f^*$  for guaranteed constant factor approximation.

To facilitate discussion, we define  $S(h) = |\{x \in S \mid h(x) = 0\}| = |S \cap h^{-1}(0)|$  to be the random variable indicating how many elements of  $S$  survive  $h$ , when  $h$  is randomly chosen from  $\mathcal{H}_{m \times n}^f$  as in Definition 1. Let  $\mu_S = \mathbb{E}[S(h)]$  and  $\sigma_S^2 = \text{Var}[S(h)]$ . Then, it is easy to verify that irrespective of the value of  $f$ ,  $\mu_S = |S|2^{-m}$ .  $\text{Var}[S(h)]$  and  $\Pr[S(h) \geq 1]$ , however, do depend on  $f$ .

### Tighter Lower Bound on $|S|$

Our lower bound is based on Markov’s inequality and the fact that the mean of  $S(h)$ ,  $\mu_S = |S|2^{-m}$ , has a simple linear relationship to  $|S|$ . Previous lower bounds (Gomes et al. 2007; Ermon et al. 2013a) are based on the observation that if at least half of  $T$  repetitions of applying  $h$  to  $S$  resulted in some element of  $S$  surviving, there are likely to be at least  $2^{m-2}$  solutions in  $S$ . Otherwise,  $\mu_S$  would be too small (specifically,  $\leq 1/4$ ) and it would be unlikely to see solutions surviving often. Unlike previous methods, we not only check whether the estimated  $\Pr[S(h) \geq 1]$  is at least  $1/2$ , but also consider an empirical estimate of  $\Pr[S(h) \geq 1]$ . This results in a tighter lower bound, with a probabilistic correctness guarantee derived using Chernoff’s bound.

**Lemma 1.** *Let  $S \subseteq \{0, 1\}^n$ ,  $f \in [0, 1/2]$ , and for each  $m \in \{1, 2, \dots, n\}$ , let hash function  $h_m$  be drawn randomly from  $\mathcal{H}_{m \times n}^f$ . Then,*

$$|S| \geq \max_{m=1}^n 2^m \Pr[S(h_m) \geq 1]. \quad (1)$$

Our theoretical lower bound is  $L = 2^m \Pr[S(h) \geq 1]$ , which satisfies  $L \leq |S|$  by the previous Lemma. In practice we cannot compute  $\Pr[S(h) \geq 1]$  with infinite precision, so our practical lower bound is  $\hat{L} = 2^m \Pr_{\text{est}}[S(h) \geq 1]$ , which will be defined in a moment. We would like to have a statement of the form  $\Pr[|S| \geq \hat{L}] \geq 1 - \delta$ . It turns out that we can guarantee that  $|S|$  is larger than the estimated lower bound shrunk by a  $(1 + \kappa)$  factor with high probability. Let  $Y_k = \mathbb{I}[\{h_k^{-1}(0) \cap S\} \geq 1]$  and  $Y = \sum_{k=1}^T Y_k$ . Let  $\nu = \mathbb{E}[Y] = T\mathbb{E}[Y_1] = T\Pr[S(h) \geq 1]$ . We define our estimator to be  $\Pr_{\text{est}}[S(h) \geq 1] = Y/T$ . Using Chernoff’s bound, we have the following result.

**Theorem 3.** *Let  $\kappa > 0$ . If  $\Pr_{\text{est}}[S(h) \geq 1] \geq c$ , then*

$$\Pr\left[|S| \geq \frac{2^m c}{(1 + \kappa)}\right] \geq 1 - \exp\left(-\frac{\kappa^2 c T}{(1 + \kappa)(2 + \kappa)}\right).$$

### New Upper Bound for $|S|$

The upper bound expression for  $f$  that we derive next is based on the contrapositive of the observation of Ermon et al. (2014) that the larger  $|S|$  is, the smaller an  $f$  suffices to shatter it.

For  $n, m, f$ , and  $\epsilon(n, m, q, f)$  from (Ermon et al. 2014), define:

$$v(q) = \frac{q}{2^m} \left(1 + \epsilon(n, m, q, f) \cdot (q - 1) - \frac{q}{2^m}\right) \quad (2)$$

This quantity is an upper bound on the variance  $\text{Var}[S(h)]$  of the number of surviving solutions as a function of the size of the set  $q$ , the number of constraints used  $m$ , and the statistical quality of the hash functions, which is controlled by the constraint density  $f$ . The following Lemma characterizes the asymptotic behavior of this upper bound on the variance:

**Lemma 2.**  *$q^2/v(q)$  is an increasing function of  $q$ .*

Using Lemma 2, we are ready to obtain an upper bound on the size of the set  $S$  in terms of the probability  $\Pr[S(h) \geq 1]$  that at least one configuration from  $S$  survives after adding the randomly generated constraints.

**Lemma 3.** *Let  $S \subseteq \{0, 1\}^n$  and  $h \in_R \mathcal{H}_{m \times n}^f$ . Then*

$$|S| \leq \min \left\{ z \mid \frac{1}{1 + 2^{2m} v(z)/z^2} > \Pr[S(h) \geq 1] \right\}$$

The probability  $\Pr[S(h) \geq 1]$  is unknown, but can be estimated from samples. In particular, we can draw independent samples of the hash functions and get accurate estimates using Chernoff style bounds. We get the following theorem:

**Theorem 4.** *Let  $S \subseteq \{0, 1\}^n$ . Let  $\Delta \in (0, 1)$ . Suppose we draw  $T = 24 \ln \frac{1}{\Delta}$  hash functions  $h_1, \dots, h_T$  from  $\mathcal{H}_{m \times n}^f$ . If  $\text{Median}(\mathbb{I}[S(h_1) = 0], \dots, \mathbb{I}[S(h_T) = 0]) = 0$  then*

$$|S| \leq \min \left\{ z \mid 1 - \frac{1}{1 + 2^{2m} v(z)/z^2} \geq \frac{3}{4} \right\}$$

*with probability at least  $1 - \Delta$ .*

This theorem provides us with a way of computing upper bounds on the cardinality of  $S$  that hold with high probability. The bound is known to be tight (matching the lower bound derived in the previous section) if the family of hash function used is fully pairwise independent, i.e.  $f = 0.5$ .

## Experimental Evaluation

### Model Counting Benchmarks

We evaluate the quality of our new bounds on a standard model counting benchmark (ANOR2011) from Kroc, Sabharwal, and Selman (2011). Both lower and upper bounds presented in the previous section are parametric: they depend both on  $m$ , the number of constraints, and  $f$ , the constraint density. Increasing  $f$  is always beneficial, but can substantially increase the runtime. The dependence on  $m$  is more complex, and we explore it empirically. To evaluate our new bounds, we consider a range of values for  $f \in [0.01, 0.5]$ , and use a heuristic approach to first identify a promising value for  $m$  using a small number of samples  $T$ , and then collect more samples for that  $m$  to reliably estimate  $P[S(h) \geq 1]$  and improve the bounds on  $|S|$ .

We primarily compare our results to ApproxMC (Chakraborty, Meel, and Vardi 2013b), which can compute a constant factor approximation to user specified precision with arbitrary confidence (at the cost of more computation time). ApproxMC is similar to SPARSE-COUNT, and uses long parity constraints. For both methods, Cryptominisat version 2.9.4 (Soos, Nohl, and Castelluccia 2009) is used as the NP-oracle  $\mathcal{O}_S$  (with Gaussian elimination enabled), and the confidence parameter is set to 0.95, so that bounds reported hold with 95% probability.

Our results on the *Langford12* instance (576 variables, 13584 clauses,  $10^5 \approx \exp(11.5)$  solutions) are shown in Figure 1. The pattern is representative of all other instances we tested. The tradeoff between quality of the bounds and runtime, which is governed by  $f$ , clearly emerges. Instances with small  $f$  values can be solved orders of magnitude faster than with full length XORs ( $f \approx 0.5$ ), but provide looser bounds. Interestingly, lower bounds are not very sensitive to  $f$ , and we empirically obtain good bounds even for very small values of  $f$ . We also evaluate ApproxMC (horizontal and vertical lines) with parameter setting of  $\epsilon = 0.75$  and  $\delta = 0.05$ , obtaining an 8-approximation with probability at least 0.95. The runtime is 47042 seconds. It can be seen that ApproxMC and our bounds offer comparable model counts for dense  $f \approx 0.5$ . However, our method allows to trade off computation time against the quality of the bounds. We obtain non-trivial upper bounds using as little as 0.1% of the computational resources required with long parity constraints, a flexibility not offered by any other method.

Table 1 summarizes our results on other instances from the benchmark and compares them with ApproxMC with a 12 hour timeout. We see that for the instances on which ApproxMC is successful, our method obtains approximate model counts of comparable quality and is generally faster. While ApproxMC requires searching for tens or even hundreds of solutions during each iteration, our method needs only one solution per iteration. Further, we see that long parity constraints can lead to very difficult instances that cannot be solved, thereby reinforcing the benefit of provable upper and lower bounds using sparse constraints (small  $f$ ). Our method is designed to produce non-trivial bounds even when the computational budget is significantly limited, with bounds degrading gracefully with runtime.

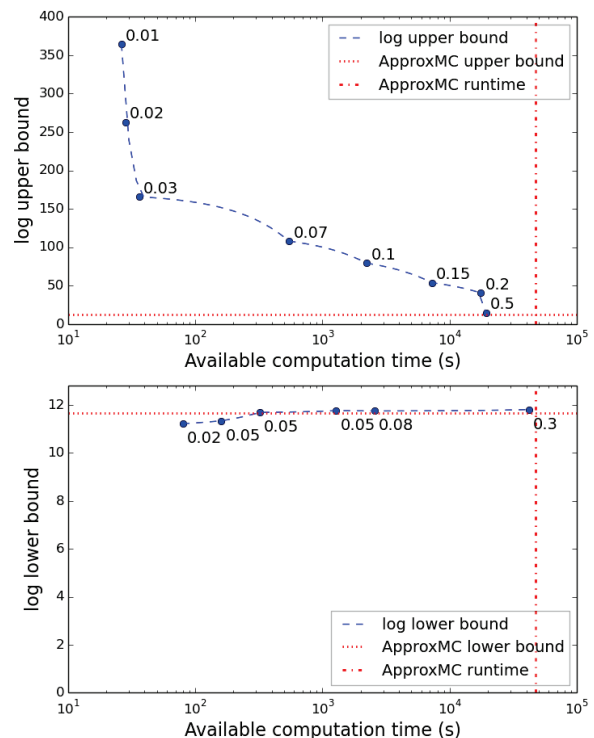


Figure 1: Log upper and lower bounds vs. computation time. Point labels indicate the value of  $f$  used.

### SMT Models for Contingency Table Analysis

In statistics, a *contingency table* is a matrix that captures the (multivariate) frequency distribution of two variables with  $r$  and  $c$  possible values, resp. For example, if the variables are gender (male or female) and handedness (left- or right-handed), then the corresponding  $2 \times 2$  contingency table contains frequencies for the four possible value combinations, and is associated with  $r$  row sums and  $c$  column sums, also known as the row and column marginals.

Fisher’s exact test (Fisher 1954) tests contingency tables for homogeneity of proportion. Given fixed row and column marginals, it computes the probability of observing the entries found in the table under the null hypothesis that the distribution across rows is independent of the distribution across columns. This exact test, however, quickly becomes intractable as  $r$  and  $c$  grow. Statisticians, therefore, often resort to approximate significance tests, such as the chi-squared test.

The associated counting question is: *Given  $r$  row marginals  $R_i$  and  $c$  column marginals  $C_j$ , how many  $r \times c$  integer matrices  $M$  are there with these row and column marginals?* When entries of  $M$  are restricted to be in  $\{0, 1\}$ , the corresponding contingency table is called binary. We are interested in counting both binary and integer tables.

This counting problem for integer tables is #P-complete even when  $r$  is 2 or  $c$  is 2 (Dyer, Kannan, and Mount 1997). Further, for binary contingency tables with so-called structural zeros (Chen 2007) (i.e., certain entries of  $M$  are re-

Table 1: Comparison of run time and bound quality on the ANOR2011 dataset. All true counts and bounds are in log scale.

SAT Instance	Num. Vars.	True Count	Upper Bound			Lower Bound			ApproxMC	
			UB	Runtime(s)	$f$	LB	Runtime (s)	$f$	Estimate	Runtime (s)
lang12	576	–	14.96	19000	0.5	11.17	1280	0.1	12.25	47042
wff.3.150.525	150	32.57	35.06	3800	0.5	31.69	3600	0.1	32.58	43571
2bitmax-6	252	67.52	69.72	355	0.5	67.17	1000	0.5	–	timeout
lang15	1024	–	352.5	240	0.02	19.58	6400	0.02	–	timeout
ls8-normalized	301	27.01	201.5	3800	0.02	25.34	3600	0.02	–	timeout
wff.3.100.150	100	48.94	51.00	2100	0.5	48.30	3600	0.5	–	timeout
wff.4.100.500	100	–	69.31	1100	0.05	37.90	3600	0.05	–	timeout

quired to be 0), we observe that the counting problem is still #P-complete. This can be shown via a reduction from the well-known *permanent* computation problem, which is #P-complete even for 0/1 matrices (Valiant 1979a).

Model counting for contingency tables is formulated most naturally using integer variables and arithmetic constraints for capturing the row and column sums. While integer linear programming (ILP) appears to be a natural fit, ILP solvers do not scale very well on this problem as they are designed to solve optimization problems and not feasibility queries. We therefore propose to encode the problem using a Satisfiability Modulo Theory (SMT) framework (Barrett, Stump, and Tinelli 2010), which extends propositional logic with other underlying theories, such as bitvectors and real arithmetic. We choose a bitvector encoding where each entry  $a_{ij}$  of  $M$  is represented as a bitvector of size  $\lceil \log_2 \min(R_i, C_j) \rceil$ . The parity constraints are then randomly generated over the individual bits of each bitvector, and natively encoded into the model as XORs. As a solver, we use Z3 (De Moura and Bjørner 2008).

We evaluate our bounds on six datasets:

**Darwin’s Finches (df).** The marginals for this binary contingency table dataset are from Chen et al. (2005). This is one of the few datasets with known ground truth:  $\log_2 |S| \approx 55.8982$ , found using a clever divide-and-conquer algorithm of David desJardins. The 0-1 label in cell  $(x, y)$  indicates the presence or absence of one of 13 finch species  $x$  at one of 17 locations  $y$  in the Galápagos Islands. To avoid trivialities, we drop one of the species that appears in every island, resulting in  $12 \times 17 = 204$  binary variables.

**Climate Change Perceptions (icons).** This  $9 \times 6$  non-binary dataset is taken from the `alymer` R package (West and Hankin 2008). It concerns lay perception of climate change. The dataset is based on a study reported by O’Neil (2008) in which human subjects are asked to identify which icons (such as polar bears) they find the most concerning. There are 18 structural zeros representing that not all icons were shown to all subjects.

**Social Anthropology (purum).** This  $5 \times 5$  non-binary dataset (West and Hankin 2008) concerns marriage rules of an isolated tribe in India called the Purums, which is subdivided into 5 sibs. Structured zeros represent marriage rules that prevent some sibs from marrying other sibs.

**Industrial Quality Control (iqd).** This  $4 \times 7$  non-binary dataset (West and Hankin 2008) captures an industrial quality control setting. Cell  $(x, y)$  is the number of defects in the

Table 2: Lower (LB) and upper (UB) bounds on  $\log_2 |S|$ . The trivial upper bound (Trv. UB) is the number of binary variables.  $f^*$  denotes the best previously known minimum  $f$  (Ermon et al. 2014) required for provable upper bounds.

Dataset	Table Size	$f^*$	LB ( $f$ )	$\log_2  S $	UB	Trv. UB
df	$12 \times 17$	0.18	<b>53</b> (0.03)	55.90	<b>150</b>	204
icons	$9 \times 6$	0.19	<b>58</b> (0.04)	–	<b>183</b>	236
purum	$5 \times 5$	0.26	<b>29</b> (0.13)	–	<b>52</b>	125
iqd	$4 \times 7$	0.34	<b>15</b> (0.10)	–	<b>17</b>	76
synth_8	$8 \times 8$	0.41	<b>5</b> (0.30)	5.64	<b>16</b>	64
synth_20	$20 \times 20$	0.42	<b>8</b> (0.40)	8.49	<b>14</b>	400

$x$ -th run attributable to machine  $y$ . It has 9 structured zeros, representing machines switched off for certain runs.

**Synthetic Data (synth).** This  $n \times n$  binary dataset contains *blocked matrices* (Golshan, Byers, and Terzi 2013). The row and column marginals are both  $\{1, n - 1, \dots, n - 1\}$ . It can be seen that a blocked matrix has either has a value of 1 in entry  $(1, 1)$  or it has two distinct entries with value 1 in the first row and the first column, cell  $(1, 1)$  excluded. Instantiating the first row and the first column completely determines the rest of the table. It is also easy to verify that the desired count is  $1 + (n - 1)^2$ .

Table 2 summarizes the obtained lower and upper bounds on the number of contingency tables, with a 10 minute timeout. For the datasets with ground truth, we see that very sparse parity constraints (e.g.,  $f = 0.03$  for the Darwin finches dataset, as opposed to a theoretical minimum of  $f^* = 0.18$ ) often suffice in practice to obtain very accurate lower bounds. For the *iqd* dataset, we obtain upper and lower bounds within a small constant factor. For other datasets, there is a wider gap between the upper and lower bounds. However, the upper bounds we obtain are orders of magnitude tighter than the trivial log-upper bounds, which is the number of variables in a binary encoding of the problem.

## Conclusions

We introduced a novel analysis of the randomized hashing schemes used by numerous recent approximate model counters and probabilistic inference algorithms. We close a theoretical gap, providing a tight asymptotic estimate for the minimal constraint density required. Our analysis also shows, for the first time, that even very short parity constraints can be used to generate non-trivial upper bounds on

model counts. Thanks to this finding, we proposed a new scheme for computing upper and lower bounds on the model count. Asymptotically, these bounds are guaranteed to become tight (up to a constant factor) as the constraint density grows. Empirically, given very limited computational resources, we are able to obtain new upper bounds on a variety of benchmarks, including a novel application for the analysis of statistical contingency tables.

A promising direction for future research is the analysis of related ensembles of random parity constraints, such as low-density parity check codes (Achlioptas and Jiang 2015).

## Acknowledgments

This work was supported by the Future of Life Institute (grant 2015-143902).

## References

- Achlioptas, D., and Jiang, P. 2015. Stochastic integration via error-correcting codes. In *Proc. Uncertainty in Artificial Intelligence*.
- Barrett, C.; Stump, A.; and Tinelli, C. 2010. The Satisfiability Modulo Theories Library (SMT-LIB). [www.SMT-LIB.org](http://www.SMT-LIB.org).
- Belle, V.; Van den Broeck, G.; and Passerini, A. 2015. Hashing-based approximate probabilistic inference in hybrid domains. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T. 2009. Handbook of satisfiability. *frontiers in artificial intelligence and applications*, vol. 185.
- Chakraborty, S.; Meel, K.; and Vardi, M. 2013a. A scalable and nearly uniform generator of SAT witnesses. In *Proc. of the 25th International Conference on Computer Aided Verification (CAV)*.
- Chakraborty, S.; Meel, K.; and Vardi, M. 2013b. A scalable approximate model counter. In *Proc. of the 19th International Conference on Principles and Practice of Constraint Programming (CP)*, 200–216.
- Chen, Y.; Diaconis, P.; Holmes, S. P.; and Liu, J. S. 2005. Sequential monte carlo methods for statistical analysis of tables. *Journal of the American Statistical Association* 100(469):109–120.
- Chen, Y. 2007. Conditional inference on tables with structural zeros. *Journal of Computational and Graphical Statistics* 16(2).
- Cooper, C. 2000. On the rank of random matrices. *Random Structures & Algorithms* 16(2):209–232.
- De Moura, L., and Bjørner, N. 2008. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 337–340.
- Dyer, M.; Kannan, R.; and Mount, J. 1997. Sampling contingency tables. *Random Structures and Algorithms* 10(4):487–506.
- Ermon, S.; Gomes, C. P.; Sabharwal, A.; and Selman, B. 2013a. Optimization with parity constraints: From binary codes to discrete integration. In *Proc. of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Ermon, S.; Gomes, C. P.; Sabharwal, A.; and Selman, B. 2013b. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proc. of the 30th International Conference on Machine Learning (ICML)*.
- Ermon, S.; Gomes, C. P.; Sabharwal, A.; and Selman, B. 2014. Low-density parity constraints for hashing-based discrete integration. In *Proc. of the 31st International Conference on Machine Learning (ICML)*, 271–279.
- Fisher, R. 1954. *Statistical Methods for Research Workers*. Oliver and Boyd.
- Gogate, V., and Dechter, R. 2007. Approximate counting by sampling the backtrack-free search space. In *Proc. of the 22nd National Conference on Artificial Intelligence (AAAI)*, volume 22, 198–203.
- Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Uncertainty in Artificial Intelligence*.
- Golshan, B.; Byers, J.; and Terzi, E. 2013. What do row and column marginals reveal about your dataset? In *Advances in Neural Information Processing Systems*, 2166–2174.
- Gomes, C. P.; Hoffmann, J.; Sabharwal, A.; and Selman, B. 2007. Short XORs for model counting: From theory to practice. In *Theory and Applications of Satisfiability Testing (SAT)*, 100–106.
- Gomes, C. P.; Sabharwal, A.; and Selman, B. 2006. Model counting: A new strategy for obtaining good bounds. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI)*, 54–61.
- Ivrii, A.; Malik, S.; Meel, K. S.; and Vardi, M. Y. 2015. On computing minimal independent support and its applications to sampling and counting. *Constraints* 1–18.
- Kolchin, V. F. 1999. *Random graphs*. Number 53 in Encyclopedia of Mathematics and its Applications. Cambridge University Press.
- Kroc, L.; Sabharwal, A.; and Selman, B. 2011. Leveraging belief propagation, backtrack search, and statistics for model counting. *Annals of Operations Research* 184(1):209–231.
- O’Neil, S. 2008. *An Iconic Approach to Communicating Climate Change*. Ph.D. Dissertation, School of Environmental Science, University of East Anglia.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1):107–136.
- Sang, T.; Beame, P.; and Kautz, H. 2005. Solving Bayesian networks by weighted model counting. In *Proc. of the 20th National Conference on Artificial Intelligence (AAAI)*, volume 1, 475–482.
- Sang, T.; Bacchus, F.; Beame, P.; Kautz, H.; and Pitassi, T. 2004. Combining component caching and clause learning for effective model counting. In *Theory and Applications of Satisfiability Testing (SAT)*.
- Sheldon, D. R., and Dietterich, T. G. 2011. Collective graphical models. In *Advances in Neural Information Processing Systems*, 1161–1169.
- Soos, M.; Nohl, K.; and Castelluccia, C. 2009. Extending SAT solvers to cryptographic problems. In *Theory and Applications of Satisfiability Testing (SAT)*.
- Stockmeyer, L. 1985. On approximation algorithms for #P. *SIAM Journal on Computing* 14(4):849–861.
- Tseitin, G. S. 1968. On the complexity of derivation in the propositional calculus. In Slisenko, A. O., ed., *Studies in Constructive Mathematics and Mathematical Logic, Part II*.
- Valiant, L. G. 1979a. The complexity of computing the permanent. *Theoretical computer science* 8(2):189–201.
- Valiant, L. 1979b. The complexity of enumeration and reliability problems. *SIAM Journal on Computing* 8(3):410–421.
- West, L. J., and Hankin, R. K. 2008. Exact tests for two-way contingency tables with structural zeros. *Journal of Statistical Software* 28(11):1–19.
- Zhao, S.; Chaturapruek, T.; Sabharwal, A.; and Ermon, S. 2015. Closing the gap between short and long xors for model counting. Technical report, Stanford University.