Approximate Solution Sampling (and Counting) on AND/OR spaces

Vibhav Gogate and Rina Dechter

Donald Bren School of Computer Science
University of California, Irvine, CA 92697, USA
{vgogate,dechter}@ics.uci.edu

Abstract. In this paper, we describe a new algorithm for sampling solutions from a uniform distribution over the solutions of a constraint network. Our new algorithm improves upon the Sampling/Importance Resampling (SIR) component of our previous scheme of SampleSearch-SIR by taking advantage of the decomposition implied by the network's *AND/OR search space*. We also describe how our new scheme can approximately count and lower bound the number of solutions of a constraint network. We demonstrate both theoretically and empirically that our new algorithm yields far better performance than competing approaches.

1 Introduction

In this paper, we present a new Sampling/Importance Resampling (SIR) algorithm that exploits AND/OR search spaces for graphical models [1]. Although, our algorithm is quite general, in this paper, we focus on using it for sampling solutions from a constraint network. Our main contributions are: (a) We show that SIR can be understood as a method that learns a probability distribution on an OR tree. (b) We generalize SIR to AND/OR spaces yielding AO-SIR which learns a probability distribution on an AND/OR tree (or graph). (c) We show theoretically and by an experimental evaluation on satisfiability benchmarks that AO-SIR is far more accurate than SIR. (d) We derive a new unbiased estimator from the distribution learnt over the AND/OR tree which can be used to approximately count and lower bound the number of solutions, improving over our previous solution counter presented in [4].

2 Preliminaries and Previous Work

Definition 1 (constraint network, counting and sampling). *A constraint network (CN) is defined by a 3-tuple, $\mathscr{R} = \langle \mathbf{X}, \mathbf{D}, \mathbf{C} \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of variables associated with a set of discrete-valued domains, $\mathbf{D} = \{\mathbf{D_1}, \ldots, \mathbf{D_n}\}$, and $\mathbf{C} = \{C_1, \ldots, C_r\}$ is a set of constraints. Each constraint C_i is a relation $\mathbf{R_{S_i}}$ defined on a subset of variables $\mathbf{S_i} \subseteq \mathbf{X}$. The relation denotes all compatible tuples of the cartesian product of the domains of $\mathbf{S_i}$. A solution is an assignment of values to all variables $\mathbf{x} = (X_1 = x_1, \ldots, X_n = x_n)$, $x_i \in \mathbf{D}_i$, such that $\mathbf{x}$ belongs to the natural join of all constraints i.e. $\mathbf{x} \in \mathbf{R_{S_1}} \bowtie \ldots \bowtie \mathbf{R_{S_r}}$. The* **solution counting problem** *#csp is the problem of counting the number of solutions. The* **solution sampling problem** *§csp is the problem of sampling solutions from a uniform distribution over the solutions.*

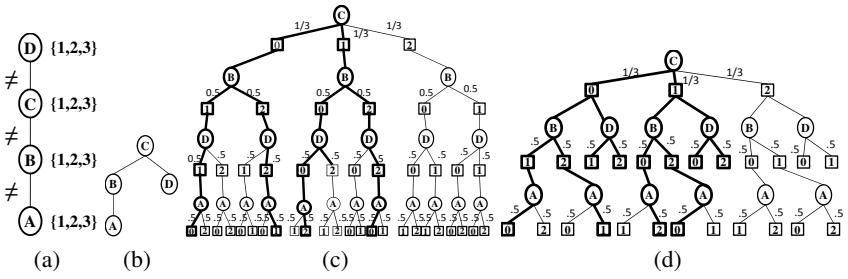

Fig. 1. (a) A 3-coloring problem, (b) Pseudo-tree (c) OR tree (d) AND/OR tree

2.1 AND/OR search spaces for graphical models

Given a constraint network, its AND/OR search space is guided by the pseudo-tree defined below (for more information see [1]).

Definition 2 (Pseudo Tree). *Given a constraint graph $G = (V, E)$, a pseudo-tree $T = (V, E')$ is a directed rooted tree in which any arc not included in E' is a back-arc.*

Definition 3 (AND/OR tree). *Given a constraint network $\mathscr{R} = \langle \mathbf{X}, \mathbf{D}, \mathbf{C} \rangle$ and a pseudo tree T, the AND/OR search tree S_{AOT}, has alternating levels of AND and OR nodes. The root of S_{AOT} is an OR node labeled by the root of T. The children of an OR node X_i are AND nodes labeled with assignment $X_i = x_i$ that are consistent with the assignment $(X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$ along the path from the root. The children of an AND node $X_i = x_i$ are OR nodes labeled with the children of variable X_i in T. A* **solution subtree** *of S_{AOT} contains the root node. For every OR node it contains one of its children and for each of its AND nodes it contains all its children. An* **OR tree** *is an AND/OR tree whose pseudo-tree is a chain.*

Example 1. Figure 1(c) and 1(d) show a complete OR tree and an AND/OR tree (guided by the pseudo-tree in Figure 1(b)) respectively for the 3-coloring problem in Figure 1(a).

2.2 Exact Solution to the §*csp* problem

Dechter et al. [2] proposed the following scheme to exactly solve the §*csp* problem. We first express the uniform distribution $\mathscr{P}(\mathbf{x})$ in a product factored form: $\mathscr{P}(\mathbf{x} = (x_1, \ldots, x_n)) = \prod_{i=1}^{n} P_i(x_i | x_1, \ldots, x_{i-1})$. The probability $P_i(X_i = x_i | x_1, \ldots, x_{i-1})$ is equal to the ratio between the number of solutions that $(x_1, \ldots, x_i)$ participates in and the number of solutions that $(x_1, \ldots, x_{i-1})$ participates in. Second, we generate multiple samples by repeating the following process: for *i*= 1 to n, 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 $P_i(X_i | x_1, \ldots, x_{i-1})$. All algorithms described in this paper are devoted to finding an approximation to $P_i(X_i | x_1, \ldots, x_{i-1})$ at each branch of the search tree.

Example 2. The labeled OR and AND/OR tree of Figure 1(c) and 1(d) respectively depict the uniform distribution over the solutions expressed in a product factored form.

2.3 Sampling Importance Resampling to solve the §*csp* problem Approximately

Because constructing $\mathscr{P}(\mathbf{x})$ can be quite hard [2], in [6] we proposed to use Sampling Importance Resampling (SIR) [8] in conjunction with the SampleSearch scheme [3] to approximate it. This scheme operates as follows. First, given a proposal distribution Q, it uses SampleSearch to draw random solution samples $\mathbf{A} = (\mathbf{x}^1, \ldots, \mathbf{x}^N)$ from the backtrack-free distribution Q^F of Q. Second, a possibly smaller number of samples $\mathbf{B} = (\mathbf{y}^1, \ldots, \mathbf{y}^M)$ are drawn from $\mathbf{A}$ with sample probabilities, proportional to the weights $w(\mathbf{x}^i) = 1/Q(\mathbf{x}^i)$ (this step is referred to as the *re-sampling step*). For $N = 1$, the distribution of solutions is same as Q^F. For a finite N, the distribution of solutions is somewhere between Q^F and $\mathscr{P}$ improving as N increases and equals $\mathscr{P}$ as $N \rightarrow \infty$.

3 Sampling Importance Resampling on AND/OR search spaces

We first describe the main intuition involved in defining a new SIR scheme called AO-SIR which operates on the AND/OR search space in the following example.

Example 3. The bold edges and nodes in Figure 1 (c) and (d) show four solution samples arranged on an OR and AND/OR tree respectively. Note that SIR and AO-SIR operate on the OR and AND/OR tree respectively. One can verify that the 4 solution samples correspond to 8 solution samples (solution sub-trees) on the AND/OR tree. Thus, the AND/OR representation yields a larger set of virtual samples. It includes for example the assignment $(C = 0, B = 2, D = 1, A = 0)$ which is not represented in the OR tree. From SIR theory [8], we know that the accuracy of SIR increases with the number of samples and therefore we expect AO-SIR to be more accurate than SIR.

In AO-SIR, the first step of using SampleSearch to generate solution samples remains the same. What changes is the way in which we (a) store samples and (b) define the distribution over the initial set of samples for resampling. We explain each of these modifications below. We first define the notion of an AND/OR sample tree (or graph) which can be used to store the initial set of samples.

Definition 4 (Arc Labeled AND/OR sample tree and graph). *Given (1) a constraint network $\mathscr{R} = \langle \mathbf{X}, \mathbf{D}, \mathbf{C} \rangle$, (2) a pseudo-tree $T(V,E)$, (3) the backtrack-free distribution $Q^F = \prod_{i=1}^{n} Q^F(X_i|\mathbf{Anc}(X_i))$ such that $\mathbf{Anc}(X_i)$ is a subset of all ancestors of X_i in T, (4) a sequence of samples $\mathbf{S}$ (assignments) generated from Q^F,* **an arc-labeled AND/OR sample tree** *S_{AOT} is a complete AND/OR tree (see definition 3) from which all assignments not in $\mathbf{S}$ are removed.The arc-label from an OR node X_i to an AND node $X_i = x_i$ in S_{AOT} is a pair $\langle w, \# \rangle$ where: (a) $w = \frac{1}{Q^F(X_i = x_i|\mathbf{anc}(X_i))}$ is called the weight of the arc. $\mathbf{anc}(X_i)$ is the assignment of values to all variables in $\mathbf{Anc}(X_i)$ and (b) #: the frequency of the arc is the number of times $(X_i = x_i, \mathbf{anc}(X_i))$ is seen in $\mathbf{S}$.*

As noted earlier, all approximate algorithms for solving the §*csp* problem can be thought of as approximating the probability labels on the arc of an AND/OR tree. Because, the probability labels are just ratios of solution counts, we define the notion of value of a node which can be semantically understood as providing an unbiased estimate of the solution counts of the subtree rooted at the node.

Algorithm 1 $AO-SIR(\mathscr{R}, Q^F, N, M)$

1: Generate N i.i.d. samples $\mathbf{A} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ from Q^F using SampleSearch.
2: Store the N solution samples on an AND/OR sample tree S_{AOT} or a graph and label it using definition 4.
3: **FOR** all leaf nodes i of S_{AOT} do
4: **IF And-node** v(i)= 1 **ELSE** v(i)=0
5: **FOR** every node n from leaves to the root do
6: Let $C(n)$ denote the child nodes of node n
7: **IF** $n = \langle X, x \rangle$ is a AND node, then $v(n) = \prod_{n' \in C(n)} v(n')$
8: **ELSE** if $n = X$ is a OR node then $v(n) = \frac{\sum_{n' \in C(n)} (\#(n,n') w(n,n') v(n'))}{\sum_{n' \in C(n)} \#(n,n')}$.
9: Update the weights by updating the arc labels as: $p(n,n') = \frac{v(n') w(n,n') \#(n,n')}{\sum_{n'' \in \mathbf{C(n)}} (\#(n,n'') w(n,n'') v(n''))}$
10: Return M solution samples generated at random from S_{AOT}

Definition 5 (Value of a node). *The value of a node in a arc-labeled AND/OR sample tree (see Definition 4) is defined recursively as follows. The value of leaf AND nodes is "1" and the value of leaf OR nodes is "0". Let* $\mathbf{C(n)}$ *denote the child nodes of n and* $v(n)$ *denotes the value of node n. If n is an AND node then:* $v(n) = \prod_{n' \in \mathbf{C(n)}} v(n')$ *and if n is a OR node then* $v(n) = \frac{\sum_{n' \in \mathbf{C(n)}} (\#(n,n') w(n,n') v(n'))}{\sum_{n' \in \mathbf{C(n)}} \#(n,n')}$.

Lemma 1. *The value of a node n is an unbiased estimate of the number of solutions of the subtree rooted at n.*

We now have the required definitions to formally present algorithm AO-SIR (see Algorithm 1). Here, we first generate samples in the usual way from Q^F. We then store these samples on an arc labeled AND/OR sample tree and compute the value of each node (Steps 3-8). The AND/OR sample tree is then converted to an AND/OR sample probability tree by normalizing the values at each OR node (Step 9). Finally, the required M solution samples are generated from the AND/OR sample probability tree. We can prove that:

Theorem 1. *As* $N \rightarrow \infty$*, the solutions generated by AO-SIR will consist of independent draws from the uniform distribution over the solutions.*

Finally, we summarize the relationship between AO-SIR and SIR in Theorem 2:

Theorem 2. *When the pseudo-tree is a chain, the solution samples output by AO-SIR will have the same distribution as those output by SIR. Asymptotically, if the pseudo-tree is not a chain then AO-SIR has lower sampling error than SIR.*

3.1 Approximate Counting on AND/OR search spaces

From Lemma 1, it is easy to see that:

Proposition 1. *The value of the root node of the AND/OR sample tree is an unbiased estimate of the number of solutions.*

We can utilize this unbiased estimate obtained from the AND/OR sample tree to obtain a lower bound on the solution counts [7, 4] in a straight forward way. The main virtue of of using the AND/OR space estimator over our previous scheme [4] is that the former may have lower variance (and therefore likely to have better accuracy) than the latter (for more details, see [5]).

Table 1. Results for Solution Sampling

Problem	#Var	#Cl	SampleSearch	SampleSearch-SIR	AO-SIR	SampleSat
			KL	KL	KL	KL
Pebbling						
grid-pbl-25	650	1226	0.13	0.027	**0.00600**	0.138
grid-pbl-30	930	1771	0.15	0.040	**0.00170**	0.154
Circuit						
2bitcomp_5	125	310	0.03	0.003	**0.00100**	0.033
2bitmax_6	252	766	0.11	0.006	**0.00100**	0.039
Coloring						
Flat-100	300	1117	0.08	**0.001**	0.00190	0.020
Flat-200	600	2237	0.11	0.016	**0.00800**	0.030

4 Experimental Results

For lack of space, we present only a part of our empirical results for solution sampling. Detailed experimental results for both solution sampling and counting are presented in the extended version of this paper [5]. For solution sampling, we experimented with the following schemes (a) SampleSearch [3] (b) SampleSearch-SIR [6] (c) SampleSat [9] and (d) AO-SIR. 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. The remaining columns report the KL distance between the exact and the approximate distribution for various competing schemes. Note that lower the KL distance the more accurate the sampling algorithm is. We can see that AO-SIR is more accurate than SampleSearch-SIR on most benchmarks. Also the SIR-type methods are more accurate than pure SampleSearch and SampleSat.

Acknowledgements This work was supported in part by the NSF under award numbers IIS-0331707, IIS-0412854 and IIS-0713118 and the NIH grant R01-HG004175-02.

References

1. R. Dechter and R. Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106, 2007.
2. Rina Dechter, Kalev Kask, Eyal Bin, and Roy Emek. Generating random solutions for constraint satisfaction problems. In *AAAI*, pages 15–21, 2002.
3. Vibhav Gogate and Rina Dechter. A new algorithm for sampling csp solutions uniformly at random. *CP*, 2006.
4. Vibhav Gogate and Rina Dechter. Approximate counting by sampling the backtrack-free search space. In *AAAI*, pages 198–203, 2007.
5. Vibhav Gogate and Rina Dechter. Approximate solution sampling (and counting) on and/or spaces. *Technical Report, University of California, Irvine*, 2008.
6. Vibhav Gogate and Rina Dechter. Studies in solution sampling. *AAAI*, 2008.
7. Carla Gomes, Jeorg Hoffmann, Ashish Sabharwal, and Bart Selman. From sampling to model counting. *IJCAI*, 2007.
8. Donald B. Rubin. The calculation of posterior distributions by data augmentation. *Jornal of the American Statistical Association*, 82, 1987.
9. Wei Wei, Jordan Erenrich, and Bart Selman. Towards efficient sampling: Exploiting random walk strategies. In *AAAI*, 2004.