Approximate Model Counting, Sparse XOR Constraints and Minimum Distance

Michele Boreale[1(✉)] and Daniele Gorla[2]

[1] Dip. Statistica, Informatica, Applicazioni (DiSIA), Università di Firenze, Florence, Italy
michele.boreale@unifi.it
[2] Department of Computer Science, Università di Roma "La Sapienza", Rome, Italy

Abstract. The problem of counting the number of models of a given Boolean formula has numerous applications, including computing the leakage of deterministic programs in Quantitative Information Flow. Model counting is a hard, #P-complete problem. For this reason, many approximate counters have been developed in the last decade, offering formal guarantees of confidence and accuracy. A popular approach is based on the idea of using random XOR constraints to, roughly, successively halving the solution set until no model is left: this is checked by invocations to a SAT solver. The effectiveness of this procedure hinges on the ability of the SAT solver to deal with XOR constraints, which in turn crucially depends on the length of such constraints. We study to what extent one can employ sparse, hence short, constraints, keeping guarantees of correctness. We show that the resulting bounds are closely related to the geometry of the set of models, in particular to the minimum Hamming distance between models. We evaluate our theoretical results on a few concrete formulae. Based on our findings, we finally discuss possible directions for improvements of the current state of the art in approximate model counting.

Keywords: Model counting · Approximate counting · XOR sampling

1 Introduction

#SAT (aka *model-counting*) is the problem of counting the number of satisfying assignments of a given Boolean formula and is a #P-complete problem. Indeed, every NP Turing machine can be encoded as a formula whose satisfying assignments correspond to the accepting paths of the machine [24]. Thus, model-counting is harder than satisfiability: #SAT is indeed intractable in cases for which SAT is tractable (e.g., sets of Horn clauses or sets of 2-literal clauses) [25]. Still, there are cases in which model-counting is tractable (e.g., OBDDs and d-DNNFs). For a very good overview of the problem and of some approaches to it see [14].

Our interest in model counting originates from its applications in the field of Quantitative Information Flow (QIF) [8,20]. Indeed, a basic result in QIF is

M. S. Alvim et al. (Eds.): Palamidessi Festschrift, LNCS 11760, pp. 363–378, 2019.
https://doi.org/10.1007/978-3-030-31175-9_21

that the maximum min-entropy leakage of a deterministic program is $\log_2 k$, with k the number of distinct outputs the program can return [20], varying the input. If the program is modeled as a Boolean formula, then computing its leakage reduces to #SAT, specifically to computing the number of models of the formula obtained by existentially projecting out the non-output variables; see [3,17].

Over the years, several *exact* counting algorithms have been put forward and implemented, such as, among others, [12,17,19,23], with applications to QIF [16]. The problem with exact counters is that, although performing reasonably well when certain parameters of the formula – size, number of variables, number of clauses – are relatively small, they rapidly go out of memory as these parameters grow.

For this reason, *approximate* counters have more and more been considered. Indeed, in many applications, the exact count of models is not required: it may suffice to provide an estimate, as long as the method is quick and it is equipped with a formal guarantee of correctness. This is typically the case in QIF, where knowing the exact count within a factor of η is sufficient to estimate leakage within $\log_2 \eta$ bits. For probabilistic counters, correctness is usually expressed in terms of two parameters: *accuracy* – the desired maximum difference between the reported and the true count; and *confidence* – the probability that the reported result is actually within the specified accuracy.

We set ourselves in the line of research pioneered by [25] and followed, e.g., by [1,6,13,17]. The basic idea of a probabilistic model counting algorithm is the following (Sect. 2): given a formula ϕ in the Boolean variables $y_1, \ldots, y_m$, one chooses at random $\langle a_0, \ldots, a_m \rangle \in \{0,1\}^{m+1}$. The resulting *XOR constraint* $a_0 = a_1 y_1 \oplus \cdots \oplus a_m y_m$ splits evenly the set of models of ϕ into two parts: those satisfying the constraint and those not satisfying it. If one independently generates s such constraints $c_1, \ldots, c_s$, the formula $\phi' \stackrel{\triangle}{=} \phi \wedge c_1 \wedge \cdots \wedge c_s$ has an expected $\frac{N}{2^s}$ models, where N is the number of models of ϕ (i.e., the number we aim at estimating). If ϕ' is still satisfiable, then with high probability $N \geq 2^s$, otherwise $N < 2^s$. By repeating this process, one can arrive at a good estimate of N. This procedure can be implemented by relying on any SAT-solver capable of dealing with XOR constraints, e.g CryptoMiniSat [22]; or even converting the XOR constraints into CNF before feeding ϕ' to the SAT solver. In any case, the branching factor associated with searching for models of ϕ' quickly explodes as the *length* (number of variables) of the XOR constraints grows. The random generation outlined above will lead to an expected length of $\frac{m}{2}$ for each constraint, making the procedure not easily scalable as m grows.

In the present paper, we study under what conditions one can employ sparse, hence shorter, constraints, keeping the same guarantees of correctness. We generalize the results of [9] to arrive at an improved understanding of how sparsity is related to minimum distance between models, and how this affects the counting procedure. Based on these results, we also suggest a possible direction for a new counting methodology based on the use of Low Density Parity Check

(LDPC) codes [11,18]; however, we leave a through experimentation with this new methodology for future work.

The main point is to generate the coefficients $a_1, \ldots, a_m$ according to a probability value $\lambda \in \left(0, \frac{1}{2}\right]$, rather than uniformly. This way, the constraints will have an average length of $\lambda m \leq \frac{m}{2}$ each. Basically, the correctness guarantees of the algorithm depend, via the Chebyshev inequality, on keeping the variance of the number of models of ϕ', the formula obtained by joining the constraints, below a certain threshold. A value of the density λ that achieves this is said to be *feasible* for the formula. In our main result (Sect. 3), we provide a bound on the variance that also depends on the minimum Hamming distance d between the formula's models: a larger d yields a smaller variance, hence smaller feasible λ's. Our bound essentially coincides with that of [9] for $d = 1$. Therefore, in principle, a lower bound on the minimum distance can be used to obtain tighter bounds on λ, making the XOR constraints shorter and the counting procedure pragmatically more efficient.

We will show this phenomenon at work (Sect. 4) on some formulae where the value of d is known by construction, comparing our results with the state of the art model counter ApproxMC3 [21]. These considerations also suggest that, if no information on d is available, one can encode the formula's models using an error correcting code with a known minimum distance. We will briefly discuss the use of LDPC codes to this purpose, although at the moment we have no experimental results available in this respect. A comparison with recent related work concludes the paper (Sect. 5). For reasons of space, all proofs have been sketched and are fully available in [4].

2 A General Counting Algorithm

In what follows, we let $\phi(y)$, or just ϕ, denote a generic boolean formula with boolean variables $y = (y_1, ..., y_m)$ and $m \geq 1$.

2.1 A General Scheme

According to a well-known general scheme [13], the building block of a statistical counting procedure is a probabilistic decision algorithm: with high probability, this algorithm correctly decides whether the cardinality of the set is, or is not, below a given threshold 2^s, within some tolerance factors, given by the slack parameters α and β below.

Definition 1 (#SAT decision algorithm). *Let* $0 \leq \delta < frac12$ *(error probability),* $\alpha > 1$ *and* $\beta > 1$ *(two slack parameters) be three reals. An* (α, β, δ)-decision algorithm *(for #SAT) is a probabilistic algorithm* $A(\cdot, \cdot)$*, taking a pair of an integer* s *and a boolean formula* ϕ*, and returning either* 1 *(meaning '*$\#\phi \geq 2^{s-\alpha}$*') or* 0 *(meaning '*$\#\phi \leq 2^{s+\beta}$*') and such that for each integer* $s \geq 0$ *and formula* ϕ*:*

1. $\#\phi > 2^{s+\beta}$ *implies* $\Pr(A(s,\phi)=0) \leq \delta$;
2. $\#\phi < 2^{s-\alpha}$ *implies* $\Pr(A(s,\phi)=1) \leq \delta$.

The use of two different slack parameters in the definition above is justified by the need of stating formal guarantees about the outcome of the algorithm, while keeping the precision of the algorithm as high as possible.

As usual, we can boost the confidence in the reported answer, and get an arbitrarily small error probability, by running $A(s,\phi)$ several times independently. In particular, consider the algorithm $RA_t(s,\phi)$ obtained by running $A(s,\phi)$ an odd $t \geq 1$ number times independently, and then reporting the majority answer. Call Err the event that RA_t reports a wrong answer; then,

$$\begin{aligned}\Pr(Err) &= \Pr\left(\text{at least } \left\lceil \frac{t}{2} \right\rceil \text{ runs of } A(s,\phi) \text{ report the wrong answer}\right)\\ &= \sum_{k=\lceil \frac{t}{2} \rceil}^{t} \Pr(\text{exactly } k \text{ runs of } A(s,\phi) \text{ report the wrong answer})\\ &= \sum_{k=\lceil \frac{t}{2} \rceil}^{t} \binom{t}{k} p^k (1-p)^{t-k} \qquad (1)\end{aligned}$$

where

$$\begin{aligned}p &\stackrel{\triangle}{=} \Pr(A(s,\phi) \text{ reports the wrong answer})\\ &= \begin{cases} \Pr(A(s,\phi)=0) & \text{if } \#\phi > 2^{s+\beta} \\ \Pr(A(s,\phi)=1) & \text{if } \#\phi < 2^{s-\alpha} \end{cases}\\ &\leq \delta \qquad (2)\end{aligned}$$

Now, replacing (2) in (1), we obtain

$$\Pr(Err) \leq \sum_{k=\lceil \frac{t}{2} \rceil}^{t} \binom{t}{k} \delta^k (1-\delta)^{t-k} \qquad (3)$$

Let us call $\Delta(t,\delta)$ the right hand side of (3); then, RA_t is an $(\alpha,\beta,\Delta(t,\delta))$-decision algorithm whenever A is an (α,β,δ)-decision algorithm.

We now show that any (α,β,δ)-decision algorithm A for #SAT can be used as a building block for a counting algorithm, $C_A(\phi)$, that determines an interval $[\ell,u]$ such that $\lfloor 2^\ell \rfloor \leq \#\phi \leq \lceil 2^u \rceil$ with high probability. Informally, starting with an initial interval $[-1,m]$, the algorithm C_A performs a binary search, using A to decide which half of the current interval $\log_2(\#\phi)$ lies in. The search stops when the current interval cannot be further narrowed, taking into account the slack parameters α and β, or when a certain predefined number of iterations is reached. Formally, let $I_0 \stackrel{\triangle}{=} [-1,m]$. Assume $k > 0$ and $I_k = [l_k,u_k]$, then:

(a) if $u_k - l_k \leq 2\max(\alpha,\beta)+1$ or $k = \lceil \log_2(m) \rceil$, then return I_k;

(b) otherwise, let $s = round\left(\frac{u_k+l_k}{2}\right)$; if $A(s,\phi) = 0$ then $I_{k+1} \stackrel{\triangle}{=} [l_k, s+\beta]$ otherwise $I_{k+1} \stackrel{\triangle}{=} [s-\alpha, u_k]$.

Theorem 1. *Let A be a (α,β,δ)-decision algorithm. Then:*

1. *$C_A(\phi)$ terminates in $k \leq \lceil \log_2 m \rceil$ iterations returning an interval $I_k = [l,u]$ such that $u - l \leq 2\max(\alpha,\beta) + 2$;*
2. *The probability that $\#\phi \notin [\lfloor 2^l \rfloor, \lceil 2^u \rceil]$ is at most $\lceil \log_2 m \rceil \delta$.*

Proof (Sketch). If the algorithm terminates because $u_k - l_k \leq 2\max(\alpha,\beta)+1$, the first claim is trivial. Otherwise, by construction of the algorithm, we have that $|I_0| = m+1$. Furthermore, by passing from I_{k-1} to I_k, we have that $s_k = round\left(\frac{u_{k-1}+l_{k-1}}{2}\right)$ and

$$|I_k| \leq \begin{cases} \frac{u_{k-1}-l_{k-1}}{2} + \alpha + \frac{1}{2} & \text{if } A(s_k,m) = 1 \text{ and } s_k = \left\lfloor \frac{u_{k-1}-l_{k-1}}{2} \right\rfloor \\ \frac{u_{k-1}-l_{k-1}}{2} + \beta & \text{if } A(s_k,m) = 0 \text{ and } s_k = \left\lfloor \frac{u_{k-1}-l_{k-1}}{2} \right\rfloor \\ \frac{u_{k-1}-l_{k-1}}{2} + \alpha & \text{if } A(s_k,m) = 1 \text{ and } s_k = \left\lceil \frac{u_{k-1}-l_{k-1}}{2} \right\rceil \\ \frac{u_{k-1}-l_{k-1}}{2} + \beta + \frac{1}{2} & \text{if } A(s_k,m) = 0 \text{ and } s_k = \left\lceil \frac{u_{k-1}-l_{k-1}}{2} \right\rceil \end{cases}$$

Thus, by letting $M = \max(\alpha,\beta) + \frac{1}{2}$, we have that $|I_k| \leq \frac{|I_{k-1}|}{2} + M$; this suffices to obtain $|I_{\lceil \log_2 m \rceil}| \leq 2\max(\alpha,\beta) + 2$.

The error probability is the probability that either $\#\phi < \lfloor 2^l \rfloor$ or $\#\phi > 2^u$; this is the probability that one of the $\lceil \log_2 m \rceil$ calls to A has returned a wrong answer. □

In all the experiments we have run, the algorithm has always returned an interval of width at most $2\max(\alpha,\beta)+1$, sometimes in less than $\lceil \log_2(m) \rceil$ iterations: this consideration pragmatically justifies the exit condition we used in the algorithm. We leave for future work a more elaborated analysis of the algorithm to formally establish the $2\max(\alpha,\beta)+1$ bound.

2.2 XOR-based Decision Algorithms

Recall that a XOR constraint c on the variables $y_1, ..., y_m$ is an equality of the form

$$a_0 = a_1 y_1 \oplus \cdots \oplus a_m y_m$$

where $a_i \in \mathbb{F}_2$ for $i = 0, ..., m$ (here $\mathbb{F}_2 = \{0,1\}$ is the two elements field.) Hence c can be identified with a $(m+1)$-tuple in $\mathbb{F}_2^{m+1}$. Assume that a probability distribution on $\mathbb{F}_2^{m+1}$ is fixed. A simple proposal for a decision algorithm $A(s,\phi)$ is as follows:

1. generate s XOR constraints $c_1, \ldots, c_s$ independently, according to the fixed probability distribution;

2. if $\phi \wedge c_1 \wedge \cdots \wedge c_s$ is unsatisfiable then return 0, else return 1.

Indeed [13], every XOR constraint splits the set of boolean assignments in two parts, according to whether the assignment satisfies the constraint or not. Thus, if ϕ has less than 2^s models (and so less than $2^{s+\beta}$), the formula $\phi \wedge c_1 \wedge \cdots \wedge c_s$ is likely to be unsatisfiable.

In step 2 above, any off-the-shelf SAT solver can be employed: one appealing possibility is using CryptoMiniSat [22], which offers support for specifying XOR constraints (see e.g. [3,6,17]). Similarly to [13], it can be proved that this algorithm yields indeed an (α, β, δ)-decision algorithm, for a suitable $\delta < \frac{1}{2}$, if the constraints c_i at step 1 are chosen uniformly at random. This however will generate 'long' constraints, with an average of $\frac{m}{2}$ variables each, which a SAT solver will not be able to manage as m grows.

3 Counting with Sparse XORs

We want to explore alternative ways of generating constraints, which will make it possible to work with short ('sparse') XOR constraints, while keeping the same guarantees of correctness. In what follows, we assume a probability distribution over the constraints, where each coefficient a_i, for $i > 0$, is chosen independently with probability λ, while a_0 is chosen uniformly (and independently from all the other a_i's). In other words, we assume that the probability distribution over $\mathbb{F}_2^{m+1}$ is of the following form, for $\lambda \in \left(0, \frac{1}{2}\right]$:

$$\Pr(a_0, a_1, ..., a_m) \triangleq p(a_0)p'(a_1)\cdots p'(a_m)$$
$$\text{where:} \qquad p(1) = p(0) = \frac{1}{2} \qquad p'(1) = \lambda \qquad p'(0) = 1 - \lambda \qquad (4)$$

The expected number of variables appearing in a constraint chosen according to this distribution will therefore be $m\lambda$. Let us still call A the algorithm presented in Sect. 2.2, with this strategy in choosing the constraints. We want to establish conditions on λ under which A can be proved to be a decision algorithm.

Throughout this section, we fix a boolean formula $\phi(y_1, ..., y_m)$ and $s \geq 1$ XOR constraints. Let

$$\chi_s \triangleq \phi \wedge c_1 \wedge \cdots \wedge c_s$$

where the c_i are chosen independently according to (4). For any assignment (model) σ from variables $y_1, ..., y_m$ to $\mathbb{F}_2$, let us denote by Y_σ the Bernoulli r.v. which is 1 iff σ satisfies χ_s.

We now list the steps needed for proving that A is a decision algorithm. This latter result is obtained by Proposition 1(2) (that derives from Lemma 1(1)) and by combining Proposition 1(1), Lemma 2(3) (that derives from Lemma 1(2)) and Lemma 3 in Theorem 2 later on. In what follows, we shall let $\rho \triangleq 1 - 2\lambda$.

Lemma 1.

1. $\Pr(Y_\sigma = 1) = E[Y_\sigma] = 2^{-s}$.
2. *Let* σ, σ' *be any two assignments and* d *be their Hamming distance in* $\mathbb{F}_2^m$ *(i.e., the size of their symmetric difference seen as subsets of* $\{1, ..., m\}$*). Then* $\Pr(Y_\sigma = 1, Y_{\sigma'} = 1) = E[Y_\sigma \cdot Y_{\sigma'}] = \left(\frac{1+\rho^d}{4}\right)^s$ *(where we let* $\rho^d \triangleq 1$ *whenever* $\rho = d = 0$*.)*

Proof (Sketch). The first claim holds by construction. For the second claim, the crucial thing to prove is that, by fixing just one constraint c_1 (so, $s = 1$), we have that $\Pr(Y_\sigma = 1, Y_{\sigma'} = 1) = \frac{1+\rho^d}{4}$. To this aim, call A and B the sets of variables that are assigned value 1 in σ and σ', respectively; then, we let $U = A \setminus B$, $V = B \setminus A$ and $I = A \cap B$. Let C be the set of variables appearing in the constraint c_1; then, U_e and U_o abbreviate $\Pr(|C \cap U| \text{ is even})$ and $\Pr(|C \cap U| \text{ is odd})$, respectively; similarly for I_e, I_o and V_e, V_o. Then,

$$\Pr(Y_\sigma = 1, Y_{\sigma'} = 1 | a_0 = 0) = U_e I_e V_e + U_o I_o V_o$$
$$\Pr(Y_\sigma = 1, Y_{\sigma'} = 1 | a_0 = 1) = U_o V_o + U_e V_e - U_o I_o V_o - U_e I_e V_e$$

By elementary probability theory, $\Pr(Y_\sigma = 1, Y_{\sigma'} = 1) = \frac{1}{2}(1 - V_e - U_e + 2U_e V_e)$; the result is obtained by noting that $U_e = \frac{1}{2}(1 + \rho^{|U|})$, $V_e = \frac{1}{2}(1 + \rho^{|V|})$ and $d = |U \cup V| = |U| + |V|$. □

Now let T_s be the random variable that counts the number of models of χ_s, when the constraints $c_1, ..., c_s$ are chosen independently according to distribution (4):

$$T_s \triangleq \#\chi_s \,.$$

The event that χ_s is unsatisfiable can be expressed as $T_s = 0$. A first step toward establishing conditions under which A yields a decision algorithm is the following result. It makes it clear that a possible strategy is to keep under control the variance of T_s, which depends in turn on λ. Let us denote by μ_s the expectation of T_s and by $\mathrm{var}(T_s)$ its variance. Note that $\mathrm{var}(T_s) > 0$ if $\#\phi > 0$.

Proposition 1.

1. $\#\phi > 2^{s+\beta}$ *implies* $\Pr(A(s, \phi) = 0) \leq \frac{1}{1 + \frac{\mu_s^2}{\mathrm{var}(T_s)}}$;
2. $\#\phi < 2^{s-\alpha}$ *implies* $\Pr(A(s, \phi) = 1) < 2^{-\alpha}$.

Proof (Sketch). The first claim relies on a version of the Cantelli-Chebyshev inequality for integer nonnegative random variables (a.k.a. Alon-Spencer's inequality); the second claim relies on Lemma 1(1) and Markov's inequality. □

By the previous proposition, assuming $\alpha > 1$, we obtain a decision algorithm (Definition 1) provided that $\mathrm{var}(T_s) < \mu_s^2$. This will depend on the value of λ that is chosen, which leads to the following definition.

Definition 2 (feasibility). *Let ϕ, s and β be given. A value $\lambda \in \left(0, \frac{1}{2}\right]$ is said to be (ϕ, s, β)-*feasible *if $\#\phi > 2^{s+\beta}$ implies $\mathrm{var}(T_s) < \mu_s^2$, where the constraints in χ_s are chosen according to (4).*

Our goal is now to give a method to minimize λ while preserving feasibility. Recall that $T_s \triangleq \#\chi_s$. Denote by $\sigma_1, ..., \sigma_N$ the distinct models of ϕ (hence, $T_s \leq N$). Note that $T_s = \sum_{i=1}^{N} Y_{\sigma_i}$. Given any two models σ_i and σ_j, $1 \leq i, j \leq N$, let d_{ij} denote their Hamming distance. The following lemma gives exact formulae for the expected value and variance of T_s.

Lemma 2. *Let $\rho = 1 - 2\lambda$.*

1. $\mu_s = E[T_s] = N2^{-s}$;
2. $\mathrm{var}(T_s) = \mu_s + 4^{-s} \sum_{i=1}^{N} \sum_{j \neq i} (1 + \rho^{d_{ij}})^s - \mu_s^2$;
3. *If $N \neq 0$, then* $\frac{\mathrm{var}(T_s)}{\mu^2} = \mu_s^{-1} + N^{-2} \sum_{i=1}^{N} \sum_{j \neq i} (1 + \rho^{d_{ij}})^s - 1$

Proof (Sketch). The first two items are a direct consequence of Lemma 1 and linearity of expectation; the third item derives from the previous ones. □

Looking at the third item above, we clearly see that the upper bound on the error probability we are after depends much on 'how sparse' the set of ϕ's models is in the Hamming space $\mathbb{F}_2^m$: the sparser, the greater the distance, the lower the value of the double summation, the better. Let us denote by S the double summation in the third item of the above lemma:

$$S \triangleq \sum_{i=1}^{N} \sum_{j \neq i} (1 + \rho^{d_{ij}})^s$$

In what follows, we will give an upper bound on S which is easy to compute and depends on the minimum Hamming distance d among any two models of ϕ. We need some notation about models of a formula. Below, we let $j = d, ..., m$.

$$l_j \triangleq \begin{cases} j - \left\lceil \frac{d}{2} \right\rceil + 1 & \text{if } j \leq \frac{m}{2} \\ \max\{0, m - j - \left\lceil \frac{d}{2} \right\rceil + 1\} & \text{if } j > \frac{m}{2} \end{cases}$$

$$w^* \triangleq \min \left\{ w : d \leq w \leq m \text{ and } \sum_{j=d}^{w} \binom{m}{l_j} \geq N - 1 \right\} \tag{5}$$

$$N^* \triangleq \sum_{j=d}^{w^*-1} \binom{m}{l_j} \tag{6}$$

where we stipulate that $\min \emptyset = 0$. Note that the definitions of w^* and N^* depend solely on N, m and d.

With the above definitions and results, we have the following upper bound on S.

Lemma 3. *Let the minimal distance between any two models of ϕ be at least d. Then*

$$S \leq N \left(\sum_{j=d}^{w^*-1} \binom{m}{l_j} (1+\rho^j)^s + (N-1-N^*)(1+\rho^{w^*})^s \right)$$

Proof (Sketch). Fix one of the models of ϕ (say σ_i), and consider the sub-summation originated by it, $S_i \stackrel{\triangle}{=} \sum_{j \neq i} \left(\frac{1+\rho^{d_{ij}}}{4}\right)^s$. Let us group the remaining $N-1$ models into disjoint families, $\mathcal{F}_d, \mathcal{F}_{d+1}, \ldots$, of models that are at distance $d, d+1, ...,$ respectively, from σ_i. Note that each of the $N-1$ models gives rise to exactly one term in the summation S_i. Hence, $S_i = \sum_{j=d}^{m} |\mathcal{F}_j| \left(\frac{1+\rho^j}{4}\right)^s$. By the Ray-Chaudhuri-Wilson Lemma [2, Th.4.2], $|\mathcal{F}_j| \leq \binom{m}{l_j}$. Hence, upper-bounding S_i consists, e.g., in choosing a tuple of integers $x_d, \ldots, x_m$ such that $\sum_{j=d}^{m} x_j \left(\frac{1+\rho^j}{4}\right)^s \geq \sum_{j=d}^{m} |\mathcal{F}_j| \left(\frac{1+\rho^j}{4}\right)^s$, under the constraints $0 \leq x_j \leq \binom{m}{l_j}$, for $j = d, \ldots, m$, and $\sum_{j=d}^{m} x_j = N-1$. An optimal solution is

$$x_j = \begin{cases} \binom{m}{l_j} & \text{for } j = d, \ldots, w^*-1 \\ N-1-N^* & \text{for } j = w^* \\ 0 & \text{for } j > w^* \end{cases}$$

The thesis is obtained by summing over all models σ_i. □

Definition 3. *Given $s \geq 1$, $\beta > 0$, $d \geq 1$ and $\lambda \in \left(0, \frac{1}{2}\right]$, let us define*

$$B(s,m,\beta,d,\lambda) \stackrel{\triangle}{=} 2^{-\beta} + 2^{-s-\beta}\left(\sum_{j=d}^{w^*-1}\binom{m}{l_j}(1+\rho^j)^s + (N-1-N^*)(1+\rho^{w^*})^s\right) - 1,$$

where $\rho \stackrel{\triangle}{=} 1-2\lambda$ and $N = \lceil 2^{s+\beta} \rceil$ also in the definition of w^* and N^*.

Using the facts collected so far, the following theorem follows, giving an upper bound on $\frac{\mathrm{var}(T_s)}{\mu_s^2}$.

Theorem 2 (upper bound). *Let the minimal distance between models of ϕ be at least d and $\#\phi > 2^{s+\beta}$. Then, $\frac{\mathrm{var}(T_s)}{\mu_s^2} \leq B(s,m,\beta,d,\lambda)$.*

Proof. First note that we can assume without loss of generality that $\#\phi = N = \lceil 2^{s+\beta} \rceil$. If this was not the case, we can consider in what follows any formula ϕ' whose models are models of ϕ but are exactly $\lceil 2^{s+\beta} \rceil$ (ϕ' can be obtained by adding some conjuncts to ϕ that exclude $\#\phi - \lceil 2^{s+\beta} \rceil$ models). Then, $\Pr(A(s,\phi)=0) \leq \Pr(A(s,\phi')=0)$ and this would suffice, for the purpose of upper-bounding $\Pr(A(s,\phi)=0)$. The result follows from Proposition 1(1), Lemma 2(3), Lemma 3 and by the fact that $N = \lceil 2^{s+\beta} \rceil \geq 2^{s+\beta}$. □

The following notation will be useful in the rest of the paper. For $0 \leq \gamma \leq 1$, define

$$\lambda_\gamma^*(s,m,\beta,d) \stackrel{\triangle}{=} \inf\left\{\lambda \in \left(0, \frac{1}{2}\right] \,:\, B(s,m,\beta,d,\lambda) \leq \gamma\right\} \tag{7}$$

where we stipulate $\inf \emptyset = +\infty$.

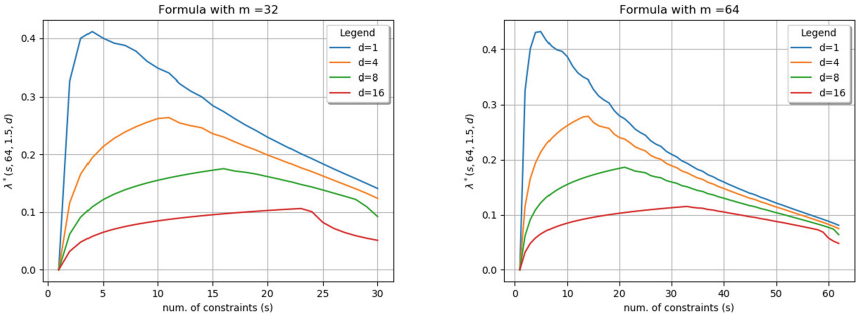

Fig. 1. Plots of λ_1^* as a function of s, for $m = 32$ and $m = 64$, $\beta = 1.5$, and different values of d. For any value of s and d, any value of λ above the curve is feasible.

Corollary 1 (Feasibility). *Assume the minimal distance between any two models of ϕ is at least d. Then every $\lambda \in \left(\lambda_1^*(s, m, \beta, d), \frac{1}{2}\right]$ is (ϕ, s, β)-feasible.*

4 Evaluation

4.1 Theoretical Bounds on Expected Constraint Length

To assess the improvements that our theory introduces on XOR-based approximate counting, we start by considering the impact of minimum Hamming distance on the expected length of the XOR constraints. First, in Fig. 1 we plot λ_1^* as a function of s, for fixed values of $m = 32$ and 64, $\beta = 1.5$, and four different values of d. Note that the difference between different values of d tends to vanish as s gets large – i.e. close to m.

Next, we compare our theoretical bounds with those in [9], where a goal similar to ours is pursued. Interestingly, their bounds coincide with ours when setting $d = 1$ – no assumption on the minimum Hamming distance – showing that our approach generalizes theirs. We report a numerical comparison in Table 1, where several hypothetical values of m (no. of variables) and s (no. of constraints) are considered. Following a similar evaluation conducted in [9, Tab. 1], here we fix the error probability to $\delta = \frac{4}{9}$ and the upper slack parameter to $\beta = 2$, and report the values of $\lambda \times m$ for the minimal value of λ that would guarantee a confidence of at least $1 - \delta$ in case an upper bound is found, computed with their approach and ours. Specifically, in their case λ is obtained via the formulae in [9, Cor.1,Th.3], while in our case $\lambda = \lambda_\gamma^*$ for $\gamma = 0.8$, which entails the wanted confidence according to Proposition 1(2). We see that, under the assumption that lower bounds on d as illustrated are known, in some cases a dramatic reduction of the expected length of the XOR constraints is obtained.

4.2 Execution Times

Although the focus of the present paper is mostly theoretical, it is instructive to look at the results of some simple experiments for a first concrete assessment

Table 1. Comparison with the provable bounds from [9, Tab. 1].

N. Vars (m)	N. Constraints (s)	$\lambda \times m$ from [9]	$\lambda \times m$ present paper		
			$d = 1$	$d = 5$	$d = 20$
50	13	16.85	16.85	11.76	3.88
50	16	15.38	15.38	11.36	4.1
50	20	13.26	13.26	10.26	4.37
50	30	9.57	9.57	8.02	4.75
50	39	7.08	7.08	6.2	4.45
100	11	39.05	39.05	23.65	7.4
100	15	35.44	35.44	25.26	8.07
100	25	27.09	27.09	21.33	9.14
119	7	50.19	50.18	25.07	7.6
136	9	55.63	55.63	30.66	9.46
149	11	60.6	60.6	35.24	11.02
352	10	147.99	147.99	81.42	25.31

of the proposed methodology. To this aim, we have implemented in Python the algorithm C_A with A as described in Sect. 3, relying on CryptoMiniSAT [22] as a SAT solver, and conducted a few experiments[1].

The crucial issue to use Theorem 2 is the knowledge of (a lower bound on) the minimal distance d among the models of the formula we are inputting to our algorithm. In general, this information is unknown and we shall discuss a possible approach to face this problem in the next section. For the moment, we use a few formulae describing the set of codewords of certain error correcting codes, for which the number of models and the minimum distance is known by construction. Each of such formulae derives from a specific BCH code [5,15] (these are very well-known error-correcting codes whose minimal distance among the codewords is lower-bounded by construction). In particular, `Fxx-yy-zz.cnf` is a formula in CNF describing membership to the BCH code for $2^{\texttt{xx}}$ messages (the number of models), codified via codewords of `yy` bits and with a distance that is at least `zz`.

For these formulae, we run 3 kinds of experiments. First, we run the tool for every formula without using the improvements of the bounds given by knowing the minimum distance (i.e., we used Theorem 2 by setting $d = 1$). Second, we run the tool for every formula by using the known minimum distance. Third, we run the state-of-the-art tool for approximate model counting, called ApproxMC3 [21] (an improved version of ApproxMC2 [7]). The results obtained are reported in Table 2.

[1] Run on a MacBook Air, with a 1,7 GHz Intel Core i7 processor, 8 GB of memory (1600 MHz DDR3) and OS X 10.9.5.

Table 2. Results for our tool with $\alpha = \beta = 1.5$, $d = 1$ and $d = d_{min}$, compared to ApproxMC3 with a tolerance $\epsilon = 3$. In all trials, the error probability δ is 0.1.

Formula	*Our tool with* $d = 1$	*Our tool with* $d = d_{min}$	*ApproxMC3*
F21-31-5.cnf	res: $[2^{19.5}, 2^{23.5}]$ time: 8.05 s	res: $[2^{19.5}, 2^{23.5}]$ time: 6.73 s	res: ?? time: > 3 h
F16-31-7.cnf	res: $[2^{14.5}, 2^{18.5}]$ time: 11.75 s	res: $[2^{14.5}, 2^{18.5}]$ time: 9.5 s	res: $[2^{13.86}, 2^{17.85}]$ time: 22 min 43 s
F11-31-9.cnf	res: $[2^{9.5}, 2^{13.5}]$ time: 6.75 s	res: $[2^{9.5}, 2^{13.5}]$ time: 4.32 s	res: $[2^{9.09}, 2^{13.09}]$ time: 15.24 s
F6-31-11.cnf	res: $[2^{4.5}, 2^{8.5}]$ time: 3.01 s	res: $[2^{4.5}, 2^{8.5}]$ time: 2.62 s	res: $[2^{4}, 2^{8}]$ time: 1.9 s
F16-63-23.cnf	res: $[2^{14.5}, 2^{18.5}]$ time: 31 min 15 s	res: $[2^{14.5}, 2^{18.5}]$ time: 2 min 36 s	res: $[2^{13.9}, 2^{17.9}]$ time: 54 min 57 s

To compare our results with theirs, we have to consider that, if our tool returns $[l, u]$, then the number of models lies in $[\lfloor 2^l \rfloor, \lceil 2^u \rceil]$ with error probability δ (set to 0.1. in all experiments). By contrast, if ApproxMC3 returns a value M, then the number of models lies in $\left[\frac{M}{1+\epsilon}, M(1+\epsilon)\right]$ with error probability δ (again, set to 0.1 in all experiments). So, we have to choose for ApproxMC3 a tolerance ϵ that produces an interval of possible solutions comparable to what we obtain with our tool. The ratio between the sup and the inf of our intervals is $2^{2\max(\alpha,\beta)+1}$ (indeed, A always returned an interval such that $u - l \leq 2\max(\alpha, \beta) + 1$); when $\alpha = \beta = 1.5$, the value is 16. By contrast, the ratio between the sup and the inf of ApproxMC3's intervals is $(1+\epsilon)^2$; this value is 16 for $\epsilon = 3$.

In all formulae that have "sufficiently many" models – empirically, at least 2^{11} – our approach outperforms ApproxMC3. Moreover, making use of the minimum distance information implies a gain in performance. Of course, the larger the distance, the greater the gain – again, provided that the formula has sufficiently many models: compare, e.g., the first and the last formula.

4.3 Towards a Practical Methodology

To be used in practice, our technique requires a lower bound on the minimum Hamming distance between any two models of ϕ. We discuss below how error-correcting codes might be used, in principle, to obtain such a bound. Generally, speaking an error-correcting code adds redundancy to a string of bits and inflates the minimum distance between the resulting codewords. The idea here is to transform the given formula ϕ into a new formula ϕ' that describes an encoding of the original formula's models: as a result $\#\phi' = \#\phi$, but the models of ϕ' live in a higher dimensional space, where a minimum distance between models is ensured.

Assume that $\phi(y)$ is already in Conjunctive Normal Form. Fix a binary linear $[n, m, d]$ block code $\mathcal{C}$ (i.e. a code with 2^m codewords of n bits and minimum distance d), and let G be its generator matrix, i.e. a $m \times n$ binary matrix such that the codeword associated to $u \in \mathbb{F}_2^m$ is uG (where we use the vector-matrix multiplication in the field $\mathbb{F}_2$). The fact that a $c \in \mathbb{F}_2^n$ is a codeword can be expressed by finding some u that satisfies the conjunction of the n XOR constraints:

$$c_i = \bigoplus_{j=1}^{m} u_j \cdot G_{ij} \quad \text{for } i = 1, \ldots, n\,.$$

Again, the important condition here is that G be *sparse* (on its columns), so that the above formula effectively corresponds to a conjunction of sparse XOR constraints. That is, we should confine ourselves to low-density parity check (LDPC) codes [11,18]. Now, we consider the formula

$$\phi'(z) \stackrel{\triangle}{=} \exists y(\phi(y) \wedge z = yG)\,.$$

It can be easily proved that ϕ and ϕ' have the same number of models. If we now assume a minimum distance of d when applying Theorem 2, we have a decrease in the feasibility threshold λ^*, as prescribed by (7). This gain must of course be balanced against the increased number of boolean variables in the formula (viz. n). We will have an actual advantage using $\mathcal{C}$ over not using it (and simply assuming $d = 1$) if and only if, by using $\mathcal{C}$, the expected length of the resulting XOR constraints is actually smaller. By letting $\lambda^{*,d} \stackrel{\triangle}{=} \lambda_1^*(s, n, \beta, d)$, the latter fact holds if and only if

$$n\lambda^{*,d} \leq m\lambda^{*,1} \tag{8}$$

or equivalently $\lambda^{*,d} \leq R\lambda^{*,1}$, where $R \stackrel{\triangle}{=} \frac{m}{n}$ is the *rate* of $\mathcal{C}$. This points to codes with high rate and big minimum distance. Despite these two parameters pull one against the other, (8) can be fulfilled, and good expected length bounds obtained, by choosing $\mathcal{C}$ appropriately.

For example, [10] presents a $[155, 64, 20]$-LDPC code, that is a code with block length of 155, with 2^{64} codewords and with minimum distance between codewords of 20. In Fig. 2 we compare the expected length of the resulting XOR constraints in the two cases – $m \times \lambda^*(s, m, \beta, 1)$ (without the code, for $m = 32$ and $m = 64$) and $155 \times \lambda_1^*(s, 155, \beta, 20)$ (with the code) – as functions of s, for fixed $\beta = 1.5$. As seen from the plots, the use of the code offers a significant advantage in terms of expected length up to $s = 10$ and $s = 26$, respectively.

We have performed a few tests for a preliminary practical assessment of this idea. Unfortunately, in all but a few cases, the use of the said $[155, 64, 20]$-LDPC code does not offer a significant advantage in terms of execution time. Indeed, embedding a formula in this code implies adding 155 new XOR constraints: the presence of so many constraints, however short, apparently outweights the benefit of a minimum distance $d = 20$. We hope that alternative codes, with a more advantageous block length versus minimum distance tradeoff, would fare

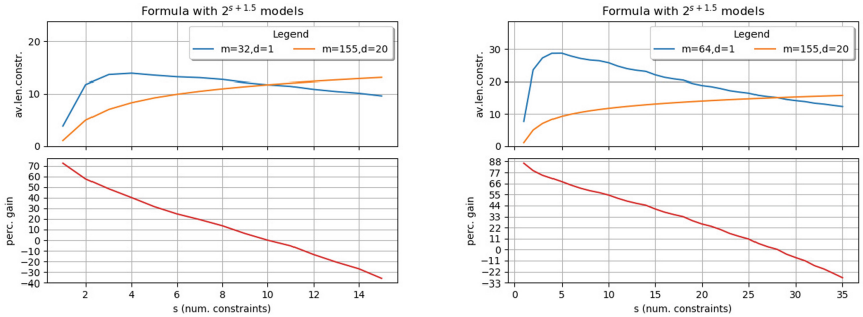

Fig. 2. Plots of the expected length of XOR constraints as a function of s, with and without code, and the relative percentual gain. Here, $m = 32$ (left) and 64 (right), $\beta = 1.5$ and the code is a $[155, 64, 20]$-LDPC.

better. Indeed, as we showed in Table 1, relatively small distances (e.g. $d = 5$) can already give interesting gains, if the number of extra constraints is small. We leave that as subject for future research.

5 Conclusion

We have studied the relation between sparse XOR constraints and minimum Hamming distance in model counting. Our findings suggest that minimum distance plays an important role in making the feasibility threshold for λ (density) lower, thus potentially improving the effectiveness of XOR based model counting procedures. These results also prompt a natural direction for future research: embedding the set of models into a higher dimensional Hamming space, so as to enforce a given minimum distance.

Beside the already mentioned [9], our work also relates to the recent [1]. There, constraints are represented as systems $Ay = b$, for A a random LDPC matrix enjoying certain properties, b a random vector, and y the variable vector. Their results are quite different from ours, but also they take the geometry of the set of models into account, including minimum distance. In particular, they make their bounds depend also on a "boost" parameter which appears quite difficult to compute. This leads to a methodology that is only empirically validated – that is, the model count results are offered with no guarantee.

Acknowledgements. We thank Marco Baldi, Massimo Battaglioni and Franco Chiaraluce for providing us with the generator and parity check matrices of the LDPC code in Subsect. 4.3.

References

1. Achlioptas, D., Theodoropoulos, P.: Probabilistic model counting with short XORs. In: Gaspers, S., Walsh, T. (eds.) SAT 2017. LNCS, vol. 10491, pp. 3–19. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66263-3_1

2. Babai, L., Frankl, P.: Linear Algebra Methods in Combinatorics. The University of Chicago, Chicago (1992)
3. Biondi, F., Enescu, M.A., Heuser, A., Legay, A., Meel, K.S., Quilbeuf, J.: Scalable approximation of quantitative information flow in programs. Verification, Model Checking, and Abstract Interpretation. LNCS, vol. 10747, pp. 71–93. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73721-8_4
4. Boreale, M., Gorla, D.: Approximate model counting, sparse XOR constraints and minimum distance. https://arxiv.org/abs/1907.05121 (2019)
5. Bose, R.C., Ray-Chaudhuri, D.K.: On a class of error correcting binary group codes. Inf. Control **3**(1), 68–79 (1960)
6. Chakraborty, S., Meel, K.S., Vardi, M.Y.: A scalable approximate model counter. In: Schulte, C. (ed.) CP 2013. LNCS, vol. 8124, pp. 200–216. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40627-0_18
7. Chakraborty, S., Meel, K.S., Vardi, M.Y.: Algorithmic improvements in approximate counting for probabilistic inference: from linear to logarithmic SAT calls. In: Proceedings of International Joint Conference on Artificial Intelligence (2016)
8. Chatzikokolakis, K., Palamidessi, C., Panangaden, P.: Anonymity protocols as noisy channels. Inf. Comput. **206**(2–4), 378–401 (2008)
9. Ermon, S., Gomes, C.P., Sabharwal, A., Selman, B.: Low-density parity constraints for hashing-based discrete integration. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014, pp. 271–279 (2014)
10. Fuja, T.E., Sridhara, D., Tanner, R.M.: A class of group-structured LDPC codes. In: International Symposium on Communication Theory and Applications (2001)
11. Gallager, R.G.: Low Density Parity Check Codes. MIT Press, Cambridge (1963)
12. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: *clasp*: a conflict-driven answer set solver. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483, pp. 260–265. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72200-7_23
13. Gomes, C.P., Sabharwal, A., Selman, B.: Model counting: a new strategy for obtaining good bounds. In: Proceedings of AAAI, pp. 54–61 (2006)
14. Gomes, C.P., Sabharwal, A., Selman, B.: Model counting. In: Handbook of Satisfiability, pp. 633–654. IOS Press (2009)
15. Hocquenghem, A.: Codes correcteurs d'erreurs. Chiffres **2**, 147–156 (1959)
16. Klebanov, V., Manthey, N., Muise, C.: SAT-based analysis and quantification of information flow in programs. In: Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P.R. (eds.) QEST 2013. LNCS, vol. 8054, pp. 177–192. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40196-1_16
17. Klebanov, V., Weigl, A., Weibarth, J.: Sound probabilistic #SAT with projection. In: Proceedings of QAPL (2016)
18. MacKay, D.J.C.: Good error-correcting codes based on very sparse matrices. IEEE Trans. Inf. Theory **45**(3), 399–432 (1999)
19. Muise, C., McIlraith, S.A., Beck, J.C., Hsu, E.I.: Dsharp: fast d-DNNF compilation with sharpSAT. In: Kosseim, L., Inkpen, D. (eds.) AI 2012. LNCS (LNAI), vol. 7310, pp. 356–361. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30353-1_36
20. Smith, G.: On the foundations of quantitative information flow. In: de Alfaro, L. (ed.) FoSSaCS 2009. LNCS, vol. 5504, pp. 288–302. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00596-1_21

21. Soos, M., Meel, K.S.: BIRD: engineering an efficient CNF-XOR SAT solver and its applications to approximate model counting. In: Proceedings of AAAI Conference on Artificial Intelligence (AAAI) (2019)
22. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 244–257. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02777-2_24. http://www.msoos.org/cryptominisat2/
23. Thurley, M.: sharpSAT – counting models with advanced component caching and implicit BCP. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 424–429. Springer, Heidelberg (2006). https://doi.org/10.1007/11814948_38
24. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comput. **8**(3), 410–421 (1979)
25. Valiant, L.G., Vazirani, V.V.: NP is as easy as detecting unique solutions. Theor. Comput. Sci. **47**(3), 85–93 (1986)