Bayes Networks for Estimating the Number of Solutions to a CSP

Amnon Meisels, Solomon Eyal Shimony, and **Gadi Solotorevsky**

Mathematics and Computer Science Department, Ben-Gurion University
P.O. Box 653
84105 Be'er Sheva, ISRAEL
am@cs.bgu.ac.il, shimony@cs.bgu.ac.il, gadi@cs.bgu.ac.il

Abstract

The problem of counting the number of solutions to a constraint satisfaction problem (CSP) is rephrased in terms of probability updating in Bayes networks. Approximating the probabilities in Bayes networks is a problem which has been studied for a while, and may well provide a good approximation to counting the number of solutions. We use a simple approximation based on independence, and show that it is correct for tree-structured CSPs. For other CSPs, it is a less optimistic approximation than those suggested in prior work, and experiments show that it is more accurate on the average. We present empirical evidence that our approximation is a useful search heuristic for finding a single solution to a CSP.

Introduction

Estimating the number of solutions to a CSP is crucial, as it has been conjectured that problems are especially hard when they have a small number of solutions (Cheeseman, Kanefsky, & Taylor 1991). For certain binary constraint networks it has been shown empirically that their difficulty changes drastically in a "phase transition" region, which occurs when the number of solutions approaches 1 (Smith 1994). Each choice point in a CSP search algorithm creates a number of smaller CSP problems. It thus makes sense to choose the branch where the number of solutions in the reduced CSP is largest. Branching points in a search on a constraint network correspond to operations like *select next variable* or *select next value*, to assign to the selected variable. Thus, assessing the expected number of solutions at branching points leads naturally to ordering heuristics for values (cf. (Dechter & Pearl 1988)). Exact evaluation of the number of solutions is hard, but approximation may be possible.

We suggest that a CSP be converted to an equivalent Bayes network (Pearl 1988) such that the number of solutions is proportional to marginal probabilities in the network. Finding marginal probabilities is NP-hard (Cooper 1990; Dagum & Luby 1993), but there are numerous approximation algorithms for the problem (Dagum & Chavez 1993; Santos Jr., Shimony, & Williams 1996), which could result in a useful approximation to the number of solutions. The number of solutions can also be used as search heuristic, especially for value ordering, but for other types of heuristics on constraint networks as well.

Constraint satisfaction problems (CSPs), finite stochastic processes, and the Bayes networks we use below, all have a finite set of variables, each with a finite domain (cf. (Dechter & Pearl 1988; Tsang 1993; Pearl 1988)). We will thus take the liberty of abusing the notation, and use a common notation for all of them. Let $X_i, i = 1, ..., n$ be the variables, and D_{X_i} be the domain of CSP variable X_i. Random variables will have domain $D'_{X_i} = D_{X_i} \cup \{\perp\}$, with $\perp$ a unique domain value, standing for "inconsistent". Let $N = \prod_{i=1}^{n} \mid D_{X_i} \mid$. In a directed graph (or Bayes network), let $\pi(X_i)$ denote the predecessors of X_i. (We use the term "predecessors" to denote just "immediate" predecessors, and the term "ancestors" to denote the transitive closure of "predecessors".) An assignment to a set of variables is denoted by a set of $(variable, value)$ pairs, or a set of $variable = value$ instantiations. If $\mathcal{A}$ is an assignment, we denote its restriction to some set of variables S (or a single variable S) by $\mathcal{A}_S$. An assignment is complete if it assigns a value to all the variables.

An approximation to the number of solutions for uniform random CSPs appeared first in (Haralick & Elliott 1980), and was close to the exact number for this artificial family of problems. Using (different) such approximations as search heuristics was first suggested in (Dechter & Pearl 1988). The latter idea was to remove the "loosest" constraints so as to get a spanning-tree (binary) CSP, and use the number of solutions in the modified (relaxed) problem as the guide for value ordering. The number of solutions in the spanning tree was computed recursively as follows:

$$C(X, x) = \prod_{Y \in \pi'(X)} \sum_{y} C(Y, y) I((X, x), (Y, y)) \quad (1)$$

Where $C(variable, value)$ is the number of solutions where $variable$ has value $value$, in the sub-tree that consists of $variable$ and its ancestors in the spanning tree. $\pi'(X)$ are the *predecessors* of X in the tree. $I((Y, y), (X, x))$ is 1 just when the assignment $\{(Y, y), (X, x)\}$ is consistent with the CSP constraint between X and Y, and 0 otherwise. Without loss of generality, let X_1 be the last variable for which equation 1 is computed. The sum of $C(X_1, x)$ with x ranging over D_{X_1} is the number of solutions for the tree-structured CSP.

The present research is centered on approximating the number of solutions to a constraint network, without relaxing the network to form a spanning tree. Rather than attempting to find a relaxation of the problem, we achieve an approximation by assuming independence of the partial solutions, in a formal statistical sense defined over a random process. We show that our estimate reduces to that of the spanning-tree method (Dechter & Pearl 1988) for trees, but is always (not strictly) less optimistic. Empirical evidence further supports the feasibility of our approach.

CSPs as Bayes Networks

Let $G = (V, E)$ be the directed acyclic graph defined as follows: $V = \{X_1, ..., X_n\}$, and E has an edge (X_j, X_i) just when $i < j$ and there is a CSP constraint constraining X_j, X_i (Possibly a non-binary constraint, with other variables as well). Note that the direction of edges is opposite to the common practice in CSP (cf. (Dechter & Pearl 1988; Tsang 1993)) and is the default direction of edges in a Bayes net (see Figure 1). In other words, for every edge (X_j, X_i) and $i < j$, X_j is a predecessor of X_i. Without loss of generality (for a constant variable ordering), let $\{X_1, ..., X_n\}$ be the order of instantiation for a CSP search for a solution (cf. (Dechter & Pearl 1988)). Additionally, assume that every vertex has successors, except for X_1, the sink node (otherwise, add edges (X_i, X_1) for every $X_i \neq X_1$ with no successor, corresponding to vacuous constraints, permitting any pair of values).

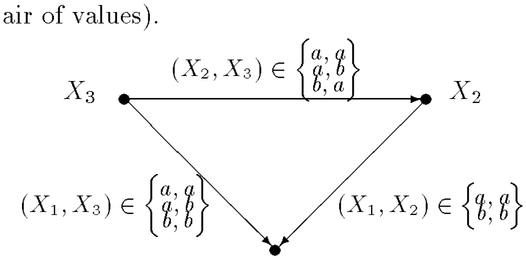

Figure 1: Constraint Graph for a 3-Node CSP

Define the following finite stochastic process (S):

1. For i from n **down to** 1 do
   (a) Randomly select $v_i \in D_{X_i}$ with uniform distribution.
   (b) If assignment $X_i = v_i$ is inconsistent with values assigned to predecessors of X_i,[1] let $X_i = \perp$.

[1] An assignment is consistent if it does not violate a CSP constraint, and does not assign the value $\perp$ to any variable. Thus, $\perp$ is defined to be inconsistent with *any* other values.

Denote by $P(e)$ the probability of event e in the above stochastic process. Probability equations with unassigned variables (e.g. $P(X_2) = P(X_2 | \{X_3 = a\})$) henceforth denote that the equation holds for all possible assignments to the variables (in this case, all possible assignments to X_2).

Theorem 1 *The number of different non-solutions is* $P(X_1 = \perp)N$.

Proof outline: Clearly, each instantiation as generated in all the steps **1a** is equally likely (probability $\frac{1}{N}$). Whenever an assignment is modified, i.e. when step **1b** applies, the instantiation is inconsistent (i.e. not a solution). If, for any i, an instantiation is modified to $X_i = \perp$, then by construction we also get $X_1 = \perp$, and thus the latter happens just when process S generated a non-solution. The theorem follows immediately. □

Theorem 2 $C(\mathcal{A})$*, the number of solutions consistent with a certain partial assignment* $\mathcal{A}$*, is given by:* $P(\mathcal{A} \wedge X_1 \neq \perp)N$.

Proof outline: By the same reasoning as above, except that we need to take the union of the events consistent with $\mathcal{A}$, but just those which are solutions (i.e. $X_1 \neq \perp$). □

The stochastic process can be modeled by a Bayes network B as follows. In order to define a Bayes network, one must define its topology (a directed acyclic graph), and for each node (variable) X_i, a conditional probability $P(X_i \mid \pi(X_i))$ for each possible assignment to X_i and its predecessors. These conditional probabilities define the joint probability of any complete assignment $\mathcal{A}$ (Pearl 1988):

$$P(\mathcal{A}) = \prod_{1 \leq i \leq n} P(\mathcal{A}_{X_i} \mid \mathcal{A}_{\pi(X_i)})$$

For process S, the topology of the network is the graph G defined above. The conditional probabilities for X_i are given below, with $v \in D_{X_i}$, and $\mathcal{A}$ denoting an arbitrary assignment to the predecessors of X_i.

$$P(X_i = v \mid \mathcal{A}) = \begin{cases} \frac{1}{|D_{X_i}|} & \mathcal{A} \cup \{X_i = v\} \text{ is consistent} \\ 0 & \text{otherwise} \end{cases}$$

and where $P(X_i = \perp \mid \mathcal{A})$ is such that the conditional probabilities sum to 1. The following property follows by construction:

Proposition 1 *The distribution of* B *is equivalent to the distribution defined by* S.

EXAMPLE For the CSP depicted in Figure 1, conditional probabilities are as follows. For X_3:

$$P(X_3 = a) = P(X_3 = b) = \frac{1}{2}$$

$$P(X_3 = \perp) = 0$$

For X_2 we have:

$$P(X_2 = a \mid X_3 = a) = P(X_2 = a \mid X_3 = b) = \frac{1}{2}$$

$$P(X_2 = b \mid X_3 = a) = \frac{1}{2}$$

$$P(X_2 = \perp \mid X_3 = b) = \frac{1}{2}$$

with all other conditional probabilities zero. For X_1, we get, e.g.:

$$P(X_1 = a \mid X_2 = a, X_3 = a) = \frac{1}{2}$$

$$P(X_1 = a \mid X_2 = a, X_3 = b) = \frac{1}{2}$$

Evaluating B (computing $P(X_1 \neq \perp)$) produces the exact number of solutions to the CSP. However, computing marginal probabilities in Bayes networks is a well-known NP-hard problem. Approximate belief updating is possible, and a number of algorithms exist (Dagum & Chavez 1993; Henrion *et al.* 1994; Santos Jr., Shimony, & Williams 1996). The present investigation proposes to use an approximation based on additional independence assumptions.

Approximation

To find the number of solutions to the CSP, we need to find, for the sink variable, $P(X_1 \neq \perp)$. In order to do that, we begin by considering the marginal probability for any variable X_i. With probabilities in the Bayes net defined as above, we can write (using the rule of conditioning), for an arbitrary assignment $\mathcal{B}$ to X_i:

$$P(\mathcal{B}) \;=\; \frac{1}{\mid D_{X_i} \mid} \sum_{\mathcal{A} \in \mathcal{C}^{\pi(X_i)}} P(\mathcal{A}) I(\mathcal{A} \cup \mathcal{B}) \qquad (2)$$

where $\mathcal{C}^Y$ is the set of all possible instantiations to a variable or set of variables Y, and $I(\mathcal{A} \cup \mathcal{B})$ is 1 when $\mathcal{A}$ is consistent with $\mathcal{B}$, and 0 otherwise. However, evaluating $P(\mathcal{A})$ is hard (time possibly exponential in n), and additionally evaluating the sum takes time exponential in $\mid \pi(X_i) \mid$.

Our approximation is based on assuming independence of the predecessors of each X_i. Define:

$$\hat{P}(\mathcal{B}) = \frac{1}{\mid D_{X_i} \mid} \sum_{\mathcal{A} \in \mathcal{C}^{\pi(X_i)}} I(\mathcal{A} \cup \mathcal{B}) \prod_{Y \in \pi(X_i)} \hat{P}(\mathcal{A}_Y) \qquad (3)$$

If indeed independence holds (it holds vacuously for nodes with no predecessors (called "source" nodes)), then $\hat{P}(X_i) = P(X_i)$. Otherwise $\hat{P}(X_i)$ is only an approximation to $P(X_i)$, where the error results from using the approximate term $\hat{P}(\mathcal{A}_Y)$ in place of $P(\mathcal{A}_Y)$, and from statistical dependencies between the random variables in Y. If, in addition, all constraints are binary, then relation I is likewise a product of independent parts, i.e.:

$$I(\mathcal{A} \cup \mathcal{B}) = \prod_{Y \in \pi(X_i)} I(\mathcal{A}_Y \cup \mathcal{B}) \qquad (4)$$

Thus, we can re-write equation 3 as:

$$\hat{P}(\mathcal{B}) = \frac{1}{\mid D_{X_i} \mid} \sum_{\mathcal{A} \in \mathcal{C}^{\pi(X_i)}} \prod_{Y \in \pi(X_i)} \hat{P}(\mathcal{A}_Y) I(\mathcal{A}_Y \cup \mathcal{B})$$

And via independence of terms:

$$\hat{P}(\mathcal{B}) = \frac{1}{\mid D_{X_i} \mid} \prod_{Y \in \pi(X_i)} \sum_{\mathcal{A} \in \mathcal{C}^Y} \hat{P}(\mathcal{A}) I(\mathcal{A} \cup \mathcal{B}) \qquad (5)$$

The latter takes $O(d \mid \pi(X_i) \mid)$ time to evaluate (with d being the largest domain size of the variables). To evaluate the approximation for all variables, one pass from X_n down to X_1 suffices, as follows. For i from n down to 1 do:

1. Compute equation 5 once for each $\mathcal{B} = \{(X_i, v)\}$, with v ranging over D_{X_i}. (A product over zero terms is assumed to be 1.)
2. Let $\hat{P}(X_i = \perp) = 1 - \sum_{v \in D_{X_i}} \hat{P}(X_i = v)$.

The latter computation for the entire CSP clearly takes $O(d^2 \mid E \mid)$, and thus is linear in the number of constraints (at most quadratic in n). This is usually negligible in comparison with the time of actually finding all the solutions, which is exponential in n.

To clarify the meaning of the above one pass calculation of probabilities, let us follow in detail the calculation of probabilities for the variables and values of figure 1. Starting with X_3 (a source variable), where $\hat{P} = P$:

$$\hat{P}(X_3 = a) = \hat{P}(X_3 = b) = \frac{1}{2}$$

These initial probabilities were based on the assumption of uniform a-priori probability for all values in the domain of the variable X_3.

Moving down the ordered graph we can now calculate the *conditional probability* for the values a, b of the variable X_2 (since we have $\pi(X_2) = \{X_3\}$). This is an approximation to the probability of instantiating the variable X_2 to the values a, b as part of a consistent solution. (Actually, in this example, the values for X_2 will be exact, since X_3 is the only predecessor, and the values for X_3 are exact.)

$$\hat{P}(X_2 = a) = \frac{1}{2}(P(X_3 = a) + P(X_3 = b)) = \frac{1}{2}$$

$$\hat{P}(X_2 = b) = \frac{1}{2} P(X_3 = a) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$$

In order to calculate the approximate probabilities for the variable X_1 one uses equation 5 to get:

$$\hat{P}(X_1 = a) = \tfrac{1}{2}(P(X_3 = a) + P(X_3 = b))\hat{P}(X_2 = a) = \tfrac{1}{4}$$

$$\hat{P}(X_1 = b) = \frac{1}{2} P(X_3 = b) \hat{P}(X_2 = b) = \frac{1}{16}$$

The approximation to the overall number of solutions in this particular example is given by

$$N\ \hat{P}(X_1 \neq \perp) = 8 \times (\frac{1}{4} + \frac{1}{16}) = \frac{5}{2} \qquad (6)$$

This is a slightly optimistic approximation in this case, since there are exactly 2 solutions to the example in Figure 1, out of 8 possible instantiations.

Our approximation (equation 5) has the following properties:

Theorem 3 *Let B be tree structured, with X_1 being the sink. Then the approximation equation 5 computes the exact distribution over X_1.*

Proof outline: For source nodes $\hat{P}(X) = P(X)$ (base case). In a Bayes network with no evidence, a node is independent of all nodes that are neither its ancestors or its descendents. We have a tree, and thus for any node X, its predecessors must be (a-priori) independent. Let X be any node in the tree. If for all $Y \in \pi(X)$ we have $\hat{P}(Y) = P(Y)$, then since the predecessors are independent, we get $\hat{P}(X) = P(X)$. The theorem follows by induction. □

Thus, for tree structured binary CSPs, since we can always create an equivalent Bayes network with the above properties, our approximation in this case reduces to that of (Dechter & Pearl 1988), and provides an exact count of the number of solutions to the CSP. However, in the general case (when the CSP is not necessarily tree structured), our estimate is less optimistic. That is, we have the following property:

Theorem 4 *For binary CSPs, the following equation holds (with $C(X,x)$ as defined by equation 1):*

$$N\ \hat{P}(X_1 \neq \perp) \leq \sum_{x \in D_{X_1}} C(X_1, x)$$

Proof outline: Re-write equation 1 in terms of probabilities, for $\mathcal{B}$ an arbitrary assignment to X_i:

$$P_C(\mathcal{B}) = \frac{1}{|\ D_{X_i}\ |} \prod_{Y \in \pi'(X_i)} \sum_{\mathcal{A} \in \mathcal{C}^Y} P_C(\mathcal{A}) I(\mathcal{A} \cup \mathcal{B})$$

where $\pi'(X)$ denotes the predecessors of X in the spanning tree generated for evaluating C. The term $P_C(X,x)$ is defined by the equation $K(X)P_C(X,x) = C(X,x)$, with $K(X)$ being the number of possible combined states of all ancestors of X (including X) in the tree (that is, the size of the cross product of the domains). For source nodes, we clearly have $P_C(X) = \hat{P}(X)$ (base case). Now, assume that for all predecessors of $Y \in \pi(X)$ in the graph, $P_C(Y) \geq \hat{P}(Y)$ (for assignments not containing $\perp$). Since $\pi'(X)$ is the set of predecessors in the spanning tree, which is a sub-graph of G, then $\pi'(X) \subseteq \pi(X)$, and the product for $P_C(X)$ contains (non-strictly) fewer terms than $\hat{P}(X)$, where each term is at most 1. Thus, since $P_C(Y) \geq \hat{P}(Y)$ for every term in the summation, we also have $P_C(X) \geq \hat{P}(X)$. The theorem follows by induction. □

Showing a non-trivial lower bound on our estimate is more difficult. Nevertheless, the following lower bound follows immediately from equation 5:

Proposition 2 *If the CSP is arc-consistent, then $\hat{P}(X_1 \neq \perp) > 0$.*

The independence assumption as stated above is problematic. While independence among CSP domain values may not be too far off the mark, that is clearly *not* the case for the value $\perp$. To see this, consider a tree-shaped CSP (for which our evaluation is exact), and add several vacuous binary constraints. Our estimate will propagate the probability of $\perp$ among multiple paths here, and thus cause a major underestimation of the number of solutions. Fixing this problem is relatively easy, if we take care not to count the probability of $\perp$ multiple times. There are several ways to do that, all of which are tantamount to including a normalization factor in equation 5, as follows:

$$\hat{P}'(\mathcal{B}) = \frac{1}{|D_{X_i}|} \prod_{Y \in \pi(X_i)} \sum_{\mathcal{A} \in \mathcal{C}^Y} \eta(Y, X_i) \hat{P}'(\mathcal{A}) I(\mathcal{A} \cup \mathcal{B}) \qquad (7)$$

where the normalizing factor can be defined in several ways, as follows:

1. Uniformly propagating the influence of $\perp$ to all successors of Y, in a manner suggested first in (Charniak & Husain 1991). To do that, let $k(Y)$ be the number of successors of Y, and:

$$\eta(Y, X_i) = \left(\sum_{y \in D_Y} \hat{P}'(Y = y) \right)^{\frac{1}{k(Y)} - 1} \qquad (8)$$

2. Propagating the influence of $\perp$ only through one preferred successor of Y. In this method, $\eta(Y, X_i)$ is 1 for the preferred successor. For all other successors:

$$\eta(Y, X_i) = \left(\sum_{y \in D_Y} \hat{P}'(Y = y) \right)^{-1} \qquad (9)$$

With the preferred successor method, the estimate is guaranteed to be no greater than predicted by equation 1, but always greater than our original estimate:

Theorem 5 *The following bounds hold, with $\hat{P}'$ as defined in equations 7, 9:*

$$N\ \hat{P}(X_1 \neq \perp) \leq N\ \hat{P}'(X_1 \neq \perp) \leq \sum_{x \in D_{X_1}} C(X_1, x)$$

where the preferred successor is such that there is an edge from Y to X_i in the spanning tree method.

Proof outline: The leftmost inequality clearly holds, since in equation 7 we are always multiplying by some $\eta \geq 1$. The rightmost equality follows from an argument similar to the proof of theorem 4, as follows: replace $\hat{P}$ by $\hat{P}'$ in the proof. The base case ($P_C(X) \geq \hat{P}'(X)$ for source nodes) still holds. Likewise, the induction step $P_C(Y) \geq \hat{P}'(Y)$ holds, because due to equation 9, we have $\sum_{y \in D_Y} \eta(Y, X_i)\hat{P}'(\mathcal{A}) \leq 1$ for every $Y \in \pi(X_i)$ in equation 7. □

In the running example, if the arc (X_3, X_1) is not in the spanning tree, all probability estimates remain the same, because $P(X_3 = \perp) = 0$.

Experimental Results

The theoretical results of the previous section show that our approximation to the number of solutions is less optimistic than the overestimate of (Dechter & Pearl 1988). However, they still leave much latitude within the bounds, and it is obviously possible to construct cases where both estimates are very far off the mark. An obvious such case would be a CSP consisting of just equality constraints between every pair of variables. It would be interesting to know whether our approximation is useful for "typical" CSPs.

Here, we present results for 96 randomly generated CSPs, for problems with 6, 8, 10, and 12 variables. Each variable has a domain size of 9. The probability that a constraint exists between any pair of variables (CSP density) ranges from 0.5 to 1. In a constraint, the probability that a given pair of values is allowed was 0.65. For a meaningful comparison, we had to count the true number of solutions, hence the relatively small problem sizes. Since the number of solutions fluctuates wildly with change in parameters, we elected not to differentiate between the parameters in presenting the results (figure 2). Rather, all results are compared to ground truth (straight line), where the horizontal axis is the number of solutions, and the vertical axis the approximate number of solutions for the trial.

Evidently, while both preferred successor and uniform propagation are well within an order of magnitude of the correct value for most problem instances, the spanning tree method is far off the mark. An exception occurs for CSPs generated with constraint densities 0.1 and 0.2, which are nearly trees, where all schemes tend to converge on the correct value (not shown). Between the preferred successor and uniform propagation there is no clear winner: the former is better theoretically (less optimistic than spanning tree), but the latter appears better in several problem instances on the right-hand side of figure 2. The "bands" of estimates resulting from the spanning-tree method in figure 2 correspond to different problem sizes.

Discussion

The paper makes an explicit and natural correspondence between counting the number of solutions of a

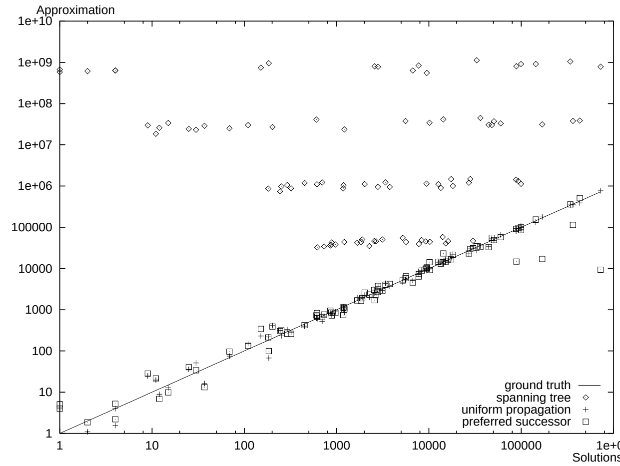

Figure 2: Comparison of Solution Counts

CSP and computing marginal probabilities in Bayes networks, by using a well defined stochastic assignment process over CSP variables. Using the notion of statistical independence in the stochastic domain, we defined a new method of approximating the number of solutions, which turned out to work significantly better than prior work (Dechter & Pearl 1988), the only deterministic competing method of which we are aware. In particular, we preserve the property of correctness for tree-shaped CSPs, while being less (over)optimistic in the general case. Experimental results over randomly generated CSPs show that while our approximation was close to the actual value, the competing method was frequently off by several orders of magnitude. We note in passing that an earlier attempt to use stochastic methods in CSPs appears in (Knuth 1975), which uses biased distributions in a stochastic model to estimate run-time as well as number of terminal nodes (not necessarily solutions) in the search tree.

In our model, the ratio of probabilities for assignments of values to any particular variable is interpreted as an approximation to the ratio of the number of complete solutions that include these particular assignments. This might well be used as a *value ordering* heuristic and in fact was suggested as such by (Dechter & Pearl 1988). We have performed some preliminary experiments to determine the impact of a static value ordering heuristic based on the ratio of the probabilities for inclusion in a solution. The complete results will be reported in a future paper dedicated to value ordering (see also (Solotorevsky, Shimony, & Meisels 1996) for results on a similar heuristic for CSPs with counters), but a preliminary run is discussed below.

Consider ordering the values of X_1 according to the number of solutions of the CSP containing that value assignment to X_1 (we get this for free in the experiments, since we already needed to find all the solutions). We have calculated a "distance" between orderings of the values of X_1 resulting from each approx-

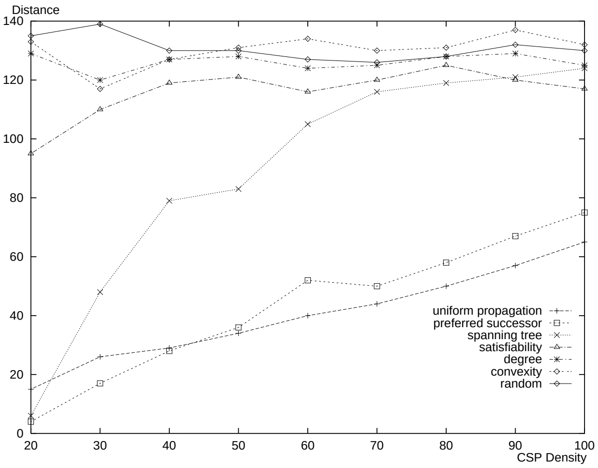

Figure 3: Comparing Value Orders for X_1

Density	0.55	0.6	0.65
Consistent	99	78	15
Value-Ordering Heuristic	Checks $\times 10^3$		
No Heuristic	783	4182	10565
Convexity	548	2652	10564
Spanning tree	731	3803	9783
Uniform Propagation	531	2394	8741

Table 1: Constraint Checks for Various Heuristics

imation or value-ordering heuristic (Tsang 1993), and the above exact ordering. Figure 3, plots distances (vertical) vs. CSP density (horizontal).

It is clear from figure 3 that our approximated probabilities perform much better than other value ordering heuristics. In particular, ordering the values by the spanning tree approximation method is as bad as randomly ordering values, for dense graphs, and is dominated by our method in general. These preliminary results on the behavior of other value ordering heuristics explain the lack of enthusiasm for value ordering heuristics in the CSP community to-date. However, our methods suggest that value ordering may still have some positive utility.

Preliminary results, examining effect of static value ordering on search for first solution (Table 1), support our scheme (Uniform Propagation). The experiment used the FC-CBJ algorithm with static variable ordering on problems with 25 variables, each with a domain size of 15. In each binary constraint, probability that a value pair is allowed was 0.7. CSP density was varied from 0.55 to 0.65 (lower values created very easy problem instances, whereas higher values created only instances with no solutions). For each density, 100 problem instances were generated, but only those with solutions (consistent) are reported.

While it is possible to construct pathological cases for our approximation, our contribution is not just in the actual approximation method: the reduction from Bayes networks might provide further useful insights in the future. For instance, it would be interesting whether other Bayes network approximation methods, such as those based on (finite-ϵ) κ-calculus (Henrion *et al.* 1994), or stochastic simulation (Dagum & Chavez 1993) would also be useful for CSPs.

References

Charniak, E., and Husain, S. 1991. A new admissible heuristic for minimal-cost proofs. In *Proceedings of AAAI Conference*, 446–451.

Cheeseman, P.; Kanefsky, B.; and Taylor, W. M. 1991. Where the really hard problems are. In *IJCAI-91*, 331–337.

Cooper, G. F. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 42 (2-3):393–405.

Dagum, P., and Chavez, R. M. 1993. Approximating probabilistic inference in Bayesian belief networks. *PAMI* 15(3):244–255.

Dagum, P., and Luby, M. 1993. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence* 60 (1):141–153.

Dechter, R., and Pearl, J. 1988. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence* 34:1–38.

Haralick, R. M., and Elliott, G. L. 1980. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence* 14:263–313.

Henrion, M.; Provan, G.; Favero, B. D.; and Sanders, G. 1994. An experimental comparison of numerical and qualitative probabilistic reasoning. In *UAI, Proceedings of the Tenth Conference*, 319–326.

Knuth, D. E. 1975. Estimating the efficiency of backtrack programs. *Mathematics of Computation* 39(129):121–136.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann.

Santos Jr., E.; Shimony, S. E.; and Williams, E. 1996. Sample-and-accumulate algorithms for belief updating in Bayes networks. In *UAI, Proceedings of the 12th Conference*, 477–484. Morgan Kaufmann. (To appear in IJAR).

Smith, B. M. 1994. Phase transition and the mushy region in csp. In *ECAI, Proceedings of the 12th European Conference*, 100–104.

Solotorevsky, G.; Shimony, S. E.; and Meisels, A. 1996. CSPs with counters: a likelihood-based heuristic. In *ECAI, Workshop on Non-Standard Constraint Processing.* (To appear in JETAI).

Tsang, E. 1993. *Foundations of Constraint Satisfaction.* Academic Press.