Studies in Solution Sampling

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

Abstract

We introduce novel algorithms for generating random solutions from a uniform distribution over the solutions of a boolean satisfiability problem. Our algorithms operate in two phases. In the first phase, we use a recently introduced SampleSearch scheme to generate biased samples while in the second phase we correct the bias by using either Sampling/Importance Resampling or the Metropolis-Hastings method. Unlike state-of-the-art algorithms, our algorithms guarantee convergence in the limit. Our empirical results demonstrate the superior performance of our new algorithms over several competing schemes.

Introduction

In this paper we present novel algorithms for sampling solutions uniformly from hard combinatorial problems. Our schemes are quite general and can be applied to complex graphical models like Bayesian and Markov networks which have deterministic relationships (Richardson and Domingos, 2006; Fishelson and Geiger, 2003). However, in this paper, we focus on boolean satisfiability problems only. Our work is motivated by a wide application of solution sampling in fields such as verification and probabilistic reasoning as elaborated in earlier work (Dechter et al., 2002; Wei et al., 2004; Gomes et al., 2007).

The solution sampling task is closely related to the #P-Complete problem of counting the number of solutions of a satisfiability problem. In fact, it is easy to show that if one can sample solutions from a uniform distribution then one can exactly count the number of solutions. Conversely, one can also design an exact solution sampling algorithm from an exact counting algorithm.

This latter approach was exploited in (Dechter et al., 2002) where an exact bucket elimination counting algorithm was used to sample solutions uniformly. Since bucket elimination is time and space exponential in a graph parameter called the treewidth, it cannot be used when the treewidth is large. Therefore, in a subsequent paper (Gogate and Dechter, 2006), we proposed a search based sampling scheme called SampleSearch. SampleSearch is a randomized backtracking procedure whose value selection is guided by sampling from approximate (polynomial time) solution counters based on mini-bucket elimination or generalized belief propagation. We found that SampleSearch is competitive with then state-of-the-art solution sampler based on Walksat (Wei et al., 2004). In spite of good empirical performance, we showed in our previous work (Gogate and Dechter, 2007) that SampleSearch has a serious shortcoming in that it generates samples from the backtrack-free distribution which can sometimes be quite far from the uniform distribution over the solutions. A similar shortcoming is shared by all other approximate solution samplers that were introduced so far (Wei et al., 2004; Gomes et al., 2007).

The main contribution of this paper is in correcting this deficiency by augmenting SampleSearch with statistical techniques of Sampling/Importance Resampling (SIR) and the Metropolis-Hastings (MH) method yielding SampleSearch-SIR and SampleSearch-MH respectively. Our new techniques operate in two phases. In the first phase, they use SampleSearch to generate a set of solution samples and in the second phase they *thin down* the samples by accepting only a subset of good samples. By carefully selecting this acceptance criteria, we can prove that the samples generated by our new schemes converge in the limit to the required uniform distribution over the solutions. This convergence is important because, as implied by sampling theory (Rubinstein, 1981), it is expected to yield a decreasing sampling error (and thus more accurate estimates) as the number of samples (or time) is increased. In contrast, pure SampleSearch will converge to the wrong backtrack-free distribution as time increases thereby yielding a constant error. Therefore, in theory, our new techniques should always be preferred over SampleSearch.

Our empirical evaluation shows that our new schemes have better performance in terms of sampling error (accuracy as a function of time) than pure SampleSearch and SampleSAT (Wei et al., 2004) on a diverse set of benchmarks; demonstrating their practical value.

Preliminaries and Related Work

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 solution of F.

Definition 1 (The Solution Sampling task)**.** Let $\mathbf{S} = \{\mathbf{x} | \mathbf{x} \textit{ is a solution of } F\}$ be the set of solutions of formula F. Given a uniform probability distribution $\mathscr{P}(\mathbf{x} \in \mathbf{S}) = 1/|\mathbf{S}|$ over all solutions of F and an integer M, the solution sampling task is to generate M solutions such that each solution is generated with probability $1/|\mathbf{S}|$.

Previous work

An obvious way to solve the solution sampling task is to first generate (and store) all solutions and then generate M samples from the stored solutions such that each solution is sampled with equal probability. The problem with this approach is obvious; in most applications we resort to sampling because we cannot enumerate the entire search space.

A more reasonable approach formally presented as Algorithm 1 works as follows. We first express the uniform distribution $\mathscr{P}(\mathbf{x})$ over the solutions in a product factored form: $\mathscr{P}(\mathbf{x} = (x_1, \ldots, x_n)) = \prod_{i=1}^{n} P_i(x_i | x_1, \ldots, x_{i-1})$. Then, we use a standard Monte Carlo (MC) sampler (also called logic sampling (Pearl, 1988)) to sample along the ordering $O = \langle X_1, \ldots, X_n \rangle$ implied by the product form. Namely, at each step, given a partial assignment $(x_1, \ldots, x_{i-1})$ to the previous $i-1$ variables, we assign a value to variable X_i by sampling it from the distribution $P_i(X_i | x_1, \ldots, x_{i-1})$. Repeating this process for each variable in the sequence produces a sample $(x_1, \ldots, x_n)$.

Algorithm 1: Monte-Carlo Sampler $(F, O, \mathscr{P})$

Input: A Formula F, An Ordering $O = (x_1, \ldots, x_n)$ and the uniform distribution over the solutions $\mathscr{P}$
Output: M Solutions drawn from the uniform distribution over the solutions

```
for j = 1 to M do
    x = φ;
    for i = 1 to n do
        p = a random number in (0,1);
        If p < P_i(X_i = 0|x) Then x = x ∪ (X_i = 0) Else
        x = x ∪ (X_i = 1);
    end
    Output x;
end
```

The probability $P_i(X_i = x_i | x_1, \ldots, x_{i-1})$ can be obtained by computing the ratio between the number of solutions that the partial assignment $(x_1, \ldots, x_i)$ participates in and the number of solutions that $(x_1, \ldots, x_{i-1})$ participates in.

Example 1. Figure 1(a) shows a complete search tree for the given SAT problem. Each arc from a parent $X_{i-1} = x_{i-1}$to a child $X_i = x_i$ in the search tree is labeled with the ratio of the number of solutions below the child (i.e. the number of solutions that $(x_1, \ldots, x_i)$ participates in) and the number of solutions below the parent (i.e. the number of solutions that $(x_1, \ldots, x_{i-1})$ participates in) which corresponds to the probability $P_i(X_i = x_1 | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$. Given random numbers $\{0.2, 0.3, 0.6\}$, the solution highlighted in bold in Figure 1(a) will be generated by Algorithm 1.

(Dechter et al., 2002) used bucket elimination based solution counting scheme to compute $P_i(X_i | x_1, \ldots, x_{i-1})$ from a SAT formula. In this scheme, the bucket elimination algorithm is run just once as a pre-processing step so that $P_i(X_i | x_1, \ldots, x_{i-1})$ can be computed for any partial assignment $(x_1, \ldots, x_{i-1})$ to the previous $i-1$ variables by performing a constant time table look-up. However, the time and space complexity of this pre-processing scheme is exponential in a graph parameter called the treewidth and when the treewidth is large, bucket elimination is impractical.

To address the exponential blow up, (Dechter et al., 2002; Gogate and Dechter, 2006) propose to compute an approximation $Q_i(X_i | x_1, \ldots, x_{i-1})$ of $P_i(X_i | x_1, \ldots, x_{i-1})$ by using polynomial schemes like mini-bucket elimination (MBE) or generalized belief propagation (GBP). However, MBE and GBP approximations may be too weak permitting the generation of inconsistent tuples. Namely, a sample generated from Q may not be a solution of the formula F and would have to be rejected. In some cases (e.g. for instances in the phase transition) almost all samples generated will be non-solutions and will be rejected. To circumvent this *rejection problem*, we proposed the SampleSearch scheme (Gogate and Dechter, 2006) which we present next.

SampleSearch and its sampling distribution

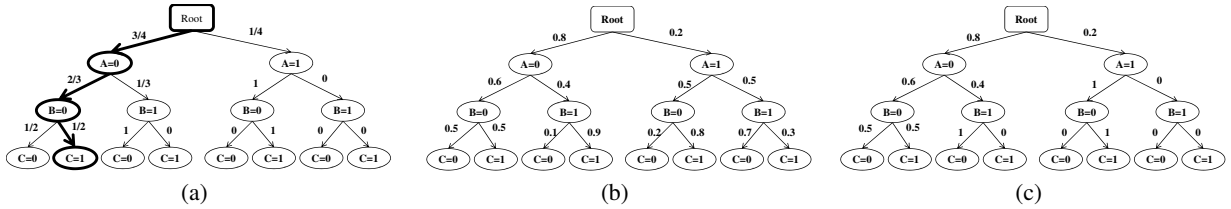
Algorithm 2: *SampleSearch* $SS(F, Q, O)$

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

```
UnitPropagate(F);
IF there is an empty clause in F THEN Return 0;
IF all variables are assigned a value THEN Return 1;
Select the earliest variable X_i in O not yet assigned a value;
p = Generate a random number between 0 and 1;
IF p < Q_i(X_i = 0|x_1, ..., x_{i-1}) THEN set x_i = 0 ELSE set
x_i = 1;
return SS((F ∧ x_i), Q, O) ∨ SS((F ∧ ¬x_i), Q, O)
```

SampleSearch (Gogate and Dechter, 2006) (see Algorithm 2) is a DPLL-style backtracking procedure whose value selection is guided by sampling from a specified distribution Q. It takes as input a formula F, an ordering $O = \langle X_1, \ldots, X_n \rangle$ of variables and a distribution $Q = \prod_{i=1}^{n} Q_i(x_i | x_1, \ldots, x_{i-1})$. Given a partial assignment $(x_1, ..., x_{i-1})$ already generated, a value $X_i = x_i$ for the next variable 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. Note that the unit-propagation step is optional. Any backtracking style search can be used, with any of the current advances (in that SampleSearch is a family of algorithms). If no empty clause is generated, then the algorithm proceeds with 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. The output of SampleSearch is a solution of F (assuming a solution exists). To generate M samples, the algorithm will be invoked M times.

We showed (Gogate and Dechter, 2006) that SampleSearch is competitive with another solution sampling approach called SampleSAT (Wei et al., 2004) which is based on the WalkSAT solver. Subsequently, we proved that



SAT Problem: $(A \vee \neg B \vee \neg C) \wedge (\neg A \vee B \vee C) \wedge (\neg A \vee \neg B \vee C) \wedge (\neg A \vee \neg B \vee \neg C)$

Figure 1: (a)Search tree for a SAT problem and its distribution for exact sampling, (b) A proposal distribution Q and (c) Backtrack-free distribution of Q.

the distribution of samples generated from SampleSearch converges to the backtrack-free distribution defined below (Gogate and Dechter, 2007).

Definition 2 (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 cnf formula F, the backtrack-free distribution is $Q^F(\mathbf{x}) = \prod_{i=1}^{n} Q_i^F(x_i|x_1,\ldots,x_{i-1})$ 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 be extended to a solution of F.
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.

Example 2. Figure 1(b) shows a distribution Q defined as a probability tree while Figure 1(c) displays the backtrack-free distribution Q^F of Q. Q^F is constructed from Q as follows. Given a parent and its two children one which participates in a solution and the other which does not, the edge-label of the child participating in a solution is changed to 1 while the edge-label of the other child which does not participate in a solution is changed to 0. If both children participate in a solution, the edge labels remain unchanged.

So, while SampleSearch samples from the backtrack-free distribution Q^F, it still does not solve the solution sampling problem because Q^F may still be quite far from the uniform distribution over the solutions. In the next two sections, we present the main contribution of this paper. Specifically we augment SampleSearch with ideas presented in the statistics literature – the Metropolis-Hastings method and the Sampling/Importance Resampling theory so that in the limit the sampling distribution of the resulting techniques is the target uniform distribution over the solutions.

SampleSearch-MH

The main idea in the Metropolis-Hastings (MH) simulation algorithm (Hastings, 1970) is to generate a Markov Chain whose limiting or stationary distribution is equal to the target distribution $\mathscr{P}$. A Markov chain consists of a sequence of states and a transition-rule $T(\mathbf{y}|\mathbf{x})$ for moving from state $\mathbf{x}$ to state $\mathbf{y}$. Given a transition function $T(\mathbf{y}|\mathbf{x})$, an acceptance function $0 \leq A(\mathbf{y}|\mathbf{x}) \leq 1$ and a current state $\mathbf{x}^t$, the Metropolis-Hastings algorithm works as follows:

- Draw $\mathbf{y}$ from $T(\mathbf{y}|\mathbf{x}^t)$.
- Draw p uniformly from $[0,1]$ and update

$$\mathbf{x}^{t+1} = \begin{cases} \mathbf{y} & \text{if } p \leq A(\mathbf{y}|\mathbf{x}^t) \\ \mathbf{x}^t & \text{otherwise} \end{cases}$$

(Hastings, 1970) proved that the stationary distribution of the above Markov Chain converges to $\mathscr{P}$ when it satisfies the *detailed balance* condition:

$$\mathscr{P}(\mathbf{x})T(\mathbf{y}|\mathbf{x})A(\mathbf{y}|\mathbf{x}) = \mathscr{P}(\mathbf{y})T(\mathbf{x}|\mathbf{y})A(\mathbf{x}|\mathbf{y})$$

Algorithm 3: *SampleSearch* $-$ *MH*(F,Q,O,M)

Input: A formula F, A distribution Q, An ordering O, Integer M
Output: M solutions

```
x^0 = SampleSearch(F,O,Q);
for t = 0 to M − 1 do
    y = SampleSearch(F,O,Q);
    Generate a random number p in the interval [0,1];
    If p ≤ Q^F(x^t)/(Q^F(x^t)+Q^F(y)) Then x^{t+1} = y Else x^{t+1} = x^t;
end
```

We can use the general MH principle for solution sampling in a straight-forward way as described by Algorithm 3. Here, we first generate a solution sample $\mathbf{y}$ using SampleSearch and then accept the solution with probability $\frac{Q^F(\mathbf{x})}{Q^F(\mathbf{x})+Q^F(\mathbf{y})}$. We can prove that:

Proposition 1. *The Markov Chain of SampleSearch-MH satisfies the detailed balance condition.*

Proof. Note that because we are supposed to generate samples from a uniform distribution over the solutions $\mathscr{P}(\mathbf{x}) = \mathscr{P}(\mathbf{y})$, the detailed balance condition reduces to:$T(\mathbf{y}|\mathbf{x})A(\mathbf{y}|\mathbf{x}) = T(\mathbf{x}|\mathbf{y})A(\mathbf{x}|\mathbf{y})$ Because each sample is generated independently from Q^F, we have:$T(\mathbf{y}|\mathbf{x}) = Q^F(\mathbf{y})$ *and* $T(\mathbf{x}|\mathbf{y}) = Q^F(\mathbf{x})$. Also, from Step 5 of SampleSearch-MH the acceptance probability is:$A(\mathbf{y}|\mathbf{x}) = \frac{Q^F(\mathbf{x})}{Q^F(\mathbf{x})+Q^F(\mathbf{y})}$ *and* $A(\mathbf{x}|\mathbf{y}) = \frac{Q^F(\mathbf{y})}{Q^F(\mathbf{x})+Q^F(\mathbf{y})}$ Consequently,

$$\begin{aligned} T(\mathbf{y}|\mathbf{x})A(\mathbf{y}|\mathbf{x}) &= Q^F(\mathbf{y})\frac{Q^F(\mathbf{x})}{Q^F(\mathbf{x})+Q^F(\mathbf{y})} \\ &= Q^F(\mathbf{x})\frac{Q^F(\mathbf{y})}{Q^F(\mathbf{x})+Q^F(\mathbf{y})} = T(\mathbf{x}|\mathbf{y})A(\mathbf{x}|\mathbf{y}) \end{aligned}$$

□

Integrating MH with SampleSearch opens the way for applying various improvements to MH proposed in the statistics literature. We describe next the integration of an improved MH algorithm with SampleSearch.

Improved SampleSearch-MH In SampleSearch-MH, we generate a state (or a sample) from the backtrack-free distribution and then a decision is made whether to accept the state based on an acceptance function. In Improved SampleSearch-MH (see Algorithm 4), we first *widen* the acceptance by generating multiple states instead of just one and then accept a good state from these multiple options. Such methods are referred to as *multiple trial MH* (Liu, Jun S. et al.,2000). Given a current sample $\mathbf{x}$, it generates k candidates $\mathbf{y}_1,\ldots,\mathbf{y}_k$ from the backtrack-free distribution Q^F in the usual SampleSearch style. It then selects a (single) sample $\mathbf{y}$ from the k candidates with probability proportional to $1/Q^F(\mathbf{y}_j)$. Sample $\mathbf{y}$ is then accepted according to the following acceptance function:

$$A(\mathbf{y}|\mathbf{x}) = \frac{\mathscr{W}}{\mathscr{W} - \frac{1}{Q^F(\mathbf{y})} + \frac{1}{Q^F(\mathbf{x})}} \quad where \ \mathscr{W} = \sum_{j=1}^{k} \frac{1}{Q^F(\mathbf{y}_j)} \quad (1)$$

Using results from (Liu, Jun S. et al.,2000), it is easy to prove that:

Proposition 2. *The Markov Chain generated by Improved-SampleSearch-MH satisfies the detailed balance condition.*

Algorithm 4: Improved-SampleSearch-MH (F,Q,O,M,k)

Input: A formula F, A distribution Q, An ordering O, Integer M and k
Output: M solutions

Generate $\mathbf{x}^0$ using SampleSearch(F,O,Q);
for $t = 0$ *to* $M-1$ **do**
  $\mathbf{Y} = \phi$;
  for $j = 1$ *to* k **do**
    $\mathbf{y}_j$ = SampleSearch(F,O,Q);
    $\mathbf{Y} = \mathbf{Y} \cup \mathbf{y}_j$;
  end
  Compute $\mathscr{W} = \sum_{j=1}^{k} \frac{1}{Q^F(\mathbf{y}_j)}$;
  Select $\mathbf{y}$ from the set $\mathbf{Y}$ by sampling each element $\mathbf{y}_j$ in $\mathbf{Y}$ with probability proportional to $\frac{1}{Q^F(\mathbf{y}_j)}$;
  Generate a random number p in the interval $[0,1]$;
  If $p \leq \frac{\mathscr{W}}{\mathscr{W} - \frac{1}{Q^F(\mathbf{y})} + \frac{1}{Q^F(\mathbf{x}^t)}}$ **Then** $\mathbf{x}^{t+1} = \mathbf{y}$ **Else** $\mathbf{x}^{t+1} = \mathbf{x}^t$;
end

Clearly, it follows that:

THEOREM **1.** *The stationary distribution of SampleSearch-MH and Improved-SampleSearch-MH is the uniform distribution over the solutions.*

Proof. Proof follows from proposition 1 and 2 and the Metropolis-Hastings theory (Hastings, 1970). □

SampleSearch-SIR

We now discuss our second algorithm which augments SampleSearch with Sampling/Importance Resampling (SIR) (Rubin, 1987) yielding the SampleSearch-SIR technique. Standard SIR (Rubin, 1987) aims at drawing random samples from a target distribution $\mathscr{P}(\mathbf{x})$ by using a given proposal distribution $Q(\mathbf{x})$ which satisfies $\mathscr{P}(\mathbf{x}) > 0 \Rightarrow Q(\mathbf{x}) > 0$. First, a set of independent and identically distributed random samples $\mathbf{A} = (\mathbf{x}^1,\ldots,\mathbf{x}^N)$ are drawn from a proposal distribution $Q(\mathbf{x})$. Second, a possibly smaller number of samples $\mathbf{B} = (\mathbf{y}^1,\ldots,\mathbf{y}^M)$ are drawn from $\mathbf{A}$ with or without replacement with sample probabilities which are proportional to the weights $w(\mathbf{x}^i) = \mathscr{P}(\mathbf{x}^i)/Q(\mathbf{x}^i)$ (this step is referred to as the *re-sampling step*). The samples from SIR will, as $N \to \infty$, consist of independent draws from $\mathscr{P}$.

We can easily augment SampleSearch with SIR as described in Algorithm 5. Here, a set $\mathbf{A} = (\mathbf{x}^1,\ldots,\mathbf{x}^N)$ of solution samples ($N > M$) is first generated from SampleSearch. Then, M samples are drawn from $\mathbf{A}$ with or without replacement with sample probabilities proportional to $1/Q^F(\mathbf{x}^i)$. Note that when sampling with replacement a solution may be generated multiple times in the final set of samples $\mathbf{B}$ while when sampling without replacement each solution will appear only once in the final set.

Algorithm 5: $SampleSearch-SIR(F,Q,O,N,M)$

Input: A formula F, A distribution Q, An ordering O, Integers M and N, $N > M$
Output: M solutions

1 Generate N i.i.d. samples $\mathbf{A} = \{\mathbf{x}^1,\ldots,\mathbf{x}^N\}$ by executing SampleSearch(F,O,Q) N times;
2 Compute importance weights $\{w^1 = \frac{1}{Q^F(\mathbf{x}^1)},\ldots,w^N = \frac{1}{Q^F(\mathbf{x}^N)}\}$ for each sample where Q^F is the backtrack-free distribution;
3 Normalize the importance weights using $\widetilde{w}^i = w^i / \sum_{j=1}^{N} w^j$;
4 **Re-sampling Step:** Generate M i.i.d. samples $\{\mathbf{y}^1,\ldots,\mathbf{y}^M\}$ from $\mathbf{A}$ by sampling each sample $\mathbf{x}^i$ with probability $\widetilde{w}^i$.;

THEOREM **2.** *As $N \to \infty$, the samples generated by SampleSearch − SIR consist of independent draws from the uniform distribution over the solutions of F.*

Proof. From SIR theory (Rubin, 1987), it follows that *SampleSearch − SIR* generates solutions from uniform distribution $\mathscr{P}$ if each sample is weighed as:

$$w^i \propto \frac{\mathscr{P}(\mathbf{x}^i)}{Q^F(\mathbf{x}^i)} \propto \frac{1}{Q^F(\mathbf{x}^i)}$$

Since the weight of solution $\mathbf{x}^i$, is $1/Q^F(\mathbf{x}^i)$ (Step 2 of *SampleSearch − SIR*), the proof follows. □

The integration of SampleSearch within the SIR framework allows us to use various improvements proposed in the statistics literature to the basic SIR framework. In the following subsection, we consider one such improvement known as the *Improved SIR framework*.

Improved SampleSearch-SIR Under certain restrictions (Skare et al., 2003) prove that the convergence of SIR is proportional to $O(1/N)$. To speed up this convergence to $O(1/N^2)$, they propose the Improved SIR framework. For our purposes, Improved SIR only changes the weights during resampling as follows.

In case of sampling with replacement the Improved SampleSearch-SIR weighs each sample as:

$$w(\mathbf{x}^i) \propto \frac{1}{S_{-i} * Q^F(\mathbf{x}^i)} \ where \ S_{-i} = \sum_{j=1}^{M} \frac{1}{Q^F(\mathbf{x}^j)} - \frac{1}{Q^F(\mathbf{x}^i)} \quad (2)$$

The first draw of Improved SampleSearch-SIR without replacement is specified by the weights in Equation 2. For the kth draw, $k > 1$, the distribution of w is modified to:

$$w(\mathbf{x}^k) \propto \frac{1}{Q^F(\mathbf{x}^k)(\sum_{j=1}^{N} \frac{1}{Q^F(\mathbf{x}^j)} - \sum_{j=1}^{k-1} \frac{1}{Q^F(\mathbf{x}^j)} - k\frac{1}{Q^F(\mathbf{x}^k)})}$$

Discussion

Theorems 1 and 2 are important because they guarantee that as the sample size increases, the samples drawn by SampleSearch-SIR and SampleSearch-MH would converge to the uniform distribution over the solutions. To our knowledge, the three state-of-the-art schemes in literature (a) SampleSearch, (b) SampleSAT (Wei et al., 2004) and (c) XorSample (Gomes et al., 2007) do not have such guarantees.

In particular, SampleSearch (Gogate and Dechter, 2007) converges to the wrong (backtrack-free) distribution while it is difficult to characterize the sampling distribution of SampleSAT. It is possible to make SampleSAT sample uniformly by setting the noise parameter appropriately but doing so was shown by (Wei et al., 2004) to compromise significantly the time required to generate a solution sample making SampleSAT impractical. The *XorSample* algorithm (Gomes et al., 2007) can be made to generate samples that can be guaranteed to be within any tiny constant factor of the uniform distribution by increasing the slack parameter α. However, for a fixed α, *XorSample* may not generate solutions from a uniform distribution (Gomes et al., 2007).

Experimental Evaluation

Implementation details

The performance of MH and SIR is highly dependent on the choice of Q. We compute Q from the output of Iterative Join graph propagation (IJGP), a generalized belief propagation scheme because it was shown to yield better empirical performance than other available choices (Gogate and Dechter, 2006). 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 and therefore we report results for i-bound of 1. Since we can use any systematic SAT solver as an underlying search procedure within SampleSearch, we chose to use the minisat SAT solver (Sorensson and Een, 2005).

We experimented with the following competing schemes (a) SampleSearch (b) SampleSearch-MH, (c) SampleSearch-SIR with replacement, (d) SampleSearch-SIR without replacement and (e) SampleSAT. Following previous work (Wei et al., 2004), we set the number of flips to one billion and the noise parameter to 50% in SampleSAT. We use a resampling ratio $M/N = 0.1 = 10\%$ (chosen arbitrarily) in all our experiments with the SIR scheme. All experiments are performed using improved versions of SampleSearch-SIR and SampleSearch-MH. In case of improved MH, we set the number of tries k to 10.

Results

We evaluate the quality of various algorithms by computing the average KL distance between the exact marginal distribution $P_e(x_i)$ and the approximate marginal distribution $P_a(x_i)$ of each variable ($KL = P_e(x_i)ln(P_e(x_i)/P_a(x_i))$). The exact marginal for each variable X_i can be computed as: $P_e(X_i = x_i) = |\mathbf{S}_{x_i}|/|\mathbf{S}|$ where $\mathbf{S}_{x_i}$ is the set of solutions that the assignment $X_i = x_i$ participates in and $\mathbf{S}$ is the set of all solutions. The number of solutions for the SAT problems were computed using Cachet (Sang et al., 2005). After running various sampling algorithms, we get a set of solution samples ϕ from which we compute the approximate marginal distribution as: $P_a(X_i = x_i) = \phi(x_i)/|\phi|$ where $\phi(x_i)$ is the number of solutions in the set ϕ in which $X_i = x_i$.

We experimented with four sets of benchmarks (see Table 1): (a) the grid pebbling problems, (b) the logistics planning instances, (c) circuit instances and (d) flat graph coloring instances. The first three benchmarks are available from Cachet (Sang et al., 2005) and the flat graph coloring benchmarks are available from Satlib (Hoos and Stützle, 2000). Note that we used SAT problems whose solutions can be counted in relatively small amount of time because to compute the KL distance we have to count solutions to $n+1$ SAT problems for each formula having n variables.

Table 1 summarizes the results of running each algorithm for exactly 1 hr on various benchmarks. The second and the third column report the number of variables and clauses respectively of each benchmark. For each sampling algorithm, in subsequent columns, we report the number of samples generated and the average KL distance. Note that lower the KL distance the more accurate the sampling algorithm is. We make the following observations from Table 1.

First, on most instances the SIR and MH methods are more accurate than pure SampleSearch and SampleSAT. Second, on most instances, SampleSearch generates more samples and is more accurate than SampleSAT. Third, on most instances, the number of samples generated by MH and SIR methods are less than SampleSearch. This is to be expected because both MH and SIR methods accept only 10% of the so-called good samples from SampleSearch's output and require computation of the backtrack-free distribution.

Finally, in Figures 2 and 3 we show how the KL distance of various algorithms changes with time on two instances. The remaining figures are presented in the extended version of this paper (Gogate and Dechter, 2008). We can see from Figures 2 and 3 that the KL distance converges rather slowly towards zero. Note that the convergence to zero described in Theorems 1 and 2 is only applicable in the limit and not in finite time. Namely, the sampling error of our schemes may not strictly decrease as time increases. On most instances, we found that the KL distance decreases very quickly to a particular value and stays there (or close to it) for a long time before decreasing again. At this point, we do not have a good scientific explanation for this behavior. Explaining and improving the finite time convergence properties of our new schemes is therefore a possible avenue for future work.

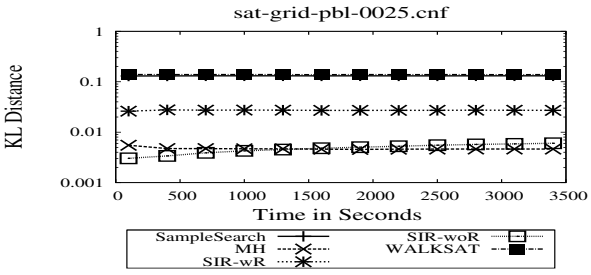


Figure 2: Time versus KL distance on Pebbling instance

Problem	#Var	#Cl	SampleSearch		SIR-wR		SIR-woR		MH		SampleSAT	
			#samples	KL	#samples	KL	#samples	KL	#samples	KL	#samples	KL
Pebbling												
grid-pbl-10	110	191	9.0E+07	0.08	7.2E+06	**0.002**	7.2E+06	0.010	1.1E+07	0.030	1.8E+08	0.110
grid-pbl-15	240	436	3.0E+07	0.12	3.7E+06	**0.017**	3.7E+06	0.050	2.7E+06	0.060	1.2E+08	0.127
grid-pbl-20	420	781	2.0E+07	0.10	3.3E+04	**0.008**	3.3E+04	0.011	3.5E+04	0.017	4.5E+07	0.153
grid-pbl-25	650	1226	1.2E+08	0.13	1.2E+07	0.027	1.2E+07	0.006	8.0E+06	**0.004**	1.2E+08	0.138
grid-pbl-30	930	1771	9.3E+05	0.15	1.0E+04	0.040	1.0E+04	0.007	1.2E+04	**0.007**	3.0E+07	0.154
Circuit												
2bitcomp_5	125	310	3.6E+08	0.03	4.5E+07	**0.003**	4.5E+07	0.006	1.2E+07	0.010	2.9E+08	0.033
2bitmax_6	252	766	3.6E+08	0.11	5.1E+07	**0.006**	5.1E+07	0.040	1.6E+07	0.053	2.5E+08	0.039
ssa7552-158	1363	3034	6.0E+07	0.06	6.9E+06	**0.006**	6.9E+06	0.020	3.5E+06	0.040	2.4E+07	0.130
ssa7552-159	1363	3032	6.0E+07	0.06	7.8E+06	**0.003**	7.8E+06	0.030	3.0E+07	0.040	3.2E+07	0.130
Logistics												
log-1	939	3785	7.2E+07	0.08	6.9E+06	**0.011**	6.9E+06	0.043	2.7E+06	0.044	4.5E+07	0.051
log-2	1337	24777	1.1E+07	0.12	2.0E+04	0.270	2.0E+04	0.107	3.5E+04	**0.101**	5.0E+04	0.203
log-3	1413	29487	1.0E+07	0.20	3.3E+04	**0.128**	3.3E+04	0.166	2.7E+04	0.166	3.5E+04	0.176
log-4	2303	20963	1.0E+07	0.21	2.5E+04	0.183	2.5E+04	0.165	3.2E+04	**0.156**	3.5E+04	0.290
log-5	2701	29534	8.0E+06	0.22	2.5E+04	0.270	2.5E+04	0.160	1.8E+04	**0.150**	3.3E+03	0.260
Coloring												
Flat-100	300	1117	5.1E+07	0.08	1.0E+05	**0.001**	1.0E+05	0.010	1.1E+05	0.032	1.4E+07	0.020
Flat-200	600	2237	5.1E+06	0.11	1.0E+04	**0.016**	1.0E+04	0.050	1.6E+04	0.050	4.9E+06	0.030

Table 1: KL distance and the number of samples after running each algorithm for 1hr

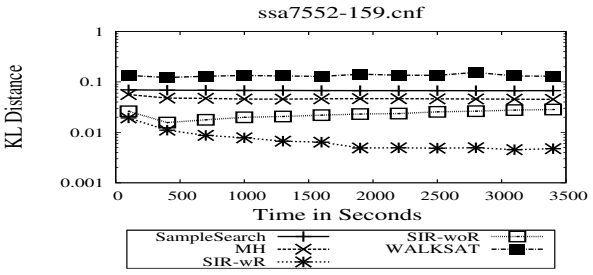


Figure 3: Time versus KL distance on circuit instance

Conclusion and Summary

The paper provides two new extensions to the SampleSearch scheme: SampleSearch-MH and SampleSearch-SIR for sampling solutions uniformly from a satisfiability problem. The origin for this task is the use of satisfiability based methods in fields such as verification and probabilistic reasoning. Our new schemes are guaranteed to (robustly) converge to the uniform distribution over the solutions as the sample size increases. Our empirical evaluation demonstrates substantial improved performance over earlier proposals for solution sampling.

Acknowledgements

This work was supported in part by the NSF under award numbers IIS-0331707, IIS-0412854 and IIS-0713118.

References

Gogate, V. and Dechter, R. (2008). Studies in solution sampling. Technical Report, University of California, Irvine, CA, USA.

Dechter, R., Kask, K., Bin, E., and Emek, R. (2002). Generating random solutions for constraint satisfaction problems. In *AAAI*, pages 15–21.

Fishelson, M. and Geiger, D. (2003). Optimizing exact genetic linkage computations. In *RECOMB 2003*.

Gogate, V. and Dechter, R. (2006). A new algorithm for sampling csp solutions uniformly at random. *CP*, pages 711–715.

Gogate, V. and Dechter, R. (2007). Approximate counting by sampling the backtrack-free search space. In *AAAI*, pages 198–203.

Gomes, C. P., Sabharwal, A., and Selman, B. (2007). Near-uniform sampling of combinatorial spaces using xor constraints. In *NIPS*. pages 481–488.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Hoos, H. H. and Stützle, T. (2000). SATLIB: An Online Resource for Research on SAT. pages 283–292.

Liu, Jun S., Liang, Faming, and Wong, Wing Hung (2000). The multiple-try method. *Journal of the American Statistical Association*, (449):121–134, .

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Rubin, D. B. (1987). The calculation of posterior distributions by data augmentation. *Jornal of the American Statistical Association*, 82:543–546.

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

Sang, T., Beame, P., and Kautz, H. A. (2005). Heuristics for fast exact model counting. In *SAT*, pages 226–240.

Skare, O., Bolviken, E., and Holden, L. (2003). Improved sampling-importance resampling and reduced bias importance sampling. *Scandinavian Journal of Statistics*, 30(4): 719–737.

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

Wei, W., Erenrich, J., and Selman, B. (2004). Towards efficient sampling: Exploiting random walk strategies. In *AAAI*, pages 670–676.