Model Counting: A New Strategy for Obtaining Good Bounds

Carla P. Gomes and **Ashish Sabharwal** and **Bart Selman**
Department of Computer Science
Cornell University, Ithaca NY 14853-7501, USA
{gomes,sabhar,selman}@cs.cornell.edu [*]

Abstract

Model counting is the classical problem of computing the number of solutions of a given propositional formula. It vastly generalizes the NP-complete problem of propositional satisfiability, and hence is both highly useful and extremely expensive to solve in practice. We present a new approach to model counting that is based on adding a carefully chosen number of so-called streamlining constraints to the input formula in order to cut down the size of its solution space in a controlled manner. Each of the additional constraints is a randomly chosen XOR or parity constraint on the problem variables, represented either directly or in the standard CNF form. Inspired by a related yet quite different theoretical study of the properties of XOR constraints, we provide a formal proof that with high probability, the number of XOR constraints added in order to bring the formula to the boundary of being unsatisfiable determines with high precision its model count. Experimentally, we demonstrate that this approach can be used to obtain good bounds on the model counts for formulas that are far beyond the reach of exact counting methods. In fact, we obtain the first non-trivial solution counts for very hard, highly structured combinatorial problem instances. Note that unlike other counting techniques, such as Markov Chain Monte Carlo methods, we are able to provide high-confidence guarantees on the quality of the counts obtained.

Introduction

Propositional model counting is the problem of computing the number of models for a given propositional formula, *i.e.*, the number of distinct variable assignments for which the formula evaluates to TRUE. This problem generalizes the well-known NP-complete problem of propositional satisfiability, SAT, which has played a key role in complexity theory as well as in automated reasoning. Indeed, computing the exact model count is a #P-complete problem, which means that it is no easier than solving a propositional formula with an unbounded number of "there exist" and "forall" quantifiers in its variables (Toda 1989). For comparison, recall that SAT can be thought of as a propositional formula with exactly one level of "there exist" quantification.

[*]Research supported by Intelligent Information Systems Institute (IISI), Cornell University (AFOSR grant F49620-01-1-0076) and DARPA (REAL grant FA8750-04-2-0216).

Effective model counting procedures would open up a range of new applications. For example, various probabilistic inference problems, such as Bayesian net reasoning, can be effectively translated into model counting problems (cf. Roth; Littman, Majercik, & Pitassi; Darwiche 1996; 2001; 2005). Another application is in the study of hard combinatorial problems, such as combinatorial designs, where the number of solutions provides further insights into the problem. Even finding a single solution can be a challenge for such problems: counting the number of solutions is much harder yet. Using our counting method, we will obtain the first non-trivial lower bounds on the number of solutions of several complex combinatorial problems.

The earliest practical approach for counting models is based on an extension of systematic DPLL-based SAT solvers. By using appropriate multiplication factors and continuing the search after a single solution is found, `Relsat` (Bayardo Jr. & Pehoushek 2000) is able to provide incremental lower bounds on the model count as it proceeds, and finally computes the exact model count. Newer tools such as `Cachet` (Sang *et al.* 2004) often improve upon this by using techniques such as component caching.

All exact counting methods, including `Relsat` and `Cachet`, essentially attack a #P-complete problem "head on" — by searching the raw combinatorial search space. Consequently, these algorithms often have difficulty scaling up to larger problem sizes. We should point out that problems with a higher solution count are not necessarily harder to determine the model count of. In fact, `Relsat` can compute the exact model count of highly under-constrained problems with many "don't care" variables and a lot of models by exploiting big clusters in the solution space. The model counting problem is instead much harder for more intricate combinatorial problems where the solutions are spread much more finely throughout the combinatorial space. We consider examples of such problems in our experiments.

A relatively new approach introduced by Wei & Selman (2005) is to use Markov Chain Monte Carlo sampling to compute an approximation of the exact model count. Their tool, `ApproxCount`, is able to solve several instances quite accurately, while scaling much better than both `Relsat` and `Cachet` as problem size increases. The drawback of `ApproxCount` is that one is not able to provide any hard guarantees on the model count it computes. To output a number close to the exact count, the counting strategy of Wei & Selman requires *uniform sampling from the set of solutions*, which is generally very difficult to achieve. Uniform sampling from the solution space is much harder than just generating a single solution. MCMC methods can provide theoretical convergence guarantees but only in the limit, generally after an exponential number of steps.

Interestingly, the inherent strength of most state-of-the-art SAT solvers comes actually from the ability to quickly narrow down to a certain portion of the search space the solver is designed to handle best. Such solvers therefore sample solutions in a highly non-uniform manner, making them seemingly ill-suited for model counting, unless one forces the solver to explore the full combinatorial space. An interesting question is whether there is a way around this apparent limitation of the use of state-of-the-art SAT solvers for model counting. Our key contribution is *a new method for model counting, which uses a state-of-the-art SAT solver "as is"*. It follows immediately that the more efficient the SAT solver used, the more powerful our counting strategy becomes.

Our approach is inspired by recent work on so-called "streamlining constraints" (Gomes & Sellmann 2004), in which additional, non-redundant constraints are added to the original problem to increase constraint propagation and to focus the search on a small part of the subspace, (hopefully) still containing solutions. This strategy was shown to be successful in solving very hard combinatorial design problems, with carefully created, domain-specific streamlining constraints. In contrast, in this work, we introduce a domain-independent streamlining technique.

Streamlining could potentially also be used to obtain an accurate estimate of the total solution count, if the solution density in the remaining (streamlined) search space was similar to that of the overall solution density. In that case, one could count the solutions remaining in the subspace and multiply this by the relative size of the subspace to the overall search space. Interestingly, there exist *generic* constraints that can be used to probabilistically streamline the search space sufficiently uniformly. These are so-called parity or XOR constraints, represented by logical XOR formulas.

The central idea of our approach is to repeatedly add randomly chosen XOR or parity constraints on the problem variables to the input formula and feed the result to a state-of-the-art SAT solver. We will discuss the technical details below, but at a very high level, our approach works as follows. Each random XOR constraint will cut the search space approximately in half. So, intuitively, if after the addition of s XOR's the formula is still satisfiable, the original formula must have at least on the order of 2^s models. More rigorously, we will show that if we perform t experiments of adding s random XOR constraints and our formula remains satisfiable in each case, then with probability at least $1-2^{-\alpha t}$, our original formula will have at least $2^{s-\alpha}$ satisfying assignments for any $\alpha \geq 1$. So, by repeated experiments or by weakening the claimed bound, one can arbitrarily boost the confidence in the lower bound count. We also give results for the upper bound, and formalize two variants of this approach as algorithms `MBound` and `Hybrid-MBound`.

Of course, the above argument might raise suspicion, because it does not depend at all on the how the solutions are distributed throughout the search space. This however is the surprising feature of the approach. We rely on the very special properties of random parity constraints, which in effect provide a good hash function, randomly dividing the solutions into two near-equal sets. Such constraints were first used by Valiant & Vazirani (1986) in a randomized reduction from SAT to so-called unique SAT. They provided evidence that unique SAT problems (formulas with at most one satisfying assignment) are essentially as hard as general SAT problems. In this work, we show a different, more positive, use of XOR constraints, allowing us to count assignments of hard combinatorial problems.

In the theoretical section of the paper, we give much more specific details on the bounds obtained from XOR-streamlining. To demonstrate that this strategy is not just of theoretical interest, we also provide experimental results. Specifically, we applied our technique to three hard combinatorial problems, the Ramsey problem, the Schur problem, and the clique coloring problem. Our technique provides the first good lower bounds on solution counts for these problems. For both problems, we compute lower bounds with 99% confidence. For the Ramsey problems, we obtained a lower bound of $2^{64} \approx 1.8 \times 10^{19}$ solutions, in under two hours of computation. By comparison, `Relsat` found only 194,127 models in over 12 hours (`Cachet` does not provide partial counts and timed out, and `ApproxCount` does not converge to a solution). For the Schur problem, we obtained a lower bound of $2^{26} \approx 6.7 \times 10^7$ solutions, in under 5 hours of computation. `Relsat`, `Cachet`, and `ApproxCount` could not find any solutions in over 12 hours. For the clique coloring problem, we found over 10^{40} solutions in only a few minutes while other methods didn't finish in 12 hours.

In summary, we provide a new approach to model counting. Our method is unique in that it can use any state-of-the-art SAT solver without any modifications. Our approach uses randomized streamlining XOR constraints and gives concrete bounds with high probability (desired confidence level is under control of the user). Our experiments provide the first non-trivial lower bounds on solution counts for three highly combinatorial problems. In each case these bounds dramatically improve upon existing methods.

Preliminaries

For the rest of this paper, fix the set of propositional variables in all formulas to be V, $|V| = n$. A *variable assignment* $\sigma : V \rightarrow \{0,1\}$ is a function that assigns a value in $\{0,1\}$ to each variable in V. We may think of the value 0 as FALSE and the value 1 as TRUE. We will often abuse notation and write $\sigma(i)$ for valuations of entities $i \notin V$ when the intended meaning is either already defined or is clear from the context. In particular, $\sigma(1) = 1$ and $\sigma(0) = 0$. When $\sigma(i) = 1$, we say that σ *satisfies* i. For $x \in V$, $\neg x$ denotes the corresponding *negated* variable; $\sigma(\neg x) = 1 - \sigma(x)$.

Let F be a formula over the set V of variables and let σ be a variable assignment. $\sigma(F)$ denotes the valuation of F under σ. If σ satisfies F, i.e., $\sigma(F) = 1$, then σ is a *model*, *solution*, or *satisfying assignment* for F. The *model* *count* of F, denoted $MC(F)$, is the number of models of F. The *(propositional) model counting problem* is to compute $MC(F)$ given a (propositional) formula F.

Although our theoretical results hold for propositional formulas in general, we present our work in the context of formulas in the standard conjunctive normal form or CNF, on which our experimental results rely. A *clause* (also called a CNF *constraint*) C is a logical disjunction of a set of possibly negated variables; σ satisfies C if it satisfies at least one signed variable of C. A formula F is in the CNF *form* if it is a logical conjunction of a set of clauses; σ satisfies F if it satisfies all clauses of F.

An XOR *constraint* D over variables V is the logical "xor" or parity of a subset of $V \cup \{1\}$; σ satisfies D if it satisfies an *odd number* of elements in D. The value 1 allows us to express even parity. For instance, $D = \{a,b,c,1\}$ represents the xor constraint $a \oplus b \oplus c \oplus 1$, which is TRUE when an even number of a,b,c are TRUE. Note that it suffices to use only positive variables. E.g., $\neg a \oplus b \oplus \neg c$ and $\neg a \oplus b$ are equivalent to $D = \{a,b,c\}$ and $D = \{a,b,1\}$, respectively.

Our focus will be on formulas in the CNFXOR *form*, i.e., a logical conjunction of clauses and XOR constraints. Note that for every two complementary XOR constraints involving the same subset of V (e.g., $c \oplus d$ and $c \oplus d \oplus 1$), any assignment σ satisfies exactly one of them. This simple property will be crucial for our analysis.

Let $\mathbb{X}$ denote the set of all XOR constraints over V. For $1 \le k \le n = |V|$, let $\mathbb{X}^k$ denote the subset of $\mathbb{X}$ containing only those constraints that involve exactly k variables (and possibly the element 1). For simplicity, we will assume in the rest of the paper that n is even, and will be interested only in $1 \le k \le n/2$. This assumption can be avoided, for instance, by adding a dummy variable with a fixed TRUE/FALSE value or by defining $\mathbb{X}^{n/2} = \mathbb{X}^{\lfloor n/2 \rfloor} \cup \mathbb{X}^{\lceil n/2 \rceil}$ when n is odd.

A Simple Model Counting Algorithm

Our main algorithm is MBound, described below as Algorithm 1. It provides bounds on the model count of an input formula F by adding s random XOR constraints to F, solving the resulting CNFXOR formula using an arbitrary SAT solver as a subroutine, repeating this process t times, and looking at the observed satisfiable vs. unsatisfiable (sat-unsat) distribution of the t CNFXOR formulas. If this observed distribution is biased away from half-and-half, a bound on $MC(F)$ is reported. Specifically, for a slack factor α, if most instances are satisfiable, $2^{s-\alpha}$ is reported as a lower bound, and if most instances are unsatisfiable, $2^{s+\alpha}$ is reported as an upper bound. The correctness of these bounds depends on various factors and is quantified in the next section.

Parameters: MBound has five parameters: (1) the *size k* of the XORs used, (2) the *number s* of the XORs used, (3) the number t of repetitions or *trials*, (4) the *deviation* $\delta \in (0, 1/2]$ from the 50-50 sat-unsat ratio, and (5) the *precision slack* $\alpha \ge 1$. s and t will be the most crucial parameters, and we will often use $k \ll n/2$ for our experiments.

Output: MBound has three modes of termination: (1) return a lower bound on $MC(F)$, (2) return an upper bound on

Algorithm 1: MBound

```
Params: $k,s,t,\delta,\alpha$ : $k,s,t$ positive integers,
        $k \le n/2,\ 0 < \delta \le 1/2,\ \alpha \ge 1$
Input   : A CNF formula $F$
Output  : A lower or upper bound on $MC(F)$, or Failure
begin
    $numSat \leftarrow 0$
    for $i \leftarrow 1$ to $t$ do
        $Q_s \leftarrow \{s$ random constraints from $\mathbb{X}^k\}$
        $F_s^k \leftarrow F \cup Q_s$
        $result \leftarrow$ SATSolve($F_s^k$)
        if $result =$ TRUE then $numSat \leftarrow numSat + 1$
    if $numSat \ge t \cdot (1/2 + \delta)$ then
        return Lower bound: $MC(F) > 2^{s-\alpha}$
    else if $numSat \le t \cdot (1/2 - \delta)$ then
        return Upper bound: $MC(F) < 2^{s+\alpha}$
    else return Failure
end
```

$MC(F)$, or (3) return "Failure" without reporting any bound whatsoever. A Failure happens when the observed sat-unsat ratio is less than δ away from 50-50.

We say that MBound *makes an error* if it reports an incorrect lower or upper bound on $MC(F)$, and that it *fails* if it reports Failure. We expect the probability of MBound making an error to go down as k,t,δ, and α increase.

MBound is based on the following central idea. As observed by Valiant & Vazirani (1986), the effect of adding a random XOR from $\mathbb{X}$ to F is to cut down the number of models by approximately a half. The same holds also when using $\mathbb{X}^{n/2}$ instead of $\mathbb{X}$. Somewhat surprisingly, this works *no matter how* solutions are structured in the space of all variable assignments. This is because constraints in $\mathbb{X}$ and $\mathbb{X}^{n/2}$ act as pairwise-independent uniform hash functions — uniformity allowing them to accept each model with probability exactly a half, and pairwise-independence making them oblivious to their acceptance or rejection of another model.

As the solution space is being iteratively cut down into halves, the number of XOR constraints one expects to add to F to bring it to the boundary of being unsatisfiable is roughly $s^* = \log_2 MC(F)$. This is the key property MBound uses to approximate $MC(F)$. Of course, this is only the expected behavior. To make the algorithm robust, we give a detailed probabilistic analysis to show that MBound is unlikely to deviate significantly from its expected behavior. This analysis forms the core of the technical contribution of this paper on the theoretical side and extends to provable guarantees for our second algorithm, Hybrid-MBound, as well.

In practice, adding XOR constraints from $\mathbb{X}^{n/2}$ (i.e., *large* XORs) can make the underlying SAT solver quite inefficient, and one is forced to consider the spaces $\mathbb{X}^k, k \ll n/2$, of *small* XORs. Such small XORs, however, do not necessarily act pairwise-independently on the solution space, resulting in an algorithm of not as high a quality as with large XORs. Interestingly, small XORs turn out to be sufficient to obtain guaranteed *lower* bounds. Moreover, as k increases, one provably approaches the truly pairwise-independent random behavior of large XORs. In fact, in several domains, fairly small XORs work quite well in practice. Further, our preliminary results suggest that this fact can be used to our advantage in order to provide key insights into the clustering structure of the solution space. This, however, is beyond the scope of this paper.

Analysis of Algorithm MBound

For $1 \leq k \leq n/2$, let Q_s^k denote a set of s XOR constraints chosen independently and uniformly at random from $\mathbb{X}^k$, and let F_s^k denote the random CNFXOR formula obtained by adding the constraints Q_s^k to F. In our analysis, the probability will be over the random choice of Q_s^k.

Our technical arguments have the following flavor. We show that when one adds too many XORs, even small ones, the resulting CNFXOR formula is quite unlikely to remain satisfiable. Further, if one conducts a *meta*-experiment by repeating this experiment with a fixed number of (too many) XORs several times, one is extremely unlikely to see many satisfiable formulas. By focusing on a meta-experiment that produces enough satisfiable formulas, one is thus able to probabilistically conclude that the number of XORs added was indeed *not* too many, providing a lower bound on the model count. A similar reasoning with too few XORs provides an upper bound, though with some complications arising from the lack of pairwise-independence of small XORs.

We will use standard bounds on the concentration of moments of a probability distribution, namely, Markov's inequality, Chebyshev's inequality, and the Chernoff bound (cf. Motwani & Raghavan 1994).

The Lower Bound

We begin by arguing using Markov's inequality that as more and more XOR constraints are added at random from $\mathbb{X}^k$ for *any* k to a formula F, its model count, $MC(F) = 2^{s^*}$, is quite likely to go down *at least* nearly as fast as expected. Later, in the upper bound section, we will argue that $MC(F)$ is likely to go down *at most* nearly as fast as expected when $k = n/2$. The precision slack α captures the notion of "nearly" as fast.

Lemma 1. *For* $1 \leq k \leq n/2$ *and* $\alpha \geq 1$, $\Pr[MC(F_s^k) \geq MC(F)/2^{s-\alpha}] \leq 2^{-\alpha}$.

Proof. Let S be the set of satisfying assignments for F; $|S| = MC(F)$. For each $\sigma \in S$, let $Y_\sigma = \sigma(F_s^k)$ be a 0-1 random variable. The expected value of Y_σ is the probability that σ satisfies all of the s XOR constraints in Q_s^k. Recall that σ satisfies exactly half the constraints in $\mathbb{X}^k$. Since the s constraints in Q_s^k are chosen uniformly and independently from $\mathbb{X}^k$, the probability that σ satisfies all of them is 2^{-s}, implying $\mathbb{E}[Y_\sigma] = 2^{-s}$.

Let $Y = \sum_\sigma Y_\sigma$. The random variable Y equals $MC(F_s^k)$, and we have $\mathbb{E}[Y] = \mathbb{E}[\sum_\sigma Y_\sigma] = \sum_\sigma \mathbb{E}[Y_\sigma] = \sum_\sigma 2^{-s} = MC(F)/2^s$. It follows that $\Pr[MC(F_s^k) \geq MC(F)/2^{s-\alpha}] = \Pr[Y \geq 2^\alpha \mathbb{E}[Y]] \leq 2^{-\alpha}$ by Markov's inequality. □

Corollary 1. *For* $1 \leq k \leq n/2, \alpha \geq 1$, *and* $s \geq s^* + \alpha$, $\Pr[F_s^k$ *is satisfiable*$] \leq 2^{-\alpha}$.

Proof. Observe that $MC(F)/2^{s-\alpha} = 2^{s^*-(s-\alpha)} \leq 1$. Therefore, $\Pr[F_s^k \text{ is satisfiable}] = \Pr[MC(F_s^k) \geq 1] \leq \Pr[MC(F_s^k) \geq MC(F)/2^{s-\alpha}] \leq 2^{-\alpha}$. □

We use this result along with the Chernoff bound to show that after adding $s \geq s^* + \alpha$ XOR constraints from $\mathbb{X}_s^k$ *several times*, the fraction of instances F_s^k that are satisfiable is unlikely to be much more than $2^{-\alpha}$. Consequently, if one does see a significantly larger fraction of satisfiable instances than $2^{-\alpha}$ in this meta-experiment, then s is very likely to be less than $(s^* + \alpha)$, providing a high probability lower bound of $(s - \alpha)$ on s^*. Clearly, a weaker bound with a large α holds with a higher probability than a stronger one with a small α.

Formally, let $F_s^{k,(i)}, 1 \leq i \leq t$, denote t random formulas obtained by independently adding s random XOR constraints from $\mathbb{X}^k$ to F. The probability in what follows is on the collective choice of these formulas. In particular, whether $F_s^{k,(i)}$ is satisfiable or not is a random event of interest.

For convenience, we define the following quantity related to the Chernoff bound.

Definition 1. For any positive integer t, $0 < \delta \leq 1/2$, and $\alpha \geq 1$, let $\beta = 2^\alpha(1/2 + \delta) - 1$ and define

$$p(t, \delta, \alpha) = \begin{cases} 2^{-\alpha t} & \text{if } \delta = 1/2 \\ \left(\frac{e^\beta}{(1+\beta)^{1+\beta}}\right)^{t/2^\alpha} & \text{if } \delta < 1/2 . \end{cases}$$

When $\delta < 1/2$, we can simplify the above expression for $\alpha \in \{1, 2\}$ to get $p(t, \delta, 1) \leq e^{-0.77\delta^2 t}$ and $p(t, \delta, 2) \leq e^{-0.07(1+4\delta)^2 t} \leq e^{-1.12\delta^2 t - 0.07t}$. This demonstrates that $p(t, \delta, \alpha)$ decreases exponentially as δ and t increase, and is significantly smaller for larger α. This will help in qualitatively understanding the correctness guarantees we provide.

Lemma 2. *For* $s \geq s^* + \alpha$ *and* $0 < \delta \leq 1/2$, *the probability that at least a* $(1/2 + \delta)$ *fraction of the* t *formulas* $F_s^{k,(i)}, 1 \leq i \leq t$, *is satisfiable is at most* $p(t, \delta, \alpha)$.

Proof. Let $Z_i, 1 \leq i \leq t$, be a random variable whose value is 1 if $F_s^{k,(i)}$ is satisfiable and 0 otherwise. Let $Z = \sum_i Z_i$ be the random variable that equals the number of satisfiable formulas $F_s^{k,(i)}, 1 \leq i \leq t$. Note that Z is the sum of *independent* 0-1 random variables, and, by Chernoff bound, is highly concentrated around its expected value.

By Corollary 1, $\Pr[F_s^{k,(i)} \text{ is satisfiable}] \leq 2^{-\alpha}$. Therefore, $\mathbb{E}[Z_i] = \Pr[Z_i = 1] = \Pr[F_s^{k,(i)} \text{ is satisfiable}] \leq 2^{-\alpha}$ so that $\mathbb{E}[Z] = \sum_i \mathbb{E}[Z_i] \leq t2^{-\alpha}$.

The probability that at least a $(1/2 + \delta)$ fraction of these t random formulas is satisfiable equals $\Pr[Z \geq t \cdot (1/2 + \delta)]$. For $\delta = 1/2$, this equals $\Pr[Z \geq t] = \Pr[Z_i = 1 \text{ for all } i] \leq 2^{-\alpha t} = p(t, 1/2, \alpha)$ because the random variables Z_i are independent. For $\delta < 1/2$, $\Pr[Z \geq t \cdot (1/2 + \delta)] \leq \Pr[Z \geq 2^\alpha(1/2 + \delta)\mathbb{E}[Z]]$. Using the Chernoff bound, this probability is bounded above by $p(t, \delta, \alpha)$. □

Recall that $p(t, \delta, \alpha)$ is a quantity that decreases exponentially with t and δ.

Theorem 1 (Main Result). *For $1 \leq k \leq n/2$, the lower bound of $2^{s-\alpha}$ reported by* `MBound` *with parameters $(k, s, t, \delta, \alpha)$ is correct with probability at least $1 - p(t, \delta, \alpha)$.*

Proof. Suppose `MBound` with parameters $(k, s, t, \delta, \alpha)$ makes a lower bound error on input F. Let r denote the final value of the variable *numSat* in that run of `MBound`, i.e., r is the number of satisfiable formulas amongst the t random formulas generated by the algorithm.

Since `MBound` reported a (wrong) lower bound, it must be that $r \geq t \cdot (\frac{1}{2} + \delta)$. Further, since it made an error, it must be that $\log_2 MC(F) \leq s - \alpha$ in reality, i.e., $s \geq s^* + \alpha$. In this case, by Lemma 2, the probability of the algorithm encountering $r \geq t \cdot (\frac{1}{2} + \delta)$ satisfiable formulas amongst the t random ones is bounded above by $p(t, \delta, \alpha)$. □

In particular, when *all* formulas encountered by `MBound` are satisfiable, we have a simpler correctness guarantee.

Corollary 2 (Simplified Result). *When* `MBound` *finds all t* CNFXOR *formulas to be satisfiable, the lower bound $2^{s-\alpha}$ it reports is correct with probability at least $1 - 2^{-\alpha t}$.*

Example 1. Consider an experiment with $t = 20$ runs and δ set to $1/4$. Then $t \cdot (\frac{1}{2} + \delta) = 15, p(t, \delta, 1) \leq 0.34$, and $p(t, \delta, 2) \leq 0.002$. It follows that if we observe at least 15 out of 20 runs to be satisfiable, the lower bound of 2^{s-1} is correct with probability at least 0.66, and that of 2^{s-2} is correct with probability at least 0.998.

This shows that the probability of `MBound` making a lower-bound error indeed goes down *exponentially* as the number of trials t increase, making the algorithm quite robust for providing lower bounds on the model count. Further, when the precision slack α is increased from 1 to 2, the error probability decreases dramatically.

The Upper Bound: Ideal Case, Large XORs

For the upper bound, Markov's inequality is insufficient. We instead argue using Chebyshev's inequality that as more and more XOR constraints are added at random from $\mathbb{X}^{n/2}$ to a formula F, its model count is quite likely to go down *at most* nearly as fast as expected. In particular, F is quite unlikely to become unsatisfiable with too few XORs. Note that this result as such *does not hold* when small XORs are used.

Lemma 3. *For $\alpha \geq 1$ and $s \leq s^*$, (A) $\Pr[MC(F_s^{n/2}) \leq MC(F)/2^{s+\alpha}] \leq 1/((1-2^{-\alpha})^2 2^{s^*-s})$ and (B) $\Pr[F_s^{n/2}$ is unsatisfiable$] \leq 1/2^{s^*-s}$.*

Proof Sketch. (See Appendix for details.) As in the proof of Lemma 1, let S be the set of satisfying assignments for F. For each $\sigma \in S$, let $Y_\sigma = \sigma(F_s^{n/2})$ be a 0-1 random variable. The expected value of Y_σ is, as before, given by $\mathbb{E}[Y_\sigma] = 2^{-s}$. Further, its variance is $\text{Var}[Y_\sigma] = \mathbb{E}[Y_\sigma^2] - \mathbb{E}[Y_\sigma]^2$. Ignoring the negative term and using the fact that Y_σ is a 0-1 variable, $\text{Var}[Y_\sigma] \leq \mathbb{E}[Y_\sigma]$.

A key thing to observe here is that the random variables Y_σ for various σ are *pairwise-independent* because of an argument that relies on both the fact that we are dealing with XOR constraints (as opposed to, say, CNF constraints) and that they are chosen uniformly from $\mathbb{X}^{n/2}$ rather than from $\mathbb{X}^k$ for $k < n$. The result then follows from a variance computation and an application of Chebyshev's inequality. □

Corollary 3. *For $\alpha \geq 1$ and $s \leq s^* - \alpha$, $\Pr[F_s^{n/2}$ is unsatisfiable$] \leq 2^{-\alpha}$.*

The meta-experiment providing an upper bound on s^* works essentially the same as Lemma 2 (see Appendix).

Lemma 4. *For $s \leq s^* - \alpha$ and $0 < \delta \leq \frac{1}{2}$, the probability that at least a $(\frac{1}{2} + \delta)$ fraction of the t formulas $F_s^{n/2,(i)}, 1 \leq i \leq t$, is unsatisfiable is at most $p(t, \delta, \alpha)$.*

From this follows our main upper bound result for large XORs, in a fashion very similar to the proof of Theorem 1.

Theorem 2. *An upper bound of $2^{s+\alpha}$ reported by* `MBound` *with parameters $(n/2, s, t, \delta, \alpha)$ is correct with probability at least $1 - p(t, \delta, \alpha)$.*

The Upper Bound: Practical Case, Small XORs

As mentioned earlier, computation with large XORs is quite expensive in practice. While the *correctness* of the lower bound reported by `MBound` does not depend on the length of the XORs, that of the upper bound does. When the solution space is highly structured, small XORs do not act pairwise-independently on various variable assignments.

After adding $s \leq s^* - 2$ small random XORs to F, while one still expects 2^{s^*-s} solutions of F to survive on average, the *variance* in this number could be quite high. In the worst case, one could have a tiny number of resulting formulas $F_s^{k,(i)}$ be satisfiable with an enormous number of solutions, and a huge number of such formulas be unsatisfiable. This would still maintain the expected number of surviving solutions, but would make the sat-unsat distribution of $F_s^{k,(i)}$ highly skewed towards unsat even with too few XORs.

On the positive side, as k increases and approaches $n/2$, one expects random XORs from $\mathbb{X}^k$ to act on different variable assignments in a more and more pairwise independent manner. This can be proved formally using a straightforward variance calculation. We omit the proof for lack of space.

Proposition 1. *As the length k of* XORs *increases, the variance in the number of satisfiable* CNFXOR *formulas observed by* `MBound` *decreases.*

A Hybrid Model Counting Algorithm

By using a good SAT solver as a subroutine, `MBound` is already able to provide high quality lower bounds in practice. Its performance can be boosted even further by combining it with an exact model counting algorithm. Algorithm 2, `Hybrid-MBound`, that we present in this section does precisely this. The idea is to add randomly chosen XORs as before, but solve the resulting formula using an *exact model counting algorithm* as a subroutine instead of a SAT solver.

Two key factors make the hybrid approach work well in practice. First, by streamlining hard-to-count formulas with random XORs, we bring them within the reach of exact counting methods while maintaining the accuracy of the

Algorithm 2: Hybrid-MBound

```
Params: $k, s, t, \alpha$ : $k, s, t$ positive integers, $k \leq n/2,\ \alpha \geq 1$
Mode   : Conservative, Moderate, or Aggressive
Input  : A CNF formula $F$
Output : A lower and an upper bound on $MC(F)$
begin
    $numSeq \leftarrow empty$
    for $i \leftarrow 1$ to $t$ do
        $Q_s \leftarrow \{s$ random constraints from $\mathbb{X}^k\}$
        $F_s^{k,(i)} \leftarrow F \cup Q_s$
        $numModels \leftarrow$ ExactModelCount($F_s^{k,(i)}$)
        $numSeq$.PushBack($numModels$)
    $minModels \leftarrow$ Min($numSeq$)
    $maxModels \leftarrow$ Max($numSeq$)
    $avgModels \leftarrow$ Average($numSeq$)
    if $mode =$ Conservative then
        return Lower bound: $MC(F) > 2^{s-\alpha} \cdot minModels$,
               Upper bound: $MC(F) < 2^{s+\alpha} \cdot maxModels$
    else if $mode =$ Moderate then
        return Lower bound: $MC(F) > 2^{s-\alpha} \cdot avgModels$,
               Upper bound: $MC(F) < 2^{s+\alpha} \cdot avgModels$
    else                    /* $mode =$ Aggressive */
        return Lower bound: $MC(F) > 2^{s-\alpha} \cdot maxModels$,
               Upper bound: $MC(F) < 2^{s+\alpha} \cdot minModels$
end
```

overall bound. Second, by throwing in relatively fewer XORs and relying on an exact counting method for the residual formula, the quality of the obtained bound is improved.

Hybrid-MBound has a subset of the parameters of MBound and has only a single mode of termination: return both a lower and an upper bound on $MC(F)$, within a factor of $2^{2\alpha+1}$ of each other. Once Hybrid-MBound generates exact counts for t streamlined formulas, the overall bound it reports can naturally be based on the minimum, average, or maximum of the t residual counts. We call this the *mode* of operation. The correctness of Hybrid-MBound is captured by the following results (see Appendix for proofs).

Theorem 3 (Main Hybrid Result). *For $1 \leq k \leq n/2$, when Hybrid-MBound is run with parameters (k, s, t, α), then*
$\Pr[$*Conservative lower bound is correct*$] \geq 1 - 2^{-\alpha t}$,
$\Pr[$*Moderate lower bound is correct*$] \geq 1 - 2^{-\alpha}$, *and*
$\Pr[$*Aggressive lower bound is correct*$] \geq (1 - 2^{-\alpha})^t$.

Theorem 4. *The lower and upper bounds reported by Hybrid-MBound with parameters $(n/2, s, t, \alpha)$ are correct with probability at least $1 - 1/2^{s^*-s-2}$ independent of t, α.*

Experimental Results

To demonstrate the practical relevance of our approach, we considered the model counting problem for three hard combinatorial domains: the Ramsey problem, the Schur problem, and the clique coloring problem. All three problems deal with the question of the existence of certain intricate combinatorial objects.

In the *Ramsey domain*, one considers all possible two-colorings (red and blue) of the edges of the complete graph on n nodes. Ramsey showed that when n gets sufficiently large, certain structures of red or blue edges will be found in every coloring. In particular, $R(k,l)$ denotes the minimum value of n such that every coloring has at least one red clique of k vertices *or* one blue clique of l vertices. It is known that $R(4,5) = 25$. So, if we consider a complete graph of 23 vertices, we are guaranteed to have solutions that neither contain a red clique of size 4 nor a blue clique of size 5. However, this is a highly non-trivial coloring problem. We can translate this problem into a SAT formula with 253 variables and 42,504 clauses. Finding a single solution using the fastest available SAT solver for this problem (MiniSAT) takes approximately 30 seconds on a 1 GHz machine. Our challenge is to find an interesting lower bound on the number of solutions.

We used MBound with small XOR constraints and MiniSAT as the subsolver. The parameters were chosen so as to make the streamlined formula easy for MiniSAT. In particular, when the XORs are too large, they do not provide enough constraint propagation for MiniSAT. (Note that the XORs are converted into CNF using auxiliary variables and clauses.) We found that we could streamline with 65 random XOR constraints with 4 to 5 variables in each constraint.

In the *Schur problem*, we are given the set of integers $\{1, 2, \ldots, n\}$. The question is whether this set can be divided into k sum-free subsets. A set S is sum-free if the sum of any pair of numbers in S is not in S. For each value of k, there is a certain value of n such that no partition into k sum-free subsets exists. For given values of n and k, we can again construct a SAT problem representing the formula. It is known that for $k = 5$ and $n = 140$, sum-free partitions still exist. The corresponding SAT problem has 700 variables and 51,600 clauses. This formula is already beyond the reach of current state-of-the-art SAT solvers (i.e., we could not find a single model in approximately 12 hours of CPU time). With random XOR streamlining with good parameters, we were able to solve the instance using MiniSAT by adding 27 XOR constraints containing an average of 9 variables.

In the *clique coloring problem* with parameters n, m, and k, the task is to construct a graph on n nodes such that it can be colored with m colors and also contains a clique of size k. This problem has interesting properties that make it very useful in proof complexity research on exponential lower bounds for powerful proof systems (Pudlák 1997). We experimented with instances that had 600-750 variables and 20,000-35,000 clauses. When satisfiable, finding a single solution to these is quite easy. However, counting all solutions turns out to be extremely challenging even for approximate methods. By streamlining with XORs of size 6-8, we obtained lower bounds of 10^{40} and higher within minutes.

Table 1 summarizes the results obtained[1] on a 550 MHz 8 processor Intel Pentium III machine with 4 GB shared memory. All reported lower bounds are based on $t = 7$ and $\alpha = 1$ so that Corollary 2 guarantees a $1 - 2^{-7} \geq 99\%$ confidence. The confidence level can, of course, be boosted by simply doing more XOR streamlined runs with MiniSAT (higher t) or reducing the reported bound by a factor of 2 (higher α).

[1]The code and complete data are available from the authors.

Table 1: MBound on problems beyond the reach of exact counting methods (99% confidence). Note that ApproxCount does not provide any guarantee on correctness or accuracy.

Instance	MBound		Relsat		Cachet		ApproxCount	
	Models	Time	Models	Time	Models	Time	Models	Time
Ramsey-20-4-5	$\geq \mathbf{1.2 \times 10^{30}}$	**< 2 hrs**	$\geq 7.1 \times 10^{8}$	12 hrs	—	12 hrs	$\approx 1.8 \times 10^{19}$	4 hrs
Ramsey-23-4-5	$\geq \mathbf{1.8 \times 10^{19}}$	**< 2 hrs**	$\geq 1.9 \times 10^{5}$	12 hrs	—	12 hrs	$\approx 7.7 \times 10^{12}$	5 hrs
Schur-5-100	$\geq \mathbf{2.8 \times 10^{14}}$	**< 2 hrs**	—	12 hrs	—	12 hrs	$\approx 2.3 \times 10^{11}$	7 hrs
Schur-5-140	$\geq \mathbf{6.7 \times 10^{7}}$	**< 5 hrs**	—	12 hrs	—	12 hrs	—	12 hrs
fclqcolor-18-14-11	$\geq \mathbf{2.1 \times 10^{40}}$	**3 mins**	$\geq 2.8 \times 10^{26}$	12 hrs	—	12 hrs	—	12 hrs
fclqcolor-20-15-12	$\geq \mathbf{2.2 \times 10^{46}}$	**9 mins**	$\geq 2.3 \times 10^{20}$	12 hrs	—	12 hrs	—	12 hrs

We see that we obtain non-trivial lower bounds on the model counts. We also see that these counting problems are effectively beyond the reach of other state-of-the-art model counting approaches. Both Relsat and Cachet do not finish in 12 hours. Relsat gives very low partial counts while Cachet is not designed to report partial counts. ApproxCount computes a medium quality approximate count without any guarantees. *These results show that counting using randomized* XOR *streamlining provides a powerful new approach for obtaining lower bounds on model counts of hard combinatorial problems.*

Often lower bounds obtained from MBound can be made stronger with Hybrid-MBound. With Cachet as a subsolver and 30 XORs, we could boost the lower bound model count for Schur-5-140 to 1.8×10^{12}. Similarly, the lower bound for fclqcolor-18-14-11 was improved to 4.1×10^{45}. Note that the ability of Hybrid-MBound to boost MBound relies partly on exact model counting technology. The latter generally lags quite a bit behind SAT solvers, which are sufficient for MBound.

Finally, the results summarized in Table 2 confirm that, in practice, lower bounds reported by MBound and Hybrid-MBound can come quite close to exact counts even with very small XORs. The instance bitmax is a circuit synthesis problem and log_a is a logistics planning problem. The exact counts for these are obtained using Relsat. The last two instances are pigeonhole-type problems, with n pigeons and k holes, $n \leq k$. Relsat timed out on these instances after 12 hours. However, the exact count can be analytically computed to be $k!/(k-n)!$.

Table 2: Comparison of lower bounds with exact counts. The XOR size column reports the average.

prob.	num vars	exact count	XOR size	MBound	Hybrid-MBound
bitmax	252	21×10^{28}	9	$\geq 1.9\times10^{28}$	$\geq 9.2\times10^{28}$
log_a	1719	26×10^{15}	36	$\geq 1.1\times10^{15}$	$\geq 11\times10^{15}$
php.10.20	200	6.7×10^{11}	17	$\geq 1.3\times10^{11}$	$\geq 2.9\times10^{11}$
php.15.20	300	20×10^{15}	20	$\geq 1.1\times10^{15}$	—

In all four cases, the average length of XORs used was less than 5% of the number of problem variables. Nevertheless, MBound came within a factor of 20 of the exact counts, which exceed 10^{11} and sometimes even 10^{28}. Hybrid-MBound typically requires shorter XORs than MBound in order to make the streamlined formula solvable using an exact counting method. Despite this, it further boosted the results to within a factor of 2 of the exact counts in three cases; in the fourth, even the streamlined problem remained hard for Relsat. Of course, one could yet further improve these bounds using somewhat larger XORs and additional computational resources.

Conclusion

Current techniques for model counting are based on either an exact counting paradigm or an approximate counting approach (e.g., MCMC methods), both of which have their limitations. We propose a third alternative based on randomized streamlining using XOR constraints. Our approach has two key strengths: it can generate model counts using any state-of-the-art SAT solver off-the-shelf, and it provides concrete bounds along with a high probability correctness guarantee that can be easily boosted by repetition. The model count lower bounds obtained using our algorithm MBound dramatically improve upon the results of existing techniques on three very difficult combinatorial problems. Our algorithm Hybrid-MBound combines the strength of existing exact counting methods with our XOR streamlining approach, boosting the results even further.

References

Bayardo Jr., R. J., and Pehoushek, J. D. 2000. Counting models using connected components. In *17th AAAI*, 157–162.

Darwiche, A. 2005. The quest for efficient probabilistic inference. Invited Talk, IJCAI-05.

Gomes, C. P., and Sellmann, M. 2004. Streamlined constraint reasoning. In *10th CP*, volume 3258 of *LNCS*, 274–289.

Littman, M. L.; Majercik, S. M.; and Pitassi, T. 2001. Stochastic boolean satisfiability. *J. Auto. Reasoning* 27(3):251–296.

Motwani, R., and Raghavan, P. 1994. *Randomized Algorithms*. Cambridge University Press.

Pudlák, P. 1997. Lower bounds for resolution & cutting plane proofs & monotone computations. *J. Symb. Logic* 62(3):981–998.

Roth, D. 1996. On the hardness of approximate reasoning. *J. AI* 82(1-2):273–302.

Sang, T.; Bacchus, F.; Beame, P.; Kautz, H. A.; and Pitassi, T. 2004. Combining component caching and clause learning for effective model counting. In *7th SAT*. Online Proceedings.

Toda, S. 1989. On the computational power of PP and $\oplus$P. In *30th FOCS*, 514–519.

Valiant, L. G., and Vazirani, V. V. 1986. NP is as easy as detecting unique solutions. *Theoretical Comput. Sci.* 47(3):85–93.

Wei, W., and Selman, B. 2005. A new approach to model counting. In *8th SAT*, volume 3569 of *LNCS*, 324–339.

Appendix: Proofs

Proof of Lemma 3. As in the proof of Lemma 1, let S be the set of satisfying assignments for F; $|S| = MC(F)$. For each $\sigma \in S$, let Y_σ be the 0-1 random variable whose value is $\sigma(F_s^{n/2})$. The expected value of Y_σ is, as before, given by $\mathbb{E}[Y_\sigma] = 1/2^s$. Further, its variance is $\mathrm{Var}[Y_\sigma] = \mathbb{E}[Y_\sigma^2] - \mathbb{E}[Y_\sigma]^2$. Ignoring the negative term and using the fact that Y_σ is a 0-1 variable, $\mathrm{Var}[Y_\sigma] \le \mathbb{E}[Y_\sigma]$.

A key point to observe here is that the random variables Y_σ for various σ are *pairwise-independent* because of the following argument which relies on both the fact that we are dealing with XOR constraints (as opposed to, say, CNF constraints) and that they are chosen uniformly from $\mathbb{X}^{n/2}$ rather than from $\mathbb{X}^k$ for $k < n$. Consider two random variables, Y_{σ_1} and Y_{σ_2}. The question we need to answer is: what can we say about the value of Y_{σ_2} when we know the value of Y_{σ_1}? The answer is determined by the behavior of σ_1 and σ_2 on *any single* XOR constraint D. We will show that for D chosen uniformly from $\mathbb{X}^{n/2}$, $\sigma_1(D)$ differs from $\sigma_2(D)$ with probability exactly a $1/2$. This will imply that knowing the value of Y_{σ_1} does not tell us anything about the value of Y_{σ_2}.

To this end, let $V' \subseteq V$ be the non-empty set of variables on which σ_1 and σ_2 differ, and D be an random XOR constraint from $\mathbb{X}^{n/2}$. Since D is an XOR constraint, the probability that $\sigma_1(D)$ differs from $\sigma_2(D)$ equals the probability that D involves an odd number of variables in V'. Since D is chosen at random from $\mathbb{X}^{n/2}$ and exactly half the constraints in $\mathbb{X}^{n/2}$ involve an odd number of variables in any subset of V, this probability is exactly $1/2$. As argued above, this implies that Y_{σ_1} and Y_{σ_2} are pairwise-independent.

Getting back to the main argument, let $Y = \sum_\sigma Y_\sigma$. The random variable Y equals the number of models of the formula $F_s^{n/2}$. The expected value of Y is given by $\mathbb{E}[Y] = \mathbb{E}[\sum_\sigma Y_\sigma] = \sum_\sigma \mathbb{E}[Y_\sigma] = \sum_\sigma 1/2^s = MC(F)/2^s$. Further, since Y is the sum of pairwise-independent random variables, its variance is given by $\mathrm{Var}[Y] = \sum_\sigma \mathrm{Var}[Y_\sigma]$. Recall that $\mathrm{Var}[Y_\sigma] \le \mathbb{E}[Y_\sigma]$. Hence, $\mathrm{Var}[Y] \le \sum_\sigma \mathbb{E}[Y_\sigma] = \mathbb{E}[Y]$.

For part (A), $\Pr[MC(F_s^{n/2}) \le MC(F)/2^{s+\alpha}] = \Pr[Y \le \mathbb{E}[Y]/2^\alpha] \le \Pr[|Y - \mathbb{E}[Y]| \ge (1-2^{-\alpha})\mathbb{E}[Y]]$. By Chebyshev's inequality, this last expression is at most $\mathrm{Var}[Y]/((1-2^{-\alpha})^2\mathbb{E}[Y]^2) \le 1/((1-2^{-\alpha})^2\mathbb{E}[Y])$. For part (B), $\Pr[F_s^{n/2}$ is unsatisfiable$] = \Pr[Y = 0] = \Pr[|Y - \mathbb{E}[Y]| \ge \mathbb{E}[Y]]$. Again, by Chebyshev's inequality, this last expression is at most $\mathrm{Var}[Y]/\mathbb{E}[Y]^2 \le 1/\mathbb{E}[Y] \le 1/2^{s^*-s}$. □

Proof of Lemma 4. Let $Z_i, 1 \le i \le t$, be a random variable whose value is 1 if $F_s^{n/2,(i)}$ is *un*satisfiable and 0 otherwise. Let $Z = \sum_i Z_i$ be the random variable that equals the number of unsatisfiable formulas $F_s^{n/2,(i)}, 1 \le i \le t$. Note that Z is the sum of *independent* 0-1 random variables, and, by Chernoff bound, is highly concentrated around its expected value.

By Corollary 3, $\Pr[F_s^{n/2,(i)}$ is unsatisfiable$] \le 1/2^\alpha$. Therefore, $\mathbb{E}[Z_i] = \Pr[Z_i = 1] = \Pr[F_s^{n/2,(i)}$ is unsatisfiable$] \le 1/2^\alpha$ so that $\mathbb{E}[Z] = \sum_i \mathbb{E}[Z_i] \le t/2^\alpha$.

The probability that at least a $(1/2+\delta)$ fraction of these t random formulas is unsatisfiable equals $\Pr[Z \ge t \cdot (1/2+\delta)]$. For $\delta = 1/2$, this is $\Pr[Z \ge t] = \Pr[Z_i = 1$ for all $i] \le 1/2^{\alpha t} = p(t, 1/2, \alpha)$ because the Z_i are independent. For $\delta < 1/2$, $\Pr[Z \ge t \cdot (1/2+\delta)] \le \Pr[Z \ge 2^\alpha(1/2+\delta)\mathbb{E}[Z]]$. Using the Chernoff bound separately for $\alpha = 1, 2$ as in Lemma 2, this probability is bounded above by $p(t, \delta, \alpha)$. □

Proof of Theorem 3. Suppose `HybridMC` makes a lower bound error in Conservative mode, that is, $MC(F) \le 2^{s-\alpha} \cdot minModels$. Equivalently, $minModels \ge MC(F)/2^{s-\alpha}$ so that *all* t of the random formulas $F_s^{k,(i)}, 1 \le i \le t$, have model counts at least $MC(F)/2^{s-\alpha}$. By Lemma 1, this happens for a single such formula with probability at most $2^{-\alpha}$, and, by independence, for all of them with probability at most $2^{-\alpha t}$.

From the opposite perspective, the lower bound reported in Aggressive mode is correct iff *all* t random formulas $F_s^{k,(i)}$ correctly have model counts less than $MC(F)/2^{s-\alpha}$. By Lemma 1, this happens for a single such formula with probability at least $1-2^{-\alpha}$, and, by independence, for all of them with probability at least $(1-2^{-\alpha})^t$.

The result for Moderate mode can be proved using a variant of Lemma 1. Specifically, for the i^{th} trial, we have a 0-1 random variable Y_σ^i instead of Y_σ in the proof of Lemma 1, and now $Y = \sum_i \sum_\sigma Y_\sigma^i$ is the total number of solutions of all t random formulas. $\mathbb{E}[Y] = t \cdot MC(F)/2^s$ and the result follows from Markov's inequality as before. □

Proof of Theorem 4. We will focus on the hardest case, namely, $t = \alpha = 1$. Let $F_s^{n/2}$ denote the CNFXOR formula generated by `HybridMC` on input F, so that $minModels = maxModels = avgModels = MC(F_s^{n/2})$. Suppose the algorithm makes an error, that is, either $MC(F) \le 2^{s-1}MC(F_s^{n/2})$ or $MC(F) \ge 2^{s+1}MC(F_s^{n/2})$. We will show that these events occur with a low probability.

The proof uses a slight generalization of Lemma 3. Assume the same setup as in the proof of that lemma, namely, a random variable Y that equals $MC(F_s^{n/2})$ and whose mean and variance are $\mathbb{E}[Y] = MC(F)/2^s$ and $\mathrm{Var}[Y] \le \mathbb{E}[Y]$, respectively. We then have $\Pr[MC(F) \ge 2^{s+1}MC(F_s^{n/2})] = \Pr[MC(F_s^{n/2}) \le MC(F)/2^{s+1}] = \Pr[Y \le \mathbb{E}[Y]/2]$. Similarly, $\Pr[MC(F) \le 2^{s-1}MC(F_s^{n/2})] = \Pr[MC(F_s^{n/2}) \ge MC(F)/2^{s-1}] = \Pr[Y \ge 2\mathbb{E}[Y]]$.

Considering the two error modes of the algorithm, $\Pr[$`HybridMC` makes an error$] = \Pr[Y \le \mathbb{E}[Y]/2$ or $Y \ge 2\mathbb{E}[Y]] \le \Pr[|Y - \mathbb{E}[Y]| \ge \mathbb{E}[Y]/2]$. By Chebyshev's inequality, this probability is at most $\mathrm{Var}[Y]/(\mathbb{E}[Y]/2)^2 \le 4/\mathbb{E}[Y] = 1/2^{s^*-s-2}$. □