Approximate Counting by Sampling the Backtrack-free Search Space

Vibhav Gogate and Rina Dechter
Donald Bren School of Information and Computer Science,
University of California, Irvine, CA 92697,
{vgogate,dechter}@ics.uci.edu

Abstract

We present a new estimator for counting the number of solutions of a Boolean satisfiability problem as a part of an importance sampling framework. The estimator uses the recently introduced SampleSearch scheme that is designed to overcome the rejection problem associated with distributions having a substantial amount of determinism. We show here that the sampling distribution of SampleSearch can be characterized as the backtrack-free distribution and propose several schemes for its computation. This allows integrating SampleSearch into the importance sampling framework for approximating the number of solutions and also allows using SampleSearch for computing a lower bound measure on the number of solutions. Our empirical evaluation demonstrates the superiority of our new approximate counting schemes against recent competing approaches.

Introduction

In this paper we present a search-based sampling algorithm for approximating the number of solutions of a Boolean Satisfiability problem. Solution counting is a well known #P-complete problem and has many applications in fields such as verification, planning and automated reasoning.

Earlier approaches to counting solutions are based on either extending systematic search-based SAT/CSP solvers such as DPLL (Bayardo & Pehoushek 2000), or variable-elimination algorithms which are known to be time and space exponential in the treewidth of the problem. When the treewidth is large variable-elimination was approximated by the mini-bucket algorithm (Dechter & Rish 2003) or by generalized belief propagation (Gogate & Dechter 2005).

A relatively new approach *ApproxCount* introduced by (Wei & Selman 2005) uses a combination of random walk and Markov Chain Monte Carlo (MCMC) sampling to compute an approximation of the exact solution count. *ApproxCount* was shown to scale quite well with problem size yielding good approximations on many problems. *ApproxCount* was recently modified to produce lower bounds on the exact solution counts by using a simple application of the Markov inequality (Gomes *et al.* 2007).

We present an alternative approximation which instead of using MCMC sampling uses *importance sampling* (Geweke 1989). However, a straight-forward application of importance sampling may lead to poor performance. The problem is that when the underlying probability distribution is not strictly positive, many of the generated samples may have zero probability and will be rejected, leading to an inefficient sampling process.

With the exception of the work on adaptive sampling schemes (Cheng & Druzdzel 2000; Yuan & Druzdzel 2006), the rejection problem has been largely ignored in the Statistics community. Recently, (Gogate & Dechter 2005) showed that a restricted form of constraint propagation can be used to reduce the amount of rejection. If the SAT problem is loosely constrained, this method worked quite well yielding relatively small error. However, when the underlying SAT problem is hard, it was observed that the method fails to generate even a single sample having non-zero weight.

More recently (Gogate & Dechter 2006) initiated a new approach. They try to circumvent the rejection problem by systematically searching for a non-zero weight sample until one is found. Only then, the generation of a new sample is initiated. We call this class of sampling schemes which combine backtracking search with sampling as *SampleSearch*. *SampleSearch* which was originally presented for the task of random solution generation is extended here in several ways. First our focus is on solution counting. Second, we provide the theoretical foundations for this scheme which were missing in (Gogate & Dechter 2006), proving it to be an "importance sampling" scheme with its desirable theoretical guarantees. In particular, importance sampling requires that the underlying sampling distribution from which the algorithm samples, be known. We characterize the sampling distribution of *SampleSearch* as the *backtrack-free* distribution. Third, we propose an approximation of the backtrack-free distribution when it is hard to compute, while still maintaining the property of *asymptotic unbiasedness* (Rubinstein 1981). Fourth, we modify *SampleSearch* to yield a lower bound measure on the number of solutions in a similar way to (Gomes *et al.* 2007). We present empirical evaluation of our new scheme against state-of-the-art methods and show that our new scheme outperforms the *ApproxCount* scheme (Wei & Selman 2005) on most instances and that our new lower-bounds are more accurate than that of (Gomes *et al.* 2007) on most instances.

Background

We represent sets by bold capital letters and members of a set by capital letters. An assignment of a value to a variable is denoted by a small letter while bold small letters indicate an assignment to a set of variables.

For the rest of the paper, let $|\mathbf{X}| = n$ be the propositional variables. A variable assignment $\mathbf{X} = \mathbf{x}$, $\mathbf{x} = (x_1, \ldots, x_n)$ assigns a value in $\{0,1\}$ to each variable in $\mathbf{X}$. We use the notation $\overline{x_i}$ to mean the negation of a value assignment x_i. Let $F = F_1 \wedge \ldots \wedge F_m$ be a formula in conjunctive normal form (cnf) with clauses $F_1, \ldots, F_m$ defined over $\mathbf{X}$ and let $\mathbf{X} = \mathbf{x}$ be a variable assignment. If $\mathbf{X} = \mathbf{x}$ satisfies all clauses of F, then $\mathbf{X} = \mathbf{x}$ is a model or a solution of F. We define $F(\mathbf{x}) = 1$ if $\mathbf{x}$ is a solution of F and $F(\mathbf{x}) = 0$ otherwise. Let $\mathbf{S} = \{\mathbf{X} = \mathbf{x} | F(\mathbf{x}) = 1\}$ be the set of models of formula F. The counting task is to compute $|\mathbf{S}|$.

Importance Sampling

Given a function $g(\mathbf{x})$ defined over the domain Ω, importance sampling (Rubinstein 1981) is a common technique used to estimate the sum: $M = \sum_{\mathbf{x} \in \Omega} g(\mathbf{x})$. Given a proposal distribution $Q(\mathbf{x})$ over the domain Ω, we can rewrite the M as:

$$M = \sum_{\mathbf{x} \in \Omega} \frac{g(\mathbf{x})}{Q(\mathbf{x})} Q(\mathbf{x}) = \mathbb{E}_Q(\frac{g(\mathbf{x})}{Q(\mathbf{x})})$$

where $\mathbb{E}_Q(\frac{g(\mathbf{x})}{Q(\mathbf{x})})$ denotes the expected value of the random variable $\frac{g(\mathbf{x})}{Q(\mathbf{x})}$ with respect to the distribution Q. The idea in importance sampling is to generate independently and identically distributed (i.i.d.) samples $(\mathbf{x}^1, \ldots, \mathbf{x}^N)$ from the proposal distribution $Q(\mathbf{x})$ such that $g(\mathbf{x}) > 0 \Rightarrow Q(\mathbf{x}) > 0$ and then estimate M as follows:

$$\widehat{M} = \frac{1}{N} \sum_{i=1}^{N} \frac{g(\mathbf{x}^i)}{Q(\mathbf{x})} = \frac{1}{N} \sum_{i=1}^{N} w(\mathbf{x}^i), \ \ where \ \ w(\mathbf{x}^i) = \frac{g(\mathbf{x}^i)}{Q(\mathbf{x}^i)} \quad (1)$$

$w(\mathbf{x}^i)$ is referred to as the importance weight. It can be shown that the expected value $\mathbb{E}_Q(\widehat{M}) = M$ (Rubinstein 1981).

Another practical requirement of importance sampling is that $Q(\mathbf{x})$ is easy to sample from. Therefore as in (Cheng & Druzdzel 2000), we assume that the proposal distribution is expressed in a product form, $Q(\mathbf{x}) = \prod_{i=1}^{n} Q_i(x_i | x_1, \ldots, x_{i-1})$ so that we can generate an i.i.d. sample $\mathbf{x}$ along the ordering $O = \langle X_1, \ldots, X_n \rangle$ as follows. **For** $1 \leq i \leq n$, Sample $X_i = x_i$ from $Q_i(x_i | x_1, \ldots, x_{i-1})$.

Estimating Solution Counts using Importance Sampling Let F be a formula defined over a set of variables $\mathbf{X}$ and Ω be the space of all possible variable-value assignments. We can rewrite the solution counting task as the sum: $|\mathbf{S}| = \sum_{\mathbf{x} \in \Omega} F(\mathbf{x})$. Given a proposal distribution $Q(\mathbf{x})$ over Ω such that $F(\mathbf{x}) > 0 \Rightarrow Q(\mathbf{x}) > 0$ [1] and i.i.d. samples $(\mathbf{x}^1, \ldots, \mathbf{x}^N)$ generated from $Q(\mathbf{x})$, we can estimate the number of solutions as follows:

$$\overline{M} = \frac{1}{N} \sum_{i=1}^{N} \frac{F(\mathbf{x}^i)}{Q(\mathbf{x}^i)} = \frac{1}{N} \sum_{i=1}^{N} w(\mathbf{x}^i), \ \ where \ \ w(\mathbf{x}^i) = \frac{F(\mathbf{x}^i)}{Q(\mathbf{x}^i)} \quad (2)$$

[1] Note that throughout the paper we assume that Q satisfies $F(\mathbf{x}) > 0 \Rightarrow Q(\mathbf{x}) > 0$

It can be shown that $\mathbb{E}(\overline{M}) = |\mathbf{S}|$ (Rubinstein 1981).

The Rejection Problem

It is known that a straight-forward application of importance sampling may lead to very poor approximations (Gogate & Dechter 2005) and we explain why below. In the discussion on importance sampling, we assumed the presence of a proposal distribution $Q(\mathbf{x})$ which satisfies the condition $F(\mathbf{x}) > 0 \Rightarrow Q(\mathbf{x}) > 0$. In other words, $Q(\mathbf{x})$ can be greater than zero even if $F(\mathbf{x})$ is zero. This may be problematic because if the probability of generating a non-solution (i.e. a sample from $\{\mathbf{x} | F(\mathbf{x}) = 0\}$) substantially dominates the probability of generating a solution (i.e. a sample from $\{\mathbf{x} | F(\mathbf{x}) = 1\}$), a large number of samples generated from Q will have zero weight. These zero weight samples would not contribute to the sum in Equation 2, thereby effectively getting rejected (*the rejection problem*). In earlier work, (Gogate & Dechter 2005) proposed to address the rejection problem by enforcing bounded relational consistency. However, unless one enforces global consistency using an algorithm such as adaptive consistency, whose complexity is bounded exponentially by the treewidth, inconsistent solutions may still be generated and the rejection problem still exists.

To overcome the rejection problem, we (Gogate & Dechter 2006) recently proposed to augment sampling with search, yielding the *SampleSearch* scheme. Instead of returning with a sample that is inconsistent, *SampleSearch* progressively revises the inconsistent sample via backtracking search until a solution is found. Since the focus here is on SAT problems, we use here the conventional backtracking procedure for SAT which is the DPLL algorithm (Davis, Logemann, & Loveland 1962).

SampleSearch with DPLL works as follows (see Algorithm 1). It takes as input a formula F, an ordering $O = \langle X_1, \ldots, X_n \rangle$ of variables and a distribution $Q = \prod_{i=1}^{n} Q_i(x_i | x_1, \ldots, x_{i-1})$ along that ordering. Given a partial assignment $(x_1, \ldots, x_{i-1})$ already generated, the next variable in the ordering X_i is selected and its value $X_i = x_i$ is sampled from the conditional distribution $Q_i(x_i | x_1, \ldots, x_{i-1})$. Then the algorithm applies unit-propagation with the new unit clause $X_i = x_i$ created over the formula F. If no empty clause is generated, then the algorithm proceeds to the next variable. Otherwise, the algorithm tries $X_i = \overline{x_i}$, performs unit propagation and either proceeds forward (if no empty clause generated) or it backtracks. On termination, the output of

Algorithm 1 *SampleSearch* $SS(F, Q, O)$
Input: a cnf formula F, a distribution Q and Ordering O
Output: A solution $\mathbf{x} = (x_1, \ldots, x_n)$

1: UnitPropagate(F)
2: **if** there is an empty clause in F **then** Return 0
3: **if** all variables are assigned a value **then** Return 1
4: Select the earliest variable X_i in O not yet assigned a value
5: p = Generate a random real number between 0 and 1
6: **Value Assignment:** Given partial assignment $(x_1, \ldots, x_{i-1})$ **if** $p < Q_i(X_i = 0 | x_1, \ldots, x_{i-1})$ **then** set $X_i = 0$ **else** set $X_i = 1$
7: Return $SS((F \wedge x_i), Q, O) \vee SS((F \wedge \overline{x_i}), Q, O)$

SampleSearch is a solution of F (assuming F has a solution).

Analysis

In order to use *SampleSearch* within the importance sampling framework for estimating solution counts, we need to know the probability with which each sample (solution) is generated (see Equation 2). Therefore, in this section, we show that *SampleSearch* generates i.i.d. samples from a distribution which we characterize as the backtrack-free distribution (to be defined next).

Because *SampleSearch* samples from a distribution Q and outputs samples which are solutions of F, the sampling distribution denoted by Q^F satisfies $Q^F(\mathbf{x}|F(\mathbf{x})=1)>0$ and $Q^F(\mathbf{x}|F(\mathbf{x})=0)=0$. We next define the components Q_i^F, namely

Definition 1 (The Backtrack-free distribution)**.** Given a distribution $Q(\mathbf{x})=\prod_{i=1}^n Q_i(x_i|x_1,\ldots,x_{i-1})$, an ordering $O=\langle X_1,\ldots,X_n\rangle$ and a formula F, the backtrack-free distribution Q^F is factored into $Q^F(\mathbf{x})=\prod_{i=1}^n Q_i^F(x_i|x_1,\ldots,x_{i-1})$ where $Q_i^F(x_i|x_1,\ldots,x_{i-1})$ is defined as follows:

1. $Q_i^F(x_i|x_1,\ldots,x_{i-1})=0$ if $(x_1,\ldots,x_{i-1},x_i)$ cannot be extended to a solution of F.
2. $Q_i^F(x_i|x_1,\ldots,x_{i-1})=1$ if $(x_1,\ldots,x_{i-1},x_i)$ can be extended to a solution but $(x_1,\ldots,x_{i-1},\overline{x_i})$ cannot.
3. $Q_i^F(x_i|x_1,\ldots,x_{i-1}) = Q_i(x_i|x_1,\ldots,x_{i-1})$ if both $(x_1,\ldots,x_{i-1},x_i)$ and $(x_1,\ldots,x_{i-1},\overline{x_i})$ can be extended to a solution of F.

THEOREM **1.** *Given a distribution Q, the sampling distribution of SampleSearch is the backtrack-free distribution Q^F.*

Proof. Let $I(\mathbf{x})=\prod_{i=1}^n I_i(x_i|x_1,\ldots,x_{i-1})$ be the factored sampling distribution of *SampleSearch* (F,Q,O). We will prove that for any arbitrary partial assignment $(x_1,\ldots,x_{i-1},x_i)$, $I_i(x_i|x_1,\ldots,x_{i-1}) = Q_i^F(x_i|x_1,\ldots,x_{i-1})$. We consider three cases corresponding to the definition of the backtrack-free distribution (see Definition 1):

Case (1): $(x_1,\ldots,x_{i-1},x_i)$ cannot be extended to a solution. Since, all samples generated by *SampleSearch* are solutions of F, $I_i(x_i|x_1,\ldots,x_{i-1})=0=Q_i^F(x_i|x_1,\ldots,x_{i-1})$.

Case (2): $(x_1,\ldots,x_{i-1},x_i)$ can be extended to a solution but $(x_1,\ldots,x_{i-1},\overline{x_i})$ cannot be extended to a solution. Since *SampleSearch* is a systematic search procedure, if it explores a partial assignment $(x_1,\ldots,x_k)$ that can be extended to a solution of F, it is guaranteed to return a full sample (solution) extending this partial assignment. Otherwise, if $(x_1,\ldots,x_k)$ is not part of any solution of F then *SampleSearch* will prove this inconsistency before it will finish generating a full sample. Consequently, if $(x_1,\ldots,x_{i-1},x_i)$ is sampled first, a full assignment (a solution) extending $(x_1,\ldots,x_i)$ will be returned and if $(x_1,\ldots,\bar{x}_i)$ is sampled first, *SampleSearch* will detect that $(x_1,\ldots,x_{i-1},\overline{x_i})$ cannot be extended to a solution and eventually explore $(x_1,\ldots,x_{i-1},x_i)$. Therefore, in both cases if $(x_1,\ldots,x_{i-1})$ is explored by *SampleSearch*, a solution extending $(x_1,\ldots,x_{i-1},x_i)$ will definitely be generated i.e. $I_i(x_i|x_1,\ldots,x_{i-1})=1=Q_i^{\mathbf{F}}(x_i|x_1,\ldots,x_{i-1})$.

Case (3): Both $(x_1,\ldots,x_{i-1},x_i)$ and $(x_1,\ldots,x_{i-1},\overline{x_i})$ can be extended to a solution. Since *SampleSearch* is a systematic search procedure, if it samples a partial assignment $(x_1,\ldots,x_{i-1},\overline{x_i})$ it will return a solution extending $(x_1,\ldots,x_{i-1},\overline{x_i})$ without sampling $(x_1,\ldots,x_{i-1},x_i)$. Therefore, the probability of sampling x_i given $(x_1,\ldots,x_{i-1})$ equals $Q_i(x_i|x_1,\ldots,x_{i-1})$ (from Steps 5 and 6 of Algorithm 1) which equals $Q_i^F(x_i|x_1,\ldots,x_{i-1})$. □

Note that the only property of backtracking search that we have used in the proof is its *systematic nature*. Therefore, if we replace naive backtracking search by any systematic SAT solver such as minisat (Sorensson & Een 2005), the above theorem would hold. The only modifications we have to make are: (a) use static variable ordering [2], (b) use value ordering based on the proposal distribution Q. Consequently,

Corollary 1. *Given a Formula F, a distribution Q and an ordering O, any systematic SAT solver replacing DPLL in SampleSearch will generate i.i.d. samples from the backtrack-free distribution Q^F.*

If we can compute the sampling distribution Q^F, we will be able to estimate the solution counts as follows. Let $(\mathbf{x}^1,\ldots,\mathbf{x}^N)$ be a set of i.i.d. samples generated by *SampleSearch*, then as dictated by Equation 2, the number of solutions can be estimated by:

$$\overline{M}=\frac{1}{N}\sum_{i=1}^{N}\frac{F(\mathbf{x}^i)}{Q^F(\mathbf{x}^i)} \tag{3}$$

Because, all samples generated by *SampleSearch* are solutions, $\forall i$, $F(\mathbf{x}^i)=1$, we get

$$\overline{M}=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{Q^F(\mathbf{x}^i)} \tag{4}$$

Proposition 1. *(Geweke 1989; Rubinstein 1981) Let $\mathbf{S}=\{\mathbf{X}=\mathbf{x}|F(\mathbf{x})=1\}$ be the set of models of Formula F, then the expected value $\mathbb{E}(\overline{M})=|\mathbf{S}|$.*

Computing $Q^F(\mathbf{x})$ given a sample $\mathbf{x}=(x_1,\ldots,x_n)$

In this subsection, we describe an exact and an approximate scheme to compute $Q^F(\mathbf{x})$.

Exact Scheme Since $(x_1,\ldots,x_i,\ldots,x_n)$ was generated, clearly $(x_1,\ldots,x_i)$ can be extended to a solution and all we need is to determine if $(x_1,\ldots,\bar{x}_i)$ can be extended to a solution (see Definition 1). To that end, we can run any complete SAT solver on the formula $(F\wedge x_1\wedge\ldots\wedge x_{i-1}\wedge\overline{x_i})$. If the solver proves that $(F\wedge x_1\wedge\ldots\wedge x_{i-1}\wedge\overline{x_i})$ is consistent, we set $Q_i^F(x_i|x_1,\ldots,x_{i-1})=Q_i(x_i|x_1,\ldots,x_{i-1})$. Otherwise, we set $Q_i^F(x_i|x_1,\ldots,x_{i-1})=1$. Then, once $Q_i^F(x_i|x_1,\ldots,x_{i-1})$ is computed for all i, we can compute $Q^F(\mathbf{x})$ using $Q^F(\mathbf{x})=\prod_{i=1}^n Q_i^F(x_i|x_1,\ldots,x_{i-1})$.

[2] We can also use some restricted dynamic variable orderings and still maintain correctness of Theorem 1

Approximating $Q^F(\mathbf{x})$ Since computing $Q^F(\mathbf{x})$ requires $O(n)$ invocations of a complete SAT solver per sample, as n gets larger, *SampleSearch* is likely to be slow. Instead, we propose an approximation of $Q^F(\mathbf{x})$ denoted by $A^F(\mathbf{x})$ while still maintaining the essential statistical property of *asymptotic unbiasedness* (Rubinstein 1981).

Definition 2 (Asymptotic Unbiasedness)**.** $\widehat{\theta_N}$ is an asymptotically unbiased estimator of θ if $\lim_{N\to\infty}\mathbb{E}(\widehat{\theta_N}) = \theta$

Note that to compute $Q^F(\mathbf{x})$, we have to determine whether $(x_1,\ldots,x_{i-1},\overline{x_i})$ can be extended to a solution. While generating a sample $\mathbf{x}$, *SampleSearch* may have already determined that $(x_1,\ldots,x_{i-1},\overline{x_i})$ cannot be extended to a solution and we can use this information uncovered by *SampleSearch* to yield an approximation as follows. If *SampleSearch* has itself determined that $(x_1,\ldots,x_{i-1},\overline{x_i})$ cannot be extended to a solution while generating $\mathbf{x}$, we set $A_i^F(x_i|x_1,\ldots,x_{i-1}) = 1$, otherwise we set $A_i^F(x_i|x_1,\ldots,x_{i-1}) = Q_i(x_i|x_1,\ldots,x_{i-1})$. Finally, we compute $A^F(\mathbf{x}) = \prod_{i=1}^n A_i^F(x_i|x_1,\ldots,x_{i-1})$. However, this approximation does not guarantee asymptotic unbiasedness. We can take this intuitive idea a step further and guarantee asymptotic unbiasedness.

We can store (cache) each solution $(x_1,\ldots,x_n)$ and all partial assignments $(x_1,\ldots,\overline{x}_i)$ that were proved to be inconsistent during each independent execution of *SampleSearch*, which we refer to as search-traces. After executing *SampleSearch* N times, (i.e. once we have our required samples) we use the stored N traces to compute the approximation $A_N^F(\mathbf{x})$ for each sample as follows (the approximation is indexed by N to denote dependence on N). For each partial sample $(x_1,\ldots,x_i)$ if $(x_1,\ldots,\overline{x}_i)$ was found to be inconsistent during any of the N executions of *SampleSearch*, we set $A_{N_i}^F(x_i|x_1,\ldots,x_{i-1}) = 1$, otherwise we set it to $Q_i(x_i|x_1,\ldots,x_{i-1})$. Finally, we compute $A_N^F(\mathbf{x}) = \prod_{i=1}^n A_{N_i}^F(x_i|x_1,\ldots,x_{i-1})$.

It is clear that as N grows, more inconsistencies will be discovered by *SampleSearch* and as $N\to\infty$, all inconsistencies will be discovered making $A_N^F(\mathbf{x})$ equal to $Q^F(\mathbf{x})$. Consequently,

Proposition 2 (Asymptotically Unbiased Property)**.** $\lim_{N\to\infty} A_N^F(\mathbf{x}) = Q^F(\mathbf{x})$.

Example 1. Figure 1 shows the probability tree associated with a given distribution Q. Each arc from a parent node to the child node is labeled with the probability of generating the child node given the assignment on the path from the root node to the parent node. The probability tree is also the complete search tree for the formula shown. Let us assume that *SampleSearch* has generated three traces as shown in Figure 2. Note that in our example, we have 3 samples but only two distinct solutions (A=0,B=1,C=1) and (A=1,B=1,C=1). One can verify that $Q^F(\mathbf{x})$ of Traces 1, 2 and 3 is 0.8, 0.8 and 0.06 respectively. On the other hand, the approximation $A_3^F(\mathbf{x})$ of Traces 1, 2 and 3 is 0.8, 0.8 and 0.036 respectively.

Lower Bounding the Solution Counts

Definition 3 (Markov Inequality)**.** For any random variable $\mathbf{X}$ and $p > 1$, $\mathbf{Pr}(X > p\mathbb{E}[X]) \leq 1/p$

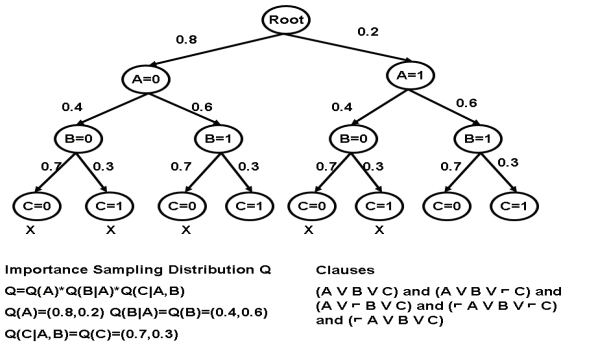

Figure 1: An example Probability (Search) Tree for the shown Formula and assuming an importance sampling distribution Q. Leaf nodes marked with X are not solutions.

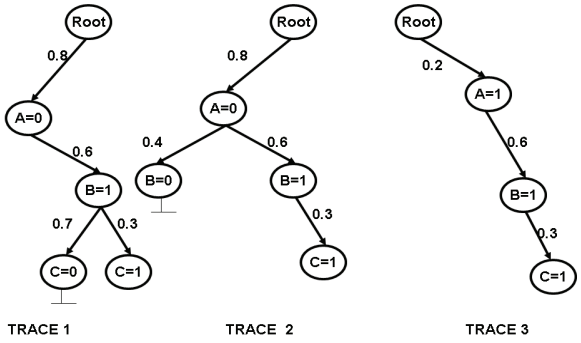

Figure 2: DPLL-Traces of SampleSearch. The grounded nodes were proved inconsistent.

(Gomes *et al.* 2007) show how a simple application of the Markov inequality can be used to obtain probabilistic lower bounds on the counting task. Using the same approach, we present a small modification of *SampleSearch*, *SampleSearch* $-$ *LB* for obtaining lower bounds on the exact solution counts (see Algorithm 2). *SampleSearch* $-$ *LB* generates k samples using *SampleSearch* and returns the minimum $\frac{1}{\alpha Q^F(\mathbf{x})}$ (minCount in Algorithm 2) over the k-samples.

Algorithm 2 *SampleSearch* $-$ *LB*$(F, Q, O, k, \alpha > 1)$

1: $minCount \leftarrow 2^n$
2: **for** $i = 1$ to k **do**
3: Generate a sample $\mathbf{x}^i$ using *SampleSearch*(F, Q, O)
4: **IF** $minCount > \frac{1}{Q^F(\mathbf{x}^i)}$ **THEN** $minCount = \frac{1}{Q^F(\mathbf{x}^i)}$
5: **end for**
6: Return $\frac{minCount}{\alpha}$

THEOREM **2** (Lower Bound)**.** *With probability of at least* $1 - 1/\alpha^k$*, SampleSearch* $-$ *LB returns a lower bound on the number of models of Formula F*

Proof. Consider an arbitrary sample $\mathbf{x}^i$. From Theorem

1 and Proposition 1, $\mathbb{E}(\frac{1}{Q^F(\mathbf{x}^i)}) = |\mathbf{S}|$. Therefore, by Markov inequality, we have $\mathbf{Pr}(\frac{1}{Q^F(\mathbf{x}^i)} > \alpha|\mathbf{S}|) < 1/\alpha$. Since, the generated k samples are independent, the probability $\mathbf{Pr}(min_{i=1}^{k} \frac{1}{Q^F(\mathbf{x}^i)} > \alpha|\mathbf{S}|) < 1/\alpha^k$ and therefore $\mathbf{Pr}(min_{i=1}^{k}(\frac{1}{\alpha Q^F(\mathbf{x}^i)}) < |\mathbf{S}|) > 1 - 1/\alpha^k$. □

Experimental Evaluation

Competing Techniques

SampleSearch takes as input a proposal distribution Q. The performance of importance sampling based algorithms is highly dependent on the proposal distribution (Cheng & Druzdzel 2000; Yuan & Druzdzel 2006). It was shown that computing the proposal distribution from the output of a generalized belief propagation scheme of Iterative Join graph propagation (IJGP) yields good empirical performance than other available choices (Gogate & Dechter 2005). Therefore, we use the output of IJGP to compute the initial proposal distribution Q. The complexity of IJGP is time and space exponential in a parameter i also called as i-bound. We tried i-bounds of 1, 2 and 3 and found that the results were not sensitive to the i-bound used in this range and therefore we report results for i-bound of 3. The preprocessing time for computing the proposal distribution using IJGP ($i = 3$) was negligible (< 2 seconds for the hardest instances).

As pointed out earlier, we can replace DPLL in Algorithm 1 with any SAT solver. We chose to use minisat as our SAT solver because currently it is the best performing SAT solver (Sorensson & Een 2005). Henceforth, we will refer to *minisat* based *SampleSearch* as *SampleMinisat*.

We experimented with three versions of SampleMinisat (a) *SampleMinisat-exact* in which the importance weights are computed using the exact backtrack-free distribution Q^F (b) *SampleMinisat-app* in which the importance weights are computed using the approximation A_N^F of Q^F and (c) *SampleMinisat-LB* for lower bounding (see Algorithm 2).

We compare *SampleMinisat-exact* and *SampleMinisat-app* with a WALKSAT-based approximate solution counting technique, *ApproxCount* (Wei & Selman 2005) while we compare the lower-bound returned by *SampleMinisat-LB* with a lower-bounding technique *SampleCount* recently presented in (Gomes *et al.* 2007).

Results

We conducted experiments on a 3.0 GHz Intel P-4 machine with 2GB memory running Linux. Table 1 summarizes our results. We tested our approach on the benchmark formulas used in (Gomes *et al.* 2007). These problems are from six domains: circuit synthesis, random k-cnf, Latin square, Langford, Ramsey and Schur's Lemma. An implementation of WALKSAT-based model counter *ApproxCount* (Wei & Selman 2005) is available on the first author's web-site while the implementation of *SampleCount* is not publicly available. Therefore, we use the results reported in (Gomes *et al.* 2007) which were performed on a faster CPU. We terminated each algorithm after 12 hrs if it did not terminate by itself (indicated by a Timeout in Table 1).

In all our experiments with *SampleMinisat-LB*, we set $k = 7$ and $\alpha = 2$ giving a correctness confidence of $1 - 1/2^7 \approx 99\%$ (see Theorem 2). The results reported in (Gomes *et al.* 2007) also use the 99% confidence level. In all our experiments for determining the average count using *SampleMinisat-app* and *SampleMinisat-exact*, we set the number of samples N to 2000. The *ApproxCount* implementation was run with default settings and the *pickeven* heuristic which was shown to perform better than other heuristics.

From Table 1 (columns 3 and 4), we see that *SampleMinisat-LB* scales well with problem size and provides good high-confidence lower-bounds close to the true counts. On most instances *SampleMinisat-LB* provides better lower bounds than *SampleCount*.

The performance of *SampleMinisat-exact* and *SampleMinisat-app* is substantially more stable than *ApproxCount* in that the error between the exact count (when it is known) and the approximate count is much larger for *ApproxCount* than both *SampleMinisat-exact* and *SampleMinisat-app* (see Table 1, columns 5, 6 and 7).

We notice that (a) the solution counts produced by *SampleMinisat-exact* and *SampleMinisat-app* are similar and (b) the time required by *SampleMinisat-app* is substantially lower than *SampleMinisat-exact* indicating that the improvement achieved using a potentially better but costly exact estimator is minor.

Note that all *SampleMinisat* implementations were not able to compute approximate counts (indicated by Timeout in Table 1) for *3bitadd_32* and *Ramsey-23-4-5* instances. These instances are beyond the reach of a backtracking (DPLL) solver like *SampleMinisat* even for finding a single solution within the 12hr time bound. On the other hand, both *SampleCount* and *ApproxCount* use *WALKSAT* for generating solutions which solves these instances quite easily.

Summary and Conclusion

We presented an approach that uses importance sampling to count the number of solutions of a SAT formula. However, a straight-forward application of importance sampling on a deterministic problem such as satisfiability may lead to very poor approximations because a large number of samples having zero weight are generated (rejection). To address the rejection problem, we developed a new scheme of *SampleSearch* which is a DPLL-based randomized backtracking procedure.

We characterize the sampling distribution of *SampleSearch* and develop two weighting schemes that can be used in conjunction with *SampleSearch* to estimate solution counts. We also introduced a minor modification to *SampleSearch* which can be used to lower-bound the number of solutions using the Markov inequality. We present promising empirical evidence showing that *SampleSearch* based counting schemes are competitive with state-of-the-art methods.

Acknowledgements

This work was supported in part by the NSF under award numbers IIS-0331707 and IIS-0412854. We would like to

Problems	Exact	SampleCount (Gomes *et al.* 2007)		SampleMinisat Lower Bound		SampleMinisat Exact		SampleMinisat Approximate		ApproxCount (Wei & Selman 2005)	
	Models	Models	Time	Models	Time	Models	Time	Models	Time	Models	Time
Circuit											
2bitcomp6	2.10E+29	$\geq$ 2.40E+28	29	$\geq$ **7.79E+28**	5	**2.08E+29**	345	1.15E+29	2	7.40E+29	32
3bitadd_32	-	$\geq$ **5.9E+1339**	1920	Timeout	43200	Timeout	43200	Timeout	43200	8.7E+1256	6705
Random											
wff-3-3.5	1.40E+14	$\geq$ 1.60E+13	240	$\geq$ **2.32E+13**	4	**1.45E+14**	145	1.59E+14	5	1.30E+15	15
wff-3.1.5	1.80E+21	$\geq$ 1.00E+20	240	$\geq$ **1.55E+20**	20	1.58E+21	128	**1.89E+21**	2	9.90E+21	13
wff-4.5.0	-	$\geq$ 8.00E+15	120	$\geq$ **2.30E+16**	28	1.09E+17	191	1.35E+17	3	7.00E+14	8
Latin-square											
ls8-norm	5.40E+11	$\geq$ 3.10E+10	1140	$\geq$ **1.03E+11**	55	2.22E+11	168	**4.84E+11**	41	3.00E+12	70
ls9-norm	3.80E+17	$\geq$ 1.40E+15	1920	$\geq$ **1.45E+16**	102	9.77E+16	212	**1.42E+17**	66	1.47E+18	48
ls10-norm	7.60E+24	$\geq$ 2.70E+21	2940	$\geq$ **1.41E+23**	179	**3.44E+24**	354	1.24E+25	91	1.00E+27	97
ls11-norm	5.40E+33	$\geq$ 1.20E+30	4140	$\geq$ **1.26E+31**	314	**3.38E+34**	527	1.24E+35	93	1.53E+37	104
ls12-norm	-	$\geq$ 6.90E+37	3000	$\geq$ **1.81E+38**	640	9.58E+38	1356	1.11E+40	131	1.67E+49	1034
ls13-norm	-	$\geq$ 3.00E+49	4020	$\geq$ **5.58E+51**	1232	1.82E+52	3200	4.36E+53	183	1.81E+63	1300
ls14-norm	-	$\geq$ 9.00E+60	2640	$\geq$ **6.07E+61**	2244	3.56E+62	9202	4.07E+62	242	1.57E+80	3000
ls15-norm	-	$\geq$ 1.10E+73	3360	$\geq$ **5.34E+79**	3903	1.50E+78	27829	2.67E+78	333	1.89E+99	7837
ls16-norm	-	$\geq$ 6.00E+85	4080	$\geq$ **3.29E+92**	3950	8.40E+92	18292	1.20E+93	453	1.15E+131	9638
Langford											
Langford-12-2	1.00E+05	$\geq$ 4.30E+03	1920	$\geq$ **1.10E+04**	340	**1.40E+05**	1003	2.12E+05	8	4.80E+05	109
Langford-15-2	3.00E+07	$\geq$ 1.00E+06	3600	$\geq$ **6.50E+06**	543	1.35E+07	1729	**1.42E+07**	25	8.60E+11	231
Langford-16-2	3.20E+08	$\geq$ 1.00E+06	3900	$\geq$ **2.50E+07**	501	**6.70E+08**	3092	2.50E+09	39	8.60E+11	929
Langford-19-2	2.10E+11	$\geq$ 3.30E+09	3720	$\geq$ **1.20E+11**	892	**5.80E+10**	5202	2.90E+12	248	1.80E+14	828
Langford-20-2	2.60E+12	$\geq$ 5.80E+09	3240	$\geq$ **6.90E+11**	982	**1.90E+13**	5001	4.50E+13	578	1.80E+16	2372
Langford-23-2	3.70E+15	$\geq$ 1.60E+11	5100	$\geq$ **1.50E+14**	1342	**9.90E+16**	3903	2.10E+17	750	9.90E+25	7562
Langford-24-2	-	$\geq$ 4.10E+13	4800	$\geq$ **9.80E+14**	1522	6.01E+16	7360	1.77E+18	650	1.80E+28	7383
Langford-27-2	-	$\geq$ 5.20E+14	6660	$\geq$ **1.50E+16**	1089	5.50E+19	20840	7.20E+19	1030	3.80E+32	18910
Langford-28-2	-	$\geq$ 4.00E+14	7020	$\geq$ **3.40E+16**	3622	8.60E+21	39260	3.70E+22	1239	1.10E+30	28200
Ramsey-20-4-5	-	$\geq$ 3.30E+35	210	$\geq$ **3.03E+36**	560	1.12E+37	1452	6.00E+39	50	2.38E+31	134
Ramsey-23-4-5	-	$\geq$ **1.40E+31**	3180	Timeout	43200	Timeout	43200	Timeout	43200	3.50E+14	24784
Schur-5-100	-	$\geq$ **1.30E+17**	1200	$\geq$ 1.75E+15	339	7.70E+15	3120	6.03E+16	104	7.80E+10	17759

Table 1: Results on benchmarks used in (Gomes *et al.* 2007). Timeout indicates that the method did not generate any answer within 43200s. Time is in seconds. A '-' in the exact column indicates that the solution count is not known. The best results for lower-bounding and approximate counting (when the exact count is known) are highlighted in each row.

thank Ashish Sabharwal for providing some benchmarks used in this paper.

References

Bayardo, R., and Pehoushek, J. 2000. Counting models using connected components. In *AAAI*, 157–162. AAAI Press / The MIT Press.

Cheng, J., and Druzdzel, M. J. 2000. Ais-bn: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *J. Artif. Intell. Res. (JAIR)* 13:155–188.

Davis, M.; Logemann, G.; and Loveland, D. 1962. A machine program for theorem proving. *Communications of the ACM* 5:394–397.

Dechter, R., and Rish, I. 2003. Mini-buckets: A general scheme for bounded inference. *J. ACM* 50(2):107–153.

Geweke, J. 1989. Bayesian inference in econometric models using monte carlo integration. *Econometrica* 57(6):1317–39.

Gogate, V., and Dechter, R. 2005. Approximate inference algorithms for hybrid bayesian networks with discrete constraints. *UAI-2005*.

Gogate, V., and Dechter, R. 2006. A new algorithm for sampling csp solutions uniformly at random. *CP*.

Gomes, C.; Hoffmann, J.; Sabharwal, A.; and Selman, B. 2007. From sampling to model counting. *IJCAI*.

Rubinstein, R. Y. 1981. *Simulation and the Monte Carlo Method*. New York, NY, USA: John Wiley & Sons, Inc.

Sorensson, N., and Een, N. 2005. Minisat v1.13-a sat solver with conflict-clause minimization. In *SAT 2005*.

Wei, W., and Selman, B. 2005. A new approach to model counting. In *SAT*.

Yuan, C., and Druzdzel, M. J. 2006. Importance sampling algorithms for Bayesian networks: Principles and performance. *Mathematical and Computer Modelling*.