Constraints (2019) 24: 162–182
https://doi.org/10.1007/s10601-018-9296-3

Model counting with error-correcting codes

Dimitris Achlioptas[1] · **Panos Theodoropoulos**[2]

Published online: 8 February 2019

Abstract
The idea of counting the number of satisfying truth assignments (models) of a formula by adding random parity constraints can be traced back to the seminal work of Valiant and Vazirani showing that NP is as easy as detecting unique solutions. While theoretically sound, the random parity constraints used in that construction suffer from the following drawback: each constraint, on average, involves half of all variables. As a result, the branching factor associated with searching for models that also satisfy the parity constraints quickly gets out of hand. In this work we prove that one can work with much shorter parity constraints and still get rigorous mathematical guarantees, especially when the number of models is large so that many constraints need to be added. Our work is motivated by the realization that the essential feature for a system of parity constraints to be useful in probabilistic model counting is that its set of solutions resembles an error-correcting code.

Keywords Model counting · Satisfiability · Randomized algorithms · Systems of parity equations · Coding theory · Low density parity check codes

1 Introduction

Imagine a blind speaker entering an amphitheater, wishing to estimate the number of people present. She starts by asking "Is anyone here?" and hears several voices saying "Yes." She then says "Flip a coin inside your head; if it comes up heads, please never answer again." She then asks again "Is anyone here?" and roughly half of the people present say "Yes." She then asks them to flip another coin, and so on. When silence occurs after i rounds, she estimates that approximately 2^i people are present.

Research supported by European Research Council Starting Grant 210743, NSF grant CCF-1514128, NSF CCF-1733884, an Adobe research grant, and the Greek State Scholarships Foundation (IKY).

✉ Panos Theodoropoulos
panosth@gmail.com

Dimitris Achlioptas
optas@cs.ucsc.edu

[1] Department of Computer Science, University of California Santa Cruz, 1156 High Street, Santa Cruz, CA 95064, USA

[2] Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, 157 84 Athens, Greece

Given a CNF formula F with n variables we would like to approximate the size of its set of satisfying assignments (models), $S = S(F)$, using a similar approach. Concretely, to check whether $|S| \geq 2^i$ we will form a random set $R_i \subseteq \{0,1\}^n$ such that $\Pr[\sigma \in R_i] = 2^{-i}$ for all $\sigma \in \{0,1\}^n$, and then check whether $S \cap R_i \neq \emptyset$. The key idea, following the theoretical path pioneered by [15, 17], and [12], is to represent the random set R_i *implicitly*, as the set of solutions of a system of i random parity (XOR) constraints (linear equations modulo 2). This way the check can be performed by either a SAT solver that also supports parity constraints or by a standard SAT solver, after converting the parity constraints to clauses.

As we discuss in Section 2, there has been a long line of works aiming to make the above theoretical scheme practical, including [5–8, 18] and [3, 4, 9]. In *all* of these works, the variables appearing in each constraint are chosen independently of the variables in all other constraints. Specifically, in most works, each constraint simply includes each variable independently with probability 1/2 so that each constraint, on average, includes $n/2$ variables. Systems of such long parity constraints have the benefit that membership in their set of solutions enjoys pairwise independence, making the statistical analysis of the random variable $|S \cap R_i|$ very simple. Unfortunately, though, as n increases, the correspondingly increasing constraint length severely limits the capacity of solvers to find elements of $S \cap R_i$, since flipping any single variable flips the value of, roughly, half of all parity constraints. This severe limitation was already recognized in the very first works in the area [7, 8], and later works [6, 18] have tried to remedy it by considering parity equations where each constraint includes each variable independently with probability $p < 1/2$. While such sparsity helps in the search for elements of $S \cap R$, the statistical properties of the resulting random sets R deteriorate rapidly as p decreases.

In this work we make two contributions. The first is to give algorithms that come with rigorous approximation guarantees without requiring pairwise independence from the sets R_i. Indeed, our analysis does not even require the sets R_i to be the solution sets of systems of parity constraints. Instead, the performance of the algorithms is characterized by a single, explicitly defined statistical quantity of the distribution on the sets R_i, recovering all previous known results when membership in R_i enjoys pairwise independence. This contribution is largely one of exposition, as most of the ideas employed have appeared implicitly or explicitly in earlier works, as we discuss in Section 2. Nevertheless, we feel that our analysis is significantly more accessible, succinct, and modular, highlighting exactly "where the pain is." As such, we hope that it will allow more researchers to contribute by focusing attention to the key issue.

Our second and main contribution is to address the aforementioned pain by introducing random sets R_i that are both statistically good and easy to intersect with S. We achieve this by taking the sets to be the codewords of Low Density Parity Check (LDPC) codes. Similarly to earlier works, the elements of our sets are the solutions of random systems of parity equations. Unlike earlier works, though, the variables in each equation are **not** selected independently of the other equations. This entanglement endows the resulting solution sets with dramatically better statistical properties, a fact established via an analysis far more sophisticated than the case of independent entries.

By analyzing the statistical properties of sets R_i defined via LDPC codes we give algorithms that yield rigorous approximation guarantees with constraints far shorter than any previously used. That said, the number of solver invocations implied by our analysis in order to have a rigorous guarantee can be impractically large, due to the large constant factors involved. We believe that the fault in this lies with our analysis, which we conjecture is very pessimistic. As experimental evidence for this conjecture we show that using our sets R_i but invoking the solver *dramatically* fewer times than what is needed by our analysis,

yields excellent accuracy (and dramatic speedup) on a wide range of formulas with known model counts. Moreover, our sets yield rigorous model count lower bounds far beyond the reach of all existing counters.

2 Previous work

We focus our discussion on previous works most directly related to ours to provide context for our contribution. For a more general review of the use of random parity equations for approximate counting and sampling, see the excellent recent survey by Meel et al. [10]. Also, to facilitate the discussion, let us define for an integer i the random set $R_i = \{\sigma \in \{0,1\}^n : A\sigma = b\}$, where $b \in \{0,1\}^i$ is uniformly random, while each entry of $A \in \{0,1\}^{i \times n}$ is set to 1 independently with probability p.

The first work on practical approximate model counting using systems of random parity equations was by Gomes, Sabharwal, and Selman [8], using sets R_i as above. Specifically, they proved that if $p = 1/2$, then one can rigorously approximate $\log_2 |S|$ within an additive constant by repeatedly checking whether $S \cap R_i = \emptyset$ for various values of i. (To perform the check the parity constraints were transformed to clauses using the Tseitin transformation and a standard SAT solver was used.) Further, they showed that in order to get rigorous lower bounds any $0 < p < 1/2$ can be used, yielding easier to solve constraints, but also lower bounds that may be arbitrarily far from the truth. In [7], Gomes et al. showed experimentally that it can be possible to achieve good accuracy (without guarantees) using parity constraints of length $k \ll n/2$, by considering formulas for which the correct answer was derived by other means. Recently, Achim, Sabharwal, and Ermon [1] showed that one can also derive rigorous lower bounds (but not upper bounds) by using constraints other than parity constraints which can be easier for the solver.

Interest in the subject was rekindled by works of Chakraborty, Meel, and Vardi [3] and of Ermon, Gomes, Sabharwal, and Selman et al. [5]. In [3], a complete rigorous approximate model counter, called ApproxMC, was given which takes as input any $\delta, \epsilon > 0$, and with probability at least $1 - \delta$ returns a number in the range $(1 \pm \epsilon)|S|$. In [5] an algorithm with a similar (δ, ϵ)-guarantee is given, called WISH, for the more general problem of approximating sums of the form $\sum_{\sigma \in \{0,1\}^n} w(\sigma)$, where w is a non-negative real-valued function over Ω^n, where Ω is a finite domain. (The basic algorithm in [5] achieves a 16-approximation, but using a transformation that blows up the number of variables, allows for arbitrary approximation accuracy.)

ApproxMC uses the satisfiability solver CryptoMiniSAT (CMS) [14] which has native support and sophisticated reasoning for parity constraints. Also, CMS can take as input a cutoff value $z \geq 1$, so that it will run until it either finds z solutions or determines the number of solutions to be less than z. ApproxMC makes use of this capability in order to target i such that $|S \cap R_i| = \Theta(\delta^{-2})$, instead of i such that $|S \cap R_i| \approx 1$. Our algorithms make similar use of this capability by using several different cutoffs.

Both ApproxMC and WISH use sets R_i with $p = 1/2$, which limits the range of problems they can handle. One idea to mitigate this issue in practice, proposed by Chakraborty et al. in [2], is motivated by the observation that there can often be a small subset, I, of variables such that every value-assignment to the variables in I has at most one satisfying extension. Such a set I is called an independent support set and, clearly, $|S|$ equals the number of value assignments to the variables of I that can be extended to satisfying assignments. As a result, it suffices to add random parity constraints that only involve variables in I, so that each constraint has $|I|/2$ instead of $n/2$ variables on average. Since independent

support sets of size much less than n can often be found [9] in practice, this allows scaling ApproxMC to certain classes of formulas with thousands of variables. Naturally, the benefit of only involving independent support variables in the parity constraints extends to all methods using such constraints for model counting, including ours.

The first effort to develop rigorous performance guarantees for sets R_i as above with $p < 1/2$ was made by Ermon, Gomes, Sabharwal, and Selman in [6], where an explicit expression was given for the smallest allowed p as a function of $|S|, n, \delta, \epsilon$. The analysis has two parts, one corresponding to the analysis of a probabilistic walk, and one that corresponds to a packing argument. We use the same packing argument in Section 8 when passing from (7) to inequality (8). Also, the analysis in [6] introduces the notion of average-universal hash functions, which is very closely related to our definition of "lumpiness" in (1). The analysis in [6] was recently improved by Zhao, Chaturapruek, Sabharwal, and Ermon [18] who, among other results, showed that when $\log_2 |S| = \Omega(n)$, in order to get rigorous approximation guarantees it is enough to use constraints of length only $O(\log n)$. While, prima facie, this seems a very promising result, we will see that the dependence on the constants involved in the asymptotics is very important in practice. For example, in our experiments we observe that already setting $p = 1/8$ yields results whose accuracy is much worse than that achieved by LDPC constraints.

Finally, Chakraborty, Meel, and Vardi [4] introduced a very nice idea for reducing the number of solver invocations without any compromise in approximation quality. It amounts to using *nested* sequences of random sets $R_1 \supseteq R_2 \supseteq R_3 \supseteq \cdots \supseteq R_n$ in the quest for $i \approx \log_2 |S|$. The key insight is that using nested (instead of independent) random sets R_i means that $|S \cap R_i|$ is deterministically non-increasing in i, so that linear search for i can be replaced with binary search, thus reducing the number of solver invocations from linear to logarithmic in n. We use the same idea in our work and give a streamlined proof that highlights its generality.

3 Lower bounds are easy

To simplify exposition we only discuss lower bounds of the form $|S| \geq 2^i$ for $i \in \mathbb{N}$, deferring the discussion of more precise estimates to Section 5. For any distribution $\mathcal{D}$, let $R \sim \mathcal{D}$ denote that random variable R has distribution $\mathcal{D}$.

Definition 1 Let $\mathcal{D}$ be a distribution on subsets of a set U and let $R \sim \mathcal{D}$. We say that $\mathcal{D}$ is i-uniform if $\Pr[\sigma \in R] = 2^{-i}$ for every $\sigma \in U$.

When $U = \{0,1\}^n$, some examples of i-uniform distributions are:

(i) R contains each $\sigma \in \{0,1\}^n$ independently with probability 2^{-i}.
(ii) R is a uniformly random subcube of dimension $n - i$.
(iii) $R = \{\sigma : A\sigma = b\}$, where $A \in \{0,1\}^{i \times n}$ is *arbitrary* and $b \in \{0,1\}^i$ is *uniformly* random.

Any i-uniform distribution $\mathcal{D}_i$ can be used to compute a *rigorous* lower bound on the number of satisfying truth assignments of a formula (models). This is because if $R_i \sim \mathcal{D}_i$ and in several independent trials we find that, typically, $S \cap R_i \neq \emptyset$, we can be highly confident that the expectation of $|S \cap R_i|$ can not be much less than 1. But since for any i-uniform distribution this expectation equals $2^{-i}|S|$, we can be highly confident that $|S|$ is not much less than 2^i.

Algorithm 1 follows this intuition closely, except that instead of non-emptiness, it focuses on $S \cap R$ typically having at least 2 elements. For this, in line 5 we trim $|S \cap R|$ to at most 4

(by running CryptoMiniSAT with a cutoff of 4), so that the event $Z \geq 2t$ in line 8 can only occur if the intersection had size at least 2 in at least $t/2$ trials.

Algorithm 1 Given i, t decides if $|S| \geq 2^i$ with 1-sided error probability $e^{-t/8}$

```
1:  Z ← 0
2:  j ← 0
3:  while j < t and Z < 2t do          ▷ The condition Z < 2t is an optimization
4:      Sample R_j ∼ D_i               ▷ D_i can be any i-uniform distribution
5:      Y_j ← min{4, |S ∩ R_j|}        ▷ Run CryptoMiniSat with cutoff 4
6:      Z ← Z + Y_j
7:      j ← j + 1
8:  if Z ≥ 2t then
9:      return "Yes"
10: else
11:     return "I Don't know"
```

To analyze Algorithm 1 we only need Hoeffding's Inequality.

Lemma 1 (**Hoeffding's Inequality**) *If $Z = Y_1 + \cdots + Y_t$, where $0 \leq Y_i \leq b$ are independent random variables, then for any $w \geq 0$,*

$$\Pr[Z/t \geq \mathbb{E}Z/t + w] \leq e^{-2t(w/b)^2} \quad \textit{and} \quad \Pr[Z/t \leq \mathbb{E}Z/t - w] \leq e^{-2t(w/b)^2} \ .$$

Theorem 1 *The output of Algorithm 1 is incorrect with probability at most $e^{-t/8}$.*

Proof For the algorithm's output to be incorrect it must be that $|S| < 2^i$ and $Z/t \geq 2$. If $|S| < 2^i$, then $\mathbb{E}Y_j \leq |S|2^{-i} < 1$ for all j, implying $\mathbb{E}Z/t < 1$. Since Z is the sum of t i.i.d. random variables $0 \leq Y_j \leq 4$, Hoeffding's inequality implies that $\Pr[Z/t \geq 2] \leq \Pr[Z/t \geq \mathbb{E}Z/t + 1] \leq \exp(-t/8)$. □

3.1 The efficacy of Algorithm 1

Notably, Theorem 1 only bounds the probability that Algorithm 1 gets us into trouble by answering "Yes" when it shouldn't, i.e., when $|S| < 2^i$. It does not address at all the *efficacy* of Algorithm 1, i.e., the probability of "Don't know" when $|S| \geq 2^i$. As we will see, bounding this probability requires much more than mere i-uniformity.

Naturally, increasing the number 4 in line 5 of Algorithm 1 and/or decreasing the number 2 in line 8 (and correspondingly in line 3) makes Algorithm 1 more likely to return "Yes." Such changes, though, also increase the number of iterations needed to guarantee the same bound on the error probability. The numbers 4 and 2 appear in practice to strike a good balance between the algorithm being fast vs. informative.

3.2 Dealing with timeouts

Line 5 of Algorithm 1 requires running the solver until either it finds 4 elements of $S \cap R_j$ or until it determines that fewer than 4 such elements exist. In reality, timeouts may prevent the solver from making this determination. Nevertheless, if we always set Y_j to a number *no greater* than $\min\{4, |S \cap R_j|\}$, then both Theorem 1 and its proof remain valid. Thus, whenever a timeout occurs, we can set Y_j to the number $s < 4$ of elements of $S \cap R_j$ found so far. Naturally, this modification may increase the probability of the algorithm returning "Don't know."

3.3 Searching for a lower bound

To derive a lower bound for $\log_2 |S|$ we can invoke Algorithm 1 with $i = 1, 2, \ldots, n$ sequentially and keep the best lower bound returned (if any). For any $\delta > 0$, Theorem 1 implies that if we set $t = \lceil 8 \ln(n/\delta) \rceil$ in all invocations of Algorithm 1, the probability we will end up with an invalid lower bound is at most δ (since invalidity requires at least one of the n answers of Algorithm 1 to be erroneous).

There is no reason, though, to increase i sequentially. We can be more aggressive and invoke Algorithm 1 with $i = 1, 2, 4, 8, \ldots$ until the first "Don't know" occurs, say at $i = 2^u$. At that point we can perform binary search in $\{2^{u-1}, \ldots, 2^u - 1\}$, treating every "Don't know" answer as a (conservative) imperative to reduce the interval's upper bound to the midpoint and every "Yes" answer as an allowance to increase the interval's lower bound to the midpoint. We call this scheme "doubling binary search." In practice this can be further accelerated by invoking Algorithm 1 with a very small number of trials in the course of the doubling-binary search, e.g., just $t = 2$. The result of this search is then treated as an optimistic "ballpark" estimate and we do a search in its vicinity using the proper number of iterations. Algorithm 2 does precisely that. To avoid trivialities, below we assume that F is satisfiable, so that $\log_2 |S| \geq 0$.

Theorem 2 *The output of Algorithm 2 exceeds* $\log_2 |S|$ *with probability at most* θ.

Proof Since $\log_2 |S| \geq 0$, for the answer to be wrong it must be that an invocation of Algorithm 1 in line 19 with $i > \log_2 |S|$ returned "Yes". Since Algorithm 2 invokes Algorithm 1 in line 19 at most $\lceil \log_2 n \rceil$ times, and in each such invocation we set t so that the probability of error is at most $\theta / \lceil \log_2 n \rceil$, the claim follows. □

Algorithm 2 Given $\theta > 0$, returns i such that $|S| \geq 2^i$ with probability at least $1 - \theta$

```
1:  t ← small integer                                   ▷ t ← 2 works well in practice
2:
3:  i ← 1                                               ▷ Doubling search until first "Don't Know"
4:  while {A1(i, t) = "Yes"} do                         ▷ A1(i, t) denotes the output of Algorithm 1
5:      i ← 2i
6:  h ← 2i − 1                                          ▷ First "Don't Know" occurred for i = h + 1
7:
8:  while i < h do                                      ▷ Binary search for "Yes" in [i, h]
9:      m ← i + ⌈(h − i)/2⌉
10:     if A1(m, t) = "Yes" then
11:         i ← m
12:     else
13:         h ← m − 1
14:
15: j ← 0
16: repeat                                              ▷ Doubling search backwards from
17:     i ← i − 2^j                                     ▷ i − 1 for a lower bound that holds
18:     j ← j + 1                                       ▷ with probability 1 − θ/⌈log2 n⌉
19: until i ≤ 0 or A1(i, ⌈8 ln(⌈log2 n⌉/θ)⌉) = "Yes"
20: return max{0, i}
```

We note that Algorithm 2 can be modified to take as input any already established lower bound $0 < \ell \leq \log_2 |S|$, e.g., one derived by running the unmodified algorithm above, so that ℓ becomes the starting point for the initial doubling search.

4 What does it take to get a *good* lower bound?

Algorithms 1 and 2 may underestimate $\log_2 |S|$ arbitrarily. The reason for this is that even though i-uniformity makes the expected size of $S \cap R_i$ be $2^{-i}|S|$, the *actual* size of $S \cap R_i$ may behave like the winnings of a lottery: huge with very small probability, but typically zero. If such a lottery phenomenon is present, then in any realistic number of trials we will always see $S \cap R_i = \emptyset$ even if $i \ll \log_2 |S|$, in exactly the same manner that anyone playing the lottery a realistic number of times will, most likely, never experience winning.

The discussion above makes it clear that the heart of the matter is controlling the *variance* of $|S \cap R|$. We will do this is by bounding the "lumpiness" of the sets in the support of the distribution $\mathcal{D}_i$, as measured by the quantity defined in (1) below, which measures lumpiness at a scale of M (the smaller the quantity in (1), the less lumpy the distribution, and the smaller the variance).

Definition 2 Let $\mathcal{D}$ be any distribution on subsets of $\{0,1\}^n$ and let $R \sim \mathcal{D}$. For any fixed $M \geq 1$, let

$$\text{Boost}(\mathcal{D}, M) = \max_{\substack{S \subseteq \{0,1\}^n \\ |S| \geq M}} \frac{1}{|S|(|S|-1)} \sum_{\substack{\sigma,\tau \in S \\ \sigma \neq \tau}} \frac{\Pr[\sigma, \tau \in R]}{\Pr[\sigma \in R]\Pr[\tau \in R]} \ . \tag{1}$$

To get some intuition for (1) observe that the ratio inside the sum equals the factor by which the a priori probability that a truth assignment belongs in R is modified by conditioning on some other truth assignment belonging in R. For example, if membership in R is pairwise independent, then $\text{Boost}(\mathcal{D}, \cdot) = 1$, i.e., there is no modification. Another thing to note is that since we require $|S| \geq M$ instead of $|S| = M$ in (1), the function $\text{Boost}(\mathcal{D}, \cdot)$ is trivially non-increasing in M.

While pairwise independence achieves $\text{Boost}(\mathcal{D}, \cdot) = 1$ for all sizes, the associated parity constraints tend to be make determining $|S \cap R|$ difficult, especially when $|R|$ is far from the extreme values 2^n and 1. Distributions achieving $\text{Boost}(\mathcal{D}, \cdot) < 1$ exist, but are even harder to work with. In the rest of the paper we restrict to the case $\text{Boost}(\mathcal{D}, \cdot) \geq 1$ (hence the name Boost). As we will see, the crucial requirement for an i-uniform distribution $\mathcal{D}_i$ to be useful is that $\text{Boost}(\mathcal{D}_i, \Omega(2^i))$ is not very large, i.e., an i-uniform distribution can be useful even if $\text{Boost}(\mathcal{D}_i)$ is huge for sets of size much less than 2^i. This is what will make it possible to derive rigorous results in the absence of pairwise independence.

The three examples of i-uniform distributions discussed earlier differ dramatically in terms of Boost.

(i) When R is formed by selecting each element of $\{0,1\}^n$ independently, trivially, $\text{Boost}(\mathcal{D}, \cdot) = 1$ since we have full, not just pairwise, independence. But just specifying R in this case requires space proportional to $|R|$.

(ii) When R is a random subcube, specifying R is extremely compact and searching for models in R is about as easy as one could hope for: just "freeze" a random subset of i variables and search over the rest. Unfortunately, random subcubes can be statistically terrible, having huge Boost, e.g., when S is itself a subcube.

(iii) When $R = \{\sigma : A\sigma = b\}$, where $b \in \{0,1\}^i$ is uniformly random, the distribution of A can make a huge difference in the value of Boost.

- At one end of the spectrum, if $A = \mathbf{0}$, then R is either the empty set or the entire cube $\{0,1\}^n$, depending on whether $b = \mathbf{0}$ or not. Thus, $\text{Boost}(\mathcal{D}, \cdot) = 2^i$, the maximum possible.
- The other end of the spectrum occurs when A is uniform in $\{0,1\}^{i\times n}$, i.e., when the entries of A are independently 0 or 1 with equal probability. In this case $\text{Boost}(\mathcal{D}, \cdot) = 1$, a remarkable and well-known fact that can be seen by the following simple argument. By the definition of R, for any fixed $\sigma \neq \tau$,

$$\Pr[\sigma, \tau \in R] = \Pr[A\sigma = A\tau \wedge A\sigma = b] = \Pr[A(\sigma - \tau) = 0] \cdot \Pr[A\sigma = b] \ ,$$

where to see the second equality imagine first selecting the matrix A and only then the vector b. As we already saw, $\Pr[A\sigma = b] = 2^{-i}$. To prove that $\Pr[A(\sigma - \tau) = 0] = 2^{-i}$, and thus conclude the proof, select any non-zero coordinate of $\sigma - \tau$, say j (as $\sigma \neq \tau$ there is at least one). Set arbitrarily all entries of A, except those in the j-th column. Observe now that whatever the setting is, there is exactly one choice for each entry in the j-th column such that $A(\sigma - \tau) = 0$. Since the i elements of the j-th column are selected uniformly and independently, the claim follows.

5 Rigorous counting without pairwise independence

For each $i \in [n]$, let us fix an arbitrary i-uniform distribution $\mathcal{D}_i$ on subsets of $\{0,1\}^n$. We will show that given a lower bound $0 \leq L \leq |S|$, we can get a rigorous probabilistic approximation of $|S|$ within a factor of $1 \pm \delta$ with a number of solver invocations proportional to the square of $B = \max_{\ell \leq i \leq n} \text{Boost}(\mathcal{D}_i, 2^i)$, where $\ell \approx \log_2(\delta L)$. To put this in perspective we note the following.

- If membership in $R_i \sim \mathcal{D}_i$ enjoys pairwise independence for all $i \in [n]$, then $B = 1$ and the number of solver invocations is $O(n\delta^{-2}\log(n/\theta))$, as in [3]. If, further, we use nested sampling sets, an idea introduced in [4], then the number of solver invocations drops to $O(\delta^{-2}\log(n/\theta))$, matching [4].
- The restriction $i \geq \ell$ in the definition of B is *key* for using short parity constraints. For small values of i it seems unreasonable to expect good statistical behavior for random sets R_i encoded by such constraints. But good statistical behavior is *not* necessary to establish rigorous lower bounds. In other words, we are free to try to derive a rigorous lower bound $L \leq |S|$ using arbitrary i-uniform random sets, e.g., encoded via short constraints; our obligation to use distributions $\mathcal{D}_i$ with provably good statistical behavior (in order to establish a rigorous upper bound), is restricted to $i \approx \log_2(\delta L)$, i.e., for i hopefully not much less than $\log_2 |S|$.

– Our Algorithm 3 is very similar in spirit to the algorithms in [3, 5]. The reason we present it is because it allows a separation of the upper and lower bound issues, it affords a clean analysis, and it makes it clear that the only essential ingredient for approximate model counting are random sampling sets with good statistical behavior over extents of size at least δL. We discuss specific distributions of such sets and approaches to bounding B in Sections 8 and 9.

Theorem 3 *The output of Algorithm 3 is not in* $(1 \pm \delta)|S|$ *with probability at most* θ.

Algorithm 3 Given $0 \leq L \leq |S|$, $\delta, \theta > 0$ returns $Z \in (1 \pm \delta)|S|$ with probability $1 - \theta$

1: **if** $|S| < 4/\delta$ **then return** $|S|$ ▷ To do this: if $L < 4/\delta$ then run CryptoMiniSat with cutoff $4/\delta$
2: Let $\ell = \max\{0, \lfloor \log_2(\delta L/4) \rfloor\}$
3:
4: Find $B \geq \max_{\ell \leq i \leq n} \text{Boost}(\mathcal{D}_i, 2^i)$ ▷ If all $\{\mathcal{D}_i\}_{i=\ell}^n$ enjoy pairwise independence $B \leftarrow 1$
5:
6: $\delta \leftarrow \min\{\delta, 1/3\}$ ▷ See Remark 1
7: $\xi \leftarrow 8/\delta$
8: $b \leftarrow \lceil \xi + 2(\xi + \xi^2(B-1)) \rceil$ ▷ If $B = 1$, then $b = \lceil 24/\delta \rceil$
9: $t \leftarrow \lceil (2b^2/9) \ln(2n/\theta) \rceil$
10:
11: **for** i from ℓ to n **do**
12: $\quad Z_i \leftarrow 0$
13: $\quad$ **for** j from 1 to t **do**
14: $\quad\quad$ Sample $R_{i,j} \sim \mathcal{D}_i$
15: $\quad\quad$ Determine $Y_{i,j} = \min\{b, |S \cap R_{i,j}|\}$ ▷ Run CryptoMiniSat with cutoff b
16: $\quad\quad Z_i \leftarrow Z_i + Y_{i,j}$
17: $\quad A_i \leftarrow Z_i / t$
18: $j \leftarrow \max\{i \geq \ell : A_i \geq (1-\delta)(4/\delta)\}$
19: **return** $\max\{L, A_j 2^j\}$

Remark 1 Algorithm 3 can be modified to have two approximation parameters $\beta < 1$ and $\gamma > 1$ so that its output is between $\beta|S|$ and $\gamma|S|$. For this, both ξ and the criterion for choosing j in line 18 must be adapted to β, γ, θ. Here, for the benefit of exposition, we focus on the symmetric, high-accuracy case $\beta = 1 - \delta$, $\gamma = 1 + \delta$, $\delta \in (0, 1/3]$, allowing us to choose near-optimal ξ, b, t with relatively simple form.

6 Proof of Theorem 3

To prove Theorem 3 we will need the following tools.

Lemma 2 *Let* $X \geq 0$ *be an arbitrary integer-valued random variable. Write* $\mathbb{E}X = \mu$ *and* $\text{Var}(X) = \sigma^2$. *For some integer* $b \geq 0$, *define the random variable* $Y = \min\{X, b\}$. *For any* $\lambda > 0$, *if* $b \geq \mu + \lambda\sigma^2$, *then* $\mathbb{E}Y \geq \mathbb{E}X - 1/\lambda$.

Proof We start by recalling the well-known fact that if $Z \geq 0$ is an integer-valued random variable, then $\mathbb{E}Z = \sum_{j>0} \Pr[Z \geq j]$. Since both X, Y are integer-valued, using Chebychev's inequality to derive (2), we see that

$$\begin{aligned}
\mathbb{E}X - \mathbb{E}Y &= \sum_{j>b} \Pr[X \geq j] \\
&\leq \sum_{t=1}^{\infty} \Pr\left[X \geq \mu + \lambda\sigma^2 + t\right] \\
&= \sum_{t=1}^{\infty} \Pr\left[X \geq \mu + \sigma\left(\lambda\sigma + \frac{t}{\sigma}\right)\right] \\
&\leq \sum_{t=1}^{\infty} \frac{1}{(\lambda\sigma + t/\sigma)^2} \qquad (2) \\
&\leq \int_{t=0}^{\infty} \frac{1}{(\lambda\sigma + t/\sigma)^2}\, \mathrm{d}t \\
&= \frac{1}{\lambda}\ . \qquad \square
\end{aligned}$$

Lemma 3 *Let $\mathcal{D}$ be any i-uniform distribution on subsets of $\{0,1\}^n$. For any fixed set $S \subseteq \{0,1\}^n$, if $R \sim \mathcal{D}$ and $X = |S \cap R|$, then* $\mathrm{Var}(X) \leq \mathbb{E}X + (\mathrm{Boost}(\mathcal{D}, |S|) - 1)(\mathbb{E}X)^2$.

Proof Recall that $\mathrm{Var}(X) = \mathbb{E}X^2 - (\mathbb{E}X)^2$. Write $\mathbf{1}\{\cdot\}$ for the indicator function. Then

$$\begin{aligned}
\mathbb{E}X^2 &= \mathbb{E}\left(\sum_{\sigma\in S} \mathbf{1}\{\sigma \in R\}\right)^2 \\
&= \mathbb{E}\left(\sum_{\sigma,\tau\in S} \mathbf{1}\{\sigma, \tau \in R\}\right) \\
&= \sum_{\sigma,\tau\in S} \Pr[\sigma, \tau \in R] \\
&= \sum_{\sigma\in S} \Pr[\sigma \in R] + \sum_{\substack{\sigma,\tau\in S \\ \sigma\neq\tau}} \Pr[\sigma, \tau \in R] \\
&\leq \sum_{\sigma\in S} \Pr[\sigma \in R] + 2^{-2i}|S|(|S|-1)\mathrm{Boost}(\mathcal{D}, |S|) \\
&< \mathbb{E}X + \mathrm{Boost}(\mathcal{D}, |S|)(\mathbb{E}X)^2\ . \qquad \square
\end{aligned}$$

Let $q = \lfloor \log_2(\delta|S|/4) \rfloor$. Recall that if $|S| < 4/\delta$, the algorithm returns $|S|$ and exits. Therefore, we can assume without loss of generality that $q \geq 0$. Armed with Lemmata 2 and 3 we will prove the following propositions:

(a) The probability that $A_q 2^q$ is not in the range $(1 \pm \delta)|S|$ is at most $2e^{-9t/(2b^2)}$.
(b) The probability that $A_{q+1} 2^{q+1}$ is not in the range $(1 \pm \delta)|S|$ is at most $2e^{-9t/(2b^2)}$.
(c) If $A_q 2^q$ is in the range $(1 \pm \delta)|S|$, then the maximum in line 18 is at least q (deterministically).
(d) For each $i \geq q + 2$, the probability that the maximum in line 18 equals i is at most e^{-8t/b^2}.

Propositions (a)–(d) imply that the probability of failure is at most the sum of the probability of the bad event in (a), the bad event in (b), and the (at most) $n-2$ bad events in (d). The fact that each of these bad events concerns only one random variable A_j allows a significant acceleration of Algorithm 3, discussed in Section 7.

Fix any $i = q + k$, where $k \geq 0$. Let $X_{i,j} = |S \cap R_{i,j}|$ and write $\mathbb{E}X_{i,j} = \mu_i$ and $\mathrm{Var}(X_{i,j}) = \sigma_i^2$.

To establish propositions (a), (b) observe that the value ℓ defined in line 2 is at most q, since $L \leq |S|$, and that $|S| \geq 2^{q+1}$, since $\delta \leq 2$. Thus, since $\mathrm{Boost}(\mathcal{D}, M)$ is non-increasing in M,

$$\begin{aligned}\max_{k\in\{0,1\}} \mathrm{Boost}(\mathcal{D}_{q+k}, |S|) &\leq \max\{\mathrm{Boost}(\mathcal{D}_q, 2^{q+1}), \mathrm{Boost}(\mathcal{D}_{q+1}, 2^{q+1})\}\\ &\leq \max\{\mathrm{Boost}(\mathcal{D}_q, 2^{q}), \mathrm{Boost}(\mathcal{D}_{q+1}, 2^{q+1})\}\\ &\leq \max_{\ell\leq i\leq n} \mathrm{Boost}(\mathcal{D}_i, 2^i)\\ &\leq B\ .\end{aligned}$$

Therefore, we can apply Lemma 3 for $i \in \{q, q+1\}$ and conclude that $\sigma_i^2 \leq \mu_i + (B-1)\mu_i^2$ for such i. Since $\mu_i < 8/\delta$ for all $i \geq q$ while $\xi = 8/\delta$, we see that $b = \lceil \xi + 2(\xi + \xi^2(B-1))\rceil \geq \mu_i + 2\sigma_i^2$. Thus, we can conclude that for $i \in \{q, q+1\}$ the random variables $X_{i,j}, Y_{i,j}$ satisfy the conditions of Lemma 2 with $\lambda = 2$, implying $\mathbb{E}Y_{i,j} \geq \mathbb{E}X_{i,j} - 1/2$. Since Z_i is the sum of t independent random variables $0 \leq Y_{i,j} \leq b$ and $\mathbb{E}Z_i/t \geq \mu_i - 1/2$, we see that for $i \in \{q, q+1\}$ Hoeffding's inequality implies

$$\Pr[Z_i/t \leq (1-\delta)\mu_i] \leq \exp\left(-2t\left(\frac{\delta\mu_i - 1/2}{b}\right)^2\right)\ . \tag{3}$$

At the same time, since Z_i is the sum of t independent random variables $0 \leq Y_{i,j} \leq b$ and $\mathbb{E}Z_i/t \leq \mu_i$, we see that for all $i \geq q$, Hoeffding's inequality implies

$$\Pr[Z_i/t \geq (1+\delta)\mu_i] \leq \exp\left(-2t\left(\frac{\delta\mu_i}{b}\right)^2\right)\ . \tag{4}$$

To conclude the proof of propositions (a) and (b) observe that $\mu_{q+k} \geq 2^{2-k}/\delta$. Therefore, (3) and (4) imply that for $k \in \{0, 1\}$, the probability that $A_{q+k}2^{q+k}$ is outside $(1 \pm \delta)|S|$ is at most

$$2\exp\left(-2t\left(\frac{2^{2-k} - 1/2}{b}\right)^2\right) \leq 2\exp(-9t/(2b^2))\ .$$

To establish proposition (c) observe that if $A_q \geq (1-\delta)\mu_q$, then $A_q \geq (1-\delta)(4/\delta)$ and, thus, $j \geq q$. Finally, to establish proposition (d) observe that $\mu_i < 2/\delta$ for all $i \geq q+2$. Thus, for any such i, in order to have $\mu_i + w \geq (1-\delta)(4/\delta)$, it must be that $w > 2(1-2\delta)/\delta$, which, since $\delta \leq 1/3$, implies $w > 2$. Therefore, for every $k \geq 2$, the probability that $j = q + k$ is at most e^{-8t/b^2}.

Having established propositions (a)–(d) we argue as follows. If $A_{q+k}2^{q+k}$ is in the range $(1 \pm \delta)|S|$ for $k \in \{0, 1\}$ and smaller than $(1-\delta)(4/\delta)$ for $k \geq 2$, then the algorithm will report either $A_q 2^q$ or $A_{q+1}2^{q+1}$, both of which are in $(1 \pm \delta)|S|$. Thus, the probability that the algorithm's answer is outside the range $(1 \pm \delta)|S|$ is at most $2 \cdot 2e^{-9t/(2b^2)} + n \cdot e^{-8t/b^2}$ which, by our choice of t, is less than θ for all $n > 2$.

7 Nested sample sets

In Algorithm 3, for each $i \in [n]$ and $j \in [t]$, we sample each set $R_{i,j}$ independently from an i-uniform distribution on subsets of $\{0,1\}^n$. Imagine instead that we generate all random subsets of $\{0,1\}^n$ that we may need *before* we start Algorithm 3, in the following manner (in reality, we will only generate them as needed).

Algorithm 4 Generates t monotone decreasing sequences of sample sets

$R_{0,j} \leftarrow \{0,1\}^n$ for all $j \in [t]$
for i from 1 to n **do**
  for j from 1 to t **do**
    Select $R_{i,j} \subseteq R_{i-1,j}$ from a 1-uniform distribution on $R_{i-1,j}$

Organize now these sets in a matrix whose rows correspond to values of $0 \leq i \leq n$ and whose columns correspond to $j \in [t]$. It is easy to see that:

1. For each (row) $i \in [n]$:
   (a) Every set $R_{i,j}$ comes from an i-uniform distribution on $\{0,1\}^n$.
   (b) The sets $R_{i,1}, \ldots, R_{i,t}$ are mutually independent.
2. For each column $j \in [t]$:
   (a) $R_{0,j} \supseteq R_{1,j} \supseteq \cdots \supseteq R_{n-1,j} \supseteq R_{n,j}$.

With these new random sets, propositions (a)–(d) from the proof of Theorem 3, hold exactly as in the fully independent case, since for each fixed $i \in [n]$ the only relevant sets are those in row i and their distribution, per (1a)–(1b), did not change. At the same time, (2a) ensures that $Y_{1,j} \geq Y_{2,j} \geq \cdots \geq Y_{n,j}$ for every $j \in [t]$. As a result, $Z_1 \geq Z_2 \geq \cdots \geq Z_n$ and since $A_i = Z_i/t$, the indicator function for $A_i \geq (1-\delta)(4/\delta)$ is now *non-increasing*. This means that in order to compute j in line 18, instead of computing Z_i for i from ℓ to n, we can compute $A_\ell, A_{\ell+1}, A_{\ell+2}, A_{\ell+4}, A_{\ell+8}, \ldots$ until we encounter our first k such that $A_k < (1-\delta)(4/\delta)$, say at $k = \ell + 2^c$, for some $c \geq 0$. At that point, if $c \geq 1$, we can perform binary search for $j \in \{A_{\ell+2^{c-1}}, \ldots, A_{\ell+2^c-1}\}$ etc., so that the number of times the loop that begins in line 13 is executed is *logarithmic* instead of linear in $n - \ell$. Moreover, as we will see, the number of iterations t for the inner loop can now be reduced from $O(\ln(n/\theta))$ to $O(\ln(1/\theta))$.

Theorem 4 *Given $\theta > 0$, modify Algorithm 3 so that $t \leftarrow \lceil (2b^2/9)\ln(5/\theta)\rceil$ in line 9 and so that the sets $R_{i,j}$ in line 14 are generated by Algorithm 4. The output of the modified algorithm will lie in the range $(1 \pm \delta)|S|$ with probability at least $1-\theta$.*

Proof Observe that for any fixed i, since the sets $R_{i,1}, \ldots, R_{i,t}$ are mutually independent, (3) and (4) remain valid and, thus, propositions (a)–(c) hold. For proposition (d) we note that if the inequality $A_{q+k}2^{q+k} < (1-\delta)(4/\delta)$ holds for $k = 2$, then, by monotonicity, it holds for all $k \geq 2$. Thus, all in all, when monotone sequences of sample sets are used, the

probability that the algorithm fails is at most $4e^{-9t/(2b^2)} + e^{-8t/b^2}$, a quantity smaller than θ for $t \geq (2b^2/9)\ln(5/\theta)$. □

8 Homogeneous distributions

In Section 4 we saw that if $\mathcal{D}_i$ is an arbitrary i-uniform distribution on subsets of $\{0,1\}^n$, then we can not say much about Boost$(\mathcal{D},\cdot)$. On the other hand, when the subsets in the support of $\mathcal{D}_i$ are the codeword-sets of linear error-correcting codes, we can say a lot, due to the symmetries present in such codes. The most important implication of these symmetries is that for any pair $\sigma,\tau \in \{0,1\}^n$, the probability that both will be codewords depends only on their Hamming distance, a fact that can be thought of as a second-order uniformity (homogeneity). Definition 3 below captures this idea, along with the fact that this probability tends to decay with distance (at least up to distance $n/2$ – when all equations have even length, the complement of each codeword is also a codeword, in which case there is a symmetry around $n/2$).

Definition 3 Let $\mathcal{D}_i$ be any i-uniform distribution on subsets of $\{0,1\}^n$. Say that $\mathcal{D}_i$ is *homogeneous* if there exists a function f, called the *density* of $\mathcal{D}_i$, such that for all $\sigma,\tau \in \{0,1\}^n$, if $R \sim \mathcal{D}_i$, then $\Pr[\tau \in R \mid \sigma \in R] = f(\text{Hamming}(\sigma,\tau))$, where for all $j < n/2$ we have $f(j) \geq f(j+1)$ and $f(j) \geq f(n-j)$.

Homogeneity allows us to bound Boost by an analysis mimicking the one in [6]. For any $S \subset \{0,1\}^n$ and $\sigma \in S$, let $H_\sigma^S(d)$ denote the number of elements of S at Hamming distance d from σ. Recalling the definition of Boost in (5), we get (6) by i-uniformity and (7) by homogeneity,

$$\text{Boost}(\mathcal{D}_i, M) = \max_{\substack{S\subseteq\{0,1\}^n \\ |S|\geq M}} \frac{1}{|S|(|S|-1)} \sum_{\substack{\sigma,\tau\in S \\ \sigma\neq\tau}} \frac{\Pr[\sigma,\tau \in R]}{\Pr[\sigma \in R]\Pr[\tau \in R]} \tag{5}$$

$$= \max_{\substack{S\subseteq\{0,1\}^n \\ |S|\geq M}} \frac{2^i}{|S|(|S|-1)} \sum_{\sigma\in S} \sum_{\tau\in S-\sigma} \Pr[\tau \in S \mid \sigma \in S] \tag{6}$$

$$= \max_{\substack{S\subseteq\{0,1\}^n \\ |S|\geq M}} \frac{2^i}{|S|(|S|-1)} \sum_{\sigma\in S} \sum_{d=1}^{n} H_\sigma^S(d) f(d) \tag{7}$$

$$\leq \max_{\substack{S\subseteq\{0,1\}^n \\ |S|\geq M \\ \sigma\in S}} \frac{2^i}{|S|-1} \sum_{d=1}^{n} H_\sigma^S(d) f(d) \ . \tag{8}$$

Observe that passing from (7) to (8) amounts to allowing the sum in (8) to take its maximum value *simultaneously* for every $\sigma \in S$. Since f is non-increasing, this sum is maximized when S comprises two Hamming balls with complementary centers one of which is σ, making it clear that this simultaneous maximization is increasingly pessimistic as $|S|/2^n$ grows.

To get a tractable bound for Boost (reflecting the above worst-case scenario), we proceed as follows. Let $z \leq n/2$ be such that $(|S|-1)/2 = \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{z-1} + \alpha\binom{n}{z}$, for some $\alpha \in [0, 1)$. Homogeneity implies (9) and (12)

$$\begin{aligned}
\frac{\sum_{d=1}^{n} H_\sigma^S(d) f(d)}{|S|-1} &\leq \frac{\sum_{d=1}^{n/2} H_\sigma^S(d) f(d) + \sum_{d>n/2} H_\sigma^S(d) f(n-d)}{|S|-1} && (9)\\
&\leq \frac{2\left(\sum_{d=0}^{z-1}\binom{n}{d} f(d) + \alpha\binom{n}{z} f(z)\right)}{|S|-1} && (10)\\
&= \frac{\sum_{d=0}^{z-1}\binom{n}{d} f(d) + \alpha\binom{n}{z} f(z)}{\sum_{d=0}^{z-1}\binom{n}{d} + \alpha\binom{n}{z}} && (11)\\
&\leq \frac{\sum_{d=0}^{z-1}\binom{n}{d} f(d)}{\sum_{d=0}^{z-1}\binom{n}{d}} && (12)\\
&:= B(z)\ . && (13)
\end{aligned}$$

To bound $B(z)$ observe that since $f(j) \geq f(j+1)$ for $j < n/2$ it follows that $B(j) \leq B(j+1)$ for $j < n/2$. Thus, to bound $B(z)$ from above it suffices to bound z for below. Let $h : x \mapsto -x\log_2 x - (1-x)\log_2(1-x)$ be the binary entropy function and let $h^{-1} : [0, 1] \to [0, 1]$ map y to the smallest number x such that $h(x) = y$. It is well-known that $\sum_{d=0}^{z}\binom{n}{d} \leq 2^{nh(z/n)}$, for every integer $0 \leq z \leq n/2$. Therefore, $z \geq \lceil nh^{-1}(\log_2(|S|/2)/n)\rceil$, which combined with (8) and (13) implies the following.

Theorem 5 *If $\mathcal{D}_i$ is a homogeneous i-uniform distribution with density f, then*

$$\mathrm{Boost}(\mathcal{D}_i, M) \leq 2^i B\left(\left\lceil nh^{-1}\left(\frac{\log_2 M - 1}{n}\right)\right\rceil\right)\ , \tag{14}$$

where $B(z) = \sum_{d=0}^{z-1}\binom{n}{d} f(d) / \sum_{d=0}^{z-1}\binom{n}{d}$ and $h^{-1} : [0, 1] \to [0, 1]$ maps y to the smallest number x such that $h(x) = y$, where h is the binary entropy function.

To get a heuristic feel for (14) observe that if we did have pairwise independence, i.e., $f(d) = 2^{-i}$ for all $d > 0$, then for i, M such that $2^{-i}M \gg 1$, we would have

$$B(z) = \frac{1 + 2^{-i}\sum_{d=0}^{z-1}\binom{n}{d}}{1 + \sum_{d=0}^{z-1}\binom{n}{d}} \approx \frac{1 + 2^{-i}M}{1 + M} \approx 2^{-i}\ ,$$

in which case the bound in (14) reads $\mathrm{Boost}(\mathcal{D}_i, M) \approx 1$, as it should.

9 Low density parity check codes

Consider a random set $R = \{\sigma : A\sigma = b\}$, where $b \in \{0, 1\}^i$ is uniformly random, while $A \in \{0, 1\}^{i\times n}$ comes from a distribution to be specified. We would like the distribution of A to be such that both of the following hold:

(a) The (average) number of ones in each row of A is small.
(b) Even for σ, τ close in $\{0, 1\}^n$ it should be that $\Pr[\tau \in R \mid \sigma \in R]$ is small.

We have seen that (b) is trivial to achieve if we take A to be uniformly random, but this completely destroys (a), as the average number of ones in each row is $n/2$. Conversely,

setting each entry of A to 1 independently with probability $p < 1/2$, achieves (a), but sacrifices (b) dramatically as $p \to 0$. Moreover, (b) remains problematic even if we require each row of A (parity equation) to have exactly $\mathtt{r}$ ones, selected uniformly from all $\binom{n}{\mathtt{r}}$ possibilities (as long as $\mathtt{r}$ is not too large).

We confront this predicament by adding a very simple requirement, motivated by the seminal work of Sipser and Spielman on expander codes [13]:

each *column* of A must have have at least 3 ones

Observe that this *correlates* the entries of A, breaking with the i.i.d. constructions.

Explaining why this seemingly very benign modification has profound implications on the geometry of the set R is beyond the scope of this paper. At a high level, if each variable appears in at least 3 equations and $A\sigma = 0$, then changing a single variable in σ initiates a cascade of parity violations which, due to the randomness of A, typically, only settles $\Omega(n)$ away from σ, making R an error-correcting code. Low Density Parity Check (LDPC) matrices are a mainstay of modern coding theory [11], forming a vast field of active research. For the purposes of this paper it will suffice to consider the simplest possible construction, based on matrices $A \in \{0,1\}^{i \times n}$ where:

(i) Every column (variable) has exactly $\mathtt{l} \geq 3$ non-zero elements.
(ii) Every row (parity constraint) has exactly $\mathtt{r} = \mathtt{l}n/i \in$ non-zero elements.

Naturally, the requirement $\mathtt{l}n/i \in \mathbb{N}$ does not always hold, in which case some rows have $\lfloor \mathtt{l}n/i \rfloor$ variables, while the rest have $\lceil \mathtt{l}n/i \rceil$ variables, so that the average is $\mathtt{l}n/i$. To simplify discussion we ignore this point in the following.

Given $n, i,$and $\mathtt{l}$ a code is generated by selecting A uniformly at random[1] among all matrices satisfying (i)–(ii) and taking the set of codewords to be the set of solutions of the linear system $A\sigma = \mathbf{0}$. (While, for model counting we must also take the right hand side of the equation to be a uniformly random vector, when talking about the geometric properties of the set of codewords, due to symmetry we can assume without loss of generality that $b = \mathbf{0}$.) In particular, note that $\sigma = \mathbf{0}$ is always a solution of the system and, therefore, to discuss the remaining solutions (codewords) instead of referring to them by their distance from our reference solution $\sigma = \mathbf{0}$ we can refer to them by their weight, i.e., their number of ones.

It is well-known [11] that the expected number of codewords of weight w in a bi-regular LDPC code is given by the following (highly implicit) expression.

Lemma 4 (Average weight-distribution of regular LDPC code ensembles) *The expected number of codewords of weight w in a bi-regular LDPC code with n variables and i parity equations, where each variable appears in $\mathtt{l}$ equations and each equation includes $\mathtt{r}$ variables equals the coefficient of $x^{w\mathtt{l}}$ in the polynomial*

$$\binom{n}{w} \frac{\left(\sum_i \binom{r}{2i} x^{2i}\right)^{n\frac{\mathtt{l}}{\mathtt{r}}}}{\binom{n\mathtt{l}}{w\mathtt{l}}} \quad . \tag{15}$$

We will denote the quantity described in Lemma 4 by codewords(w).

[1] This can be done by selecting a uniformly random permutation of size $[\mathtt{l}n]$ and using it to map each of the $\mathtt{l}n$ non-zeros to equations; when $\mathtt{l}, \mathtt{r} \in O(1)$, the variables in each equation will be distinct with probability $\Omega(1)$, so that a handful of trials suffice to generate a matrix as desired.

9.1 The lumpiness of LDPC codes

Let $\mathcal{D}_i$ be the i-uniform distribution that results when $R = \{\sigma : A\sigma = b\}$, where A is selected uniformly at random among all matrices satisfying (i)–(ii) and b is uniformly random. The row- and column-symmetry in the distribution of A implies that for any pair σ, τ having Hamming distance d, the probability they are both in R is $2^{-i} f(d)$ where $f(d) = \text{codewords}(d)/\binom{n}{d}$, making f the density of $\mathcal{D}_i$. It is also easy to see that if $\mathtt{r}$ is even, then $\text{codewords}(d) = \text{codewords}(n-d)$ for all d and to simplify exposition we will restrict to that case.

We are left to establish that $f(j) \geq f(j+1)$ for all $0 \leq j < n/2$. Unfortunately, this is not strictly true for a trivial reason: in the vicinity of $n/2$ the function f is non-monotone, exhibiting minuscule fluctuations (due to finite-scale-effects) around its globally minimum value at $n/2$. While this prevents us from applying Theorem 5 immediately, it is easy to overcome. Specifically, for the proof of Theorem 5 to go through it is enough that $f(j) \geq f(j+1)$ for all $0 \leq j < z$ (instead of all $0 \leq j < n/2$), something which for most sets of interest holds, as $z \ll n/2$. Thus, in order to provide a rigorous upper bound on Boost, as required in Algorithm 3, it is enough to verify the monotonicity of f up to z in the course of evaluating $B(z)$. This is precisely what we did for $n \in \{100, 110, \ldots, 200\}$, $\log_2 M = 2n/5$, $\mathtt{l} = 8$, and $\mathtt{r} = 20$. We present the corresponding bounds for Boost from (14) in Table 1 below.

Several comments are due here. First, the non-monotonicity of the bound is due to the interaction of several factors in (14), most anomalous of which is the ceiling. Second, it is instructive to compare the number of solver invocations necessary for a rigorous approximation based on these bounds vs. what would be needed if we had pairwise independence for some reasonable parameter values. For example, for $\delta = 1/3$, $\theta = 10^{-3}$, writing $B = 1+q$, it is not hard to see that the number of iterations in line 9 of Algorithm 3 increases by a factor close to $(16q)^2$. While this is a rather dispiriting slowdown, there are two important points to keep in mind.

- The bounds in Table 1 enable rigorous model count approximation using systems with $40 - 80$ parity equations of length 20 over $n \in [100, 200]$ variables. While the number of solver invocations is very large, each invocation is quite fast and the invocations can be run in parallel. In contrast, when equations of length $n/2$ are used, systems of this size are completely outside the reach of CryptoMiniSAT.
- The bounds in Table 1 appear to be *extremely* pessimistic. As we demonstrate experimentally in Section 10, the statistical behavior of LDPC constraints in practice appears *indistinguishable* from that of long parity constraints.

10 Experiments

The goal of this section is to demonstrate *empirically* the promise of using systems of parity equations corresponding to LDPC codes for model counting. To do this we will employ such systems while making *far fewer* solver invocations than what is mandated by our theoretical

Table 1 Upper bounds for Boost for equations of length 20

n	100	110	120	130	140	150	160	170	180	190	200
Boost	75	50	35	26	134	89	60	44	34	154	105

bounds for a rigorous approximation. The reason we do this is because we believe that while the error-probability analysis of Theorems 3 and 4 is not too far off the mark, the same can not be said for Theorem 5, providing our rigorous upper bound on Boost. So, until better such bounds are derived, we can try to demonstrate the good statistical properties of these systems empirically.

To make the demonstration as transparent as possible, we modify the state of the art approximate model counter ApproxMC2, which uses long parity equations and always gives rigorous results, as follows.

- We incorporate Algorithm 2 with LDPC constraints and run it first. The lower bound derived replaces the (trivial) original starting point for ApproxMC2.
- We replace all systems of long parity equations used by ApproxMC2 with sparse systems corresponding to LDPC codes.

Algorithm 2 is invoked at most once, while the change in the systems of equations is entirely encapsulated in the part of the code generating the random systems. No other changes to ApproxMC2 (AMC2) were made.

Naturally, **the results returned by the modified algorithm are not rigorous**. Nevertheless, on all formulas for which the original AMC2 terminates, we use its (rigorous) output as a control for the output of the modified algorithm. Indeed, to illuminate the bigger picture, besides AMC2 we also included in the comparison the *exact* model counter `sharpSAT` of Thurley [16], and a modification of the modified algorithm in which the LDPC constraints are replaced by random sparse constraints, where each equation involves each variable independently with probability $p = 1/2^j$, for $j = 2, \ldots, 5$. (Recall that AMC2 uses $j = 1$). The resulting algorithm is thus similar to the main algorithm of [7], which also uses sparse systems with independent entries.

We consider the same 387 formulas as [4]. Among these are 2 unsatisfiable formulas, which we removed. We also removed 9 formulas that were only solved by `sharpSAT` and 10 formulas whose number of solutions (and, thus, equations) is so small that the LDPC equations devolve into long XOR equations. Of the remaining 366 formulas, `sharpSAT` solves 233 in under 1 second, in every case significantly faster than all approximate methods. At the other extreme, 46 formulas are not solved by any method within the given time limits, namely 8 hours per method-formula pair (and 50 minutes for each solver invocation for the sampling based algorithms). We report on the remaining 87, most interesting, formulas. All experiments were run on a modern cluster of 13 nodes, each with 16 cores and 128GB RAM.

Our findings can be summarized as follows:

1. The LDPC-modified version of AMC2 did not time out on *any* formula. In contrast, `sharpSAT` timed out on 38% of the formulas and AMC2 on 62%.
2. In *every* formula the count returned by the LDPC-modified version of AMC2 is very close to the count returned by `sharpSAT` and/or AMC2.
3. The counts with $p = 1/4$ are as accurate as with $p = 1/2$. But for $p \leq 1/8$, the counts are very often significantly wrong and we don't report results for such p.
4. The LDPC-modified version of AMC2 is faster than AMC2 in *all but one* formulas, the speedup typically exceeding 10x and often exceeding 50x.
5. When both `sharpSAT` and the LDPC-modified version of AMC2 terminate, more often than not `sharpSAT` is faster. That said, speed victories of a factor of 50x occur for both algorithms.

In Table 2, the first four numerical columns report the binary logarithm of the estimate of $|S|$ returned by each algorithm. The next four columns report the time taken to produce the

estimate, in seconds. We note that several of the 87 formulas come with a desired *sampling set*, i.e., a subset of variables V such that the goal is to count the size of the projection of the set of all models on V. Since, unlike AMC2, `sharpSAT` does not provide such constrained counting functionality, to avoid confusion, we do not report a count for `sharpSAT` for these formulas, writing "—" instead. Timeouts are reported as "NA".

Table 2 Estimates of $\log_2$ |# models| by different methods, followed by their corresponding time in seconds

Formula name	#SAT	LDPC	AMC2	1/4	#SAT	LDPC	AMC2	1/4
jburnim_morton.sk_13_530	NA	248.49	NA	NA	NA	27826.4	NA	NA
blasted_case37	NA	151.02	NA	NA	NA	4149.9	NA	NA
blasted_case_0_b12_even1	NA	147.02	NA	NA	NA	1378.8	NA	NA
blasted_case_2_b12_even1	NA	147.02	NA	NA	NA	1157.5	NA	NA
blasted_case42	NA	147.02	NA	NA	NA	1008.0	NA	NA
blasted_case_1_b12_even1	NA	147.02	NA	NA	NA	1102.0	NA	NA
blasted_case_0_b12_even2	NA	144.02	NA	NA	NA	881.6	NA	NA
blasted_case_1_b12_even2	NA	144.02	NA	NA	NA	1156.3	NA	NA
blasted_case_2_b12_even2	NA	144.02	NA	NA	NA	1050.5	NA	NA
blasted_case_3_4_b14_even	NA	138.02	NA	NA	NA	293.4	NA	NA
blasted_case_1_4_b14_even	NA	138.02	NA	NA	NA	472.6	NA	NA
log2.sk_72_391	—	136.00	NA	NA	—	12811.1	NA	NA
blasted_case1_b14_even3	NA	122.02	NA	NA	NA	169.6	NA	NA
blasted_case_2_b14_even	NA	118.02	NA	NA	NA	89.2	NA	NA
blasted_case3_b14_even3	NA	118.02	NA	NA	NA	107.7	NA	NA
blasted_case_1_b14_even	NA	118.02	NA	NA	NA	94.7	NA	NA
partition.sk_22_155	NA	107.17	NA	NA	NA	5282.3	NA	NA
sc_tr_delete4.sb.pl.sk_4_114	—	105.09	NA	NA	—	708.4	NA	NA
blasted_case140	NA	103.02	NA	NA	NA	1869.0	NA	NA
sc_tr_search.sb.pl.sk_11_136	NA	96.46	NA	NA	NA	3314.2	NA	NA
s1423a_7_4	90.59	90.58	NA	NA	6.2	32.4	NA	NA
s1423a_3_2	90.16	90.17	NA	NA	5.7	28.3	NA	NA
s1423a_15_7	89.84	89.83	NA	NA	13.6	44.8	NA	NA
sc_tr_delete1.sb.pl.sk_3_114	—	89.15	NA	NA	—	431.3	NA	NA
blasted_case_0_ptb_2	NA	88.02	NA	NA	NA	463.6	NA	NA
blasted_case_0_ptb_1	NA	87.98	NA	NA	NA	632.0	NA	NA
sc_tr_delete2.sb.pl.sk_8_114	—	86.46	NA	NA	—	210.3	NA	NA
sc_aig_traverse.sb.pl.sk_5_102	NA	86.39	NA	NA	NA	3230.0	NA	NA
54.sk_12_97	82.50	81.55	NA	NA	20.4	235.8	NA	NA
blasted_case_0_b14_1	79.00	79.09	NA	NA	28.8	33.5	NA	NA
blasted_case_2_ptb_1	NA	77.02	NA	NA	NA	10.1	NA	NA
blasted_case_1_ptb_1	NA	77.02	NA	NA	NA	9.5	NA	NA
blasted_case_1_ptb_2	NA	77.02	NA	NA	NA	17.8	NA	NA
blasted_case_2_ptb_2	NA	77.00	NA	NA	NA	25.0	NA	NA
blasted_squaring70	66.00	66.04	NA	NA	5822.7	87.7	NA	NA
blasted_case19	66.00	66.02	NA	NA	25.1	6.9	NA	NA
blasted_case20	66.00	66.02	NA	NA	2.0	4.4	NA	NA
blasted_case15	65.00	65.02	NA	NA	172.3	12.4	NA	NA
blasted_case10	65.00	65.02	NA	NA	209.8	8.8	NA	NA

Table 2 (continued)

Formula name	#SAT	LDPC	AMC2	1/4	#SAT	LDPC	AMC2	1/4
blasted_TR_b12_2_linear	NA	63.93	NA	NA	NA	1867.1	NA	NA
blasted_case12	NA	62.02	NA	NA	NA	21.5	NA	NA
blasted_case49	61.00	61.02	NA	NA	8.9	15.6	NA	NA
blasted_TR_b12_1_linear	NA	59.95	NA	NA	NA	767.9	NA	NA
sc_tr_insert_insert.sb.pl.sk_3_68	—	51.86	NA	NA	12.1	54.3	NA	NA
blasted_case18	NA	51.00	NA	NA	NA	16.7	NA	NA
blasted_case14	49.00	49.07	NA	NA	117.2	7.6	NA	NA
blasted_case9	49.00	49.02	NA	NA	123.6	7.1	NA	NA
blasted_case61	48.00	48.02	NA	NA	154.2	6.7	NA	NA
ProjectService3.sk_12_55	—	46.55	46.58	46.55	—	12.9	273.4	267.1
blasted_case145	46.00	46.02	NA	46.02	29.2	8.4	NA	5570.4
blasted_case146	46.00	46.02	46.02	NA	29.3	4.8	9528.6	NA
ProcessBean.sk_8_64	—	42.83	42.91	42.83	—	17.0	323.2	207.3
blasted_case106	42.00	42.02	42.02	42.02	10.2	3.3	325.0	14728.3
blasted_case105	41.00	41.00	41.04	NA	7.5	4.0	368.5	NA
blasted_squaring16	40.76	40.83	NA	41.07	99.4	50.3	NA	1633.3
blasted_squaring14	40.76	40.70	NA	41.00	102.1	34.3	NA	2926.5
blasted_squaring12	40.76	40.61	NA	41.00	117.3	39.6	NA	1315.6
blasted_squaring7	38.00	38.29	38.00	38.11	45.4	34.9	432.4	263.2
blasted_squaring9	38.00	38.04	37.98	38.15	36.3	24.2	489.8	238.6
blasted_case_2_b12_2	38.00	38.02	38.02	38.00	29.3	4.4	186.8	87.2
blasted_case_0_b11_1	38.00	38.02	38.02	38.04	45.5	2.5	190.4	180.7
blasted_case_0_b12_2	38.00	38.02	38.02	38.02	29.2	3.8	181.1	69.9
blasted_case_1_b11_1	38.00	38.02	38.02	37.81	45.2	3.5	159.5	119.2
blasted_case_1_b12_2	38.00	38.02	38.02	38.02	30.6	2.9	185.3	80.0
blasted_squaring10	38.00	38.02	37.91	38.04	17.6	32.0	415.1	221.7
blasted_squaring11	38.00	37.95	38.02	38.09	19.8	19.7	470.1	207.3
blasted_squaring8	38.00	37.93	38.09	39.00	18.6	28.0	431.5	727.8
sort.sk_8_52	—	36.43	36.43	36.36	—	92.0	339.2	156.8
blasted_squaring1	36.00	36.07	36.07	36.00	6.6	20.0	367.8	156.9
blasted_squaring6	36.00	36.04	36.00	35.93	8.5	17.1	429.1	170.5
blasted_squaring3	36.00	36.02	36.02	36.02	7.7	18.7	397.3	198.5
blasted_squaring5	36.00	35.98	36.02	36.04	8.5	28.8	384.0	228.2
blasted_squaring2	36.00	35.98	36.00	36.07	7.5	30.6	411.5	195.8
blasted_squaring4	36.00	35.95	36.04	35.98	7.9	23.2	469.8	180.0
compress.sk_17_291	NA	34.00	NA	NA	NA	1898.2	NA	NA
listReverse.sk_11_43	NA	32.00	32.00	32.00	NA	2995.3	2995.3	2995.7
enqueueSeqSK.sk_10_42	NA	31.49	31.39	31.43	NA	67.6	252.0	124.6
blasted_squaring29	26.25	26.36	26.29	26.39	1.3	42.7	218.7	75.2
blasted_squaring28	26.25	26.32	26.36	26.36	1.9	57.6	185.1	59.0
blasted_squaring30	26.25	26.25	26.29	26.17	1.6	40.9	179.8	60.8
tutorial3.sk_4_31	NA	25.29	25.32	25.25	NA	3480.5	19658.2	2414.7
blasted_squaring51	24.00	24.11	24.15	24.07	1.6	4.8	49.3	5.3

Table 2 (continued)

Formula name	#SAT	LDPC	AMC2	1/4	#SAT	LDPC	AMC2	1/4
blasted_squaring50	24.00	23.86	24.00	24.02	1.3	4.7	54.2	5.1
N_S_Impl2.sk_10_36	—	22.64	22.49	22.55	—	13.7	29.6	9.6
karatsuba.sk_7_41	—	20.36	NA	20.52	—	24963.0	NA	19899.0
LoginService.sk_20_34	—	19.49	19.39	19.43	—	28.1	33.0	20.7
LoginService2.sk_23_36	—	17.55	17.43	17.43	—	72.9	40.8	32.6

11 Conclusions

We have separated the problem of probabilistically approximating the size of the set, $S(F)$, of all models of a CNF formula F, into two problems and shown that deriving rigorous lower bounds does not require controlling the geometry of the random subsets used for sampling. At the same time, we have shown that to derive rigorous upper bounds it is enough to control the geometry of the sampling subsets over extents of size similar to $|S(F)|$, a task that gets easier with increasing model density.

Motivated by this separation we introduced the idea of using as sampling subsets the codeword sets of Low Density Parity Check codes and proved that they yield rigorous model counts, with a number of solver invocations that grows (quickly) with their deviation from pairwise independence. Finally, we demonstrated experimentally that this deviation may be much smaller than indicated by our bounds for it.

We believe that developing tighter mathematical bounds for the deviation of LDPC codes from pairwise independence is a fertile direction for future research and that success in this endeavor would greatly promote the adoption of such codes as general sampling devices. At the same time, since rigorous lower bounds for model counting do not require further theoretical advances, another direction for future research is the exploration of more sophisticated LDPC ensembles, e.g., to see if properties useful in the context of communications are also helpful in the context of model counting.

Acknowledgements We are grateful to Kuldeep Meel and Moshe Vardi for sharing their code and formulas and for several valuable conversations. We also thank Zayd Hammoudeh, Ben Sherman, and Kostas Zampetakis for several comments on earlier versions.

References

1. Achim, T., Sabharwal, A., Ermon, S. (2016). Beyond parity constraints: Fourier analysis of hash functions for inference. In M. Balcan, & K.Q. Weinberger (Eds.) *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, JMLR Workshop and Conference Proceedings*, (Vol. 48 pp. 2254–2262). JMLR.org. http://jmlr.org/proceedings/papers/v48/achim16.html.
2. Chakraborty, S., Fremont, D.J., Meel, K.S., Seshia, S.A., Vardi, M.Y. (2014). Distribution-aware sampling and weighted model counting for SAT. In Brodley, C.E., & Stone, P. (Eds.) *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (pp. 1722–1730). Québec City: AAAI Press. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8364.
3. Chakraborty, S., Meel, K.S., Vardi, M.Y. (2013). A scalable approximate model counter. In Schulte, C. (Ed.) *Principles and Practice of Constraint Programming - 19th International Conference, CP*

2013, Uppsala, Proceedings, Lecture Notes in Computer Science, (Vol. 8124 pp. 200–216): Springer. https://doi.org/10.1007/978-3-642-40627-0_18.
4. Chakraborty, S., Meel, K.S., Vardi, M.Y. (2016). Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In Kambhampati, S. (Ed.) *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016* (pp. 3569–3576). New York: IJCAI/AAAI Press. http://www.ijcai.org/Abstract/16/503.
5. Ermon, S., Gomes, C.P., Sabharwal, A., Selman, B. (2013). Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proceedings of the 30th international conference on machine learning (ICML)*.
6. Ermon, S., Gomes, C.P., Sabharwal, A., Selman, B. (2014). Low-density parity constraints for hashing-based discrete integration. In *Proceedings of the 31st international conference on machine learning (ICML)* (pp. 271–279).
7. Gomes, C.P., Hoffmann, J., Sabharwal, A., Selman, B. (2007). Short XORs for model counting: From theory to practice. In *Theory and applications of satisfiability testing (SAT)* (pp. 100–106).
8. Gomes, C.P., Sabharwal, A., Selman, B. (2006). Model counting: a new strategy for obtaining good bounds. In *Proceedings of the 21st national conference on artificial intelligence (AAAI)* (pp 54–61).
9. Ivrii, A., Malik, S., Meel, K.S., Vardi, M.Y. (2016). On computing minimal independent support and its applications to sampling and counting. *Constraints*, *21*(1), 41–58. https://doi.org/10.1007/s10601-015-9204-z.
10. Meel, K.S., Vardi, M.Y., Chakraborty, S., Fremont, D.J., Seshia, S.A., Fried, D., Ivrii, A., Malik, S. (2016). Constrained sampling and counting: Universal hashing meets sat solving. In *Workshops at the thirtieth AAAI conference on artificial intelligence*.
11. Richardson, T., & Urbanke, R. (2008). *Modern coding theory*. New York: Cambridge University Press.
12. Sipser, M. (1983). A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM symposium on theory of computing (STOC)* (pp 330–335).
13. Sipser, M., & Spielman, D.A. (1996). Expander codes. *IEEE Transactions on Information Theory*, *42*(6), 1710–1722. https://doi.org/10.1109/18.556667.
14. Soos, M. (2009). Cryptominisat–a sat solver for cryptographic problems. http://www.msoos.org/cryptominisat4.
15. Stockmeyer, L. (1985). On approximation algorithms for Ṗ. *SIAM Journal on Computing*, *14*(4), 849–861.
16. Thurley, M. (2006). Sharpsat: Counting models with advanced component caching and implicit bcp. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing, SAT'06* (pp. 424–429). Berlin: Springer. https://doi.org/10.1007/11814948_38.
17. Valiant, L., & Vazirani, V. (1986). NP is as easy as detecting unique solutions. *Theoretical Computer Science*, *47*, 85–93.
18. Zhao, S., Chaturapruek, S., Sabharwal, A., Ermon, S. (2016). Closing the gap between short and long xors for model counting. In Schuurmans, D., & Wellman, M.P. (Eds.) *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 3322–3329). Phoenix: AAAI Press. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12546.