Approximate Resolution of Hard Numbering Problems

Olivier Bailleux and Jean-Jacques Chabrier

CRID Université de Bourgogne
Faculté des Sciences Mirande
BP138 21004 Dijon Cedex France
ob@crid.u-bourgogne.fr chabrier@crid.u-bourgogne.fr

Abstract

We present a new method for estimating the number of solutions of constraint satisfaction problems[1]. We use a stochastic forward checking algorithm for drawing a sample of paths from a search tree. With this sample, we compute two values related to the number of solutions of a CSP instance. First, an unbiased estimate, second, a lower bound with an arbitrary low error probability. We will describe applications to the Boolean Satisfiability problem and the Queens problem. We shall give some experimental results for these problems.

Introduction

The class NP is the set of decision problems whose instances are assertions that can be proved in polynomial time. The NP-Complete problems are the hardest problems in NP. These can not be solved in polynomial time under the assumption $P \neq NP$. All these problems have the same expression power in the sense that every NP-Complete problem can be polynomialy reduced to each another (Garey & Johnson 1979). Some of them, such as CSP[2], allow us to specify many practical problems in a very simple way.

SAT is the problem of deciding if there is an assignment for the variables in a boolean expression that makes this expression true. It is a restriction of CSP with binary domains. Since the 1960's, a lot of research and publications have been done on practical resolution of SAT instances (Andre & Dubois 1992; Billonnet & Sutter 1992; Selman, Levesque, & Mitchell 1992; Selman & Kautz 1993). There are two kind of complete solving algorithms for SAT : resolution methods (Robinson 1963), based on theorem proving, and semantical methods (Davis & Putman 1960), based on the search for solutions. Moreover, some classes of SAT instances have been the subject of theoretical and experimental studies. Without exhausting all publications, we could mention the study of random variables associated with the number of solutions (Dubois & Carlier 1991), with the satisfiability (Simon *et al.* 1986) of randomly generated instances, the generation of hard instances for the evaluation of solving algorithms (Mitchell, Selman, & Levesque 1992; Gent & Walsh 1994), the characterisation of classes of polynomial complexity instances (Parshina 1992; Dalal & Etherington 1992).

The problem of counting the solutions of SAT instances, called #SAT, is #P-complete (Valliant 1979). Every numbering problem whose solutions can be generated in polynomial time by a non-deterministic program can be polynomialy reduced to #SAT. It is interesting to see that some numbering problems related to polynomial search problems such as 2-SAT are #P-complete. Several counting algorithms for #SAT (Dubois 1991; Lozinskii 1992) have been published. Unfortunately, many instances of #SAT, even with small sizes (100 variables, 200 clauses), are intractable using these algorithms.

In this paper, we propose a method for estimating the number of solutions of CSP instances. Our method is based on the estimate of the expectancy of a random variable defined on the set of paths that link the root of a search tree to it's leaves. In the first section, we present the principle and the limits of the method. In the second section, we illustrate applications of the method to SAT and the Queens problem. We also study the interest of the proposed approach in comparison with a method based on random drawings from the initial search space. The third section presents some experimental results. Concluding remarks and research perspectives are contained in the last section.

Statistics on trees

A leaf-labeled tree is a tree where each leaf is associated with an integer.

We write $< \alpha_1, \ldots, \alpha_n >$ as a sequence of n items, and $s_1|s_2$, the sequence obtained by appending a sequence s_1 to a sequence s_2.

[1]This work is partially supported by the PRC-IA 'Aspects algorithmiques de la résolution des problèmes exprimés à l'aide de contraintes'.

[2]Constraint Satisfaction Problem

Let Γ be the set of leaf-labeled trees.

$$\begin{cases} x \in N \Rightarrow leaf(x) \in \Gamma \\ k \in N^*, x_1, \ldots, x_k \in \Gamma \Rightarrow node(< x_1, \ldots, x_k >) \in \Gamma \end{cases}$$

Where N is the set of positive integers and $N^* = N \backslash \{0\}$

For each leaf-labeled tree $t \in \Gamma$, we define a set $\Omega(t)$ of paths that link the root to the leaves. Each path is coded by a sequence of integers.

$$\begin{cases} \Omega(leaf(x)) = \{< 0 >\} \\ \Omega(node(< x_1, \ldots, x_k >) = \\ \quad \bigcup_{i=1}^{k} \{< i > | r, r \in \Omega(x_i)\} \end{cases}$$

Let f be a function which assign to each path in Ω the label of the associated leaf.

$$\begin{cases} f(leaf(x), < 0 >) = x \\ f(node(< x_1, \ldots, x_k >), < a_1, \ldots, a_n >) = \\ \quad f(x_{a_1}, < a_2, \ldots, a_n >) \end{cases}$$

Let P be a probability on Ω.

$$\begin{cases} P(leaf(x), < 0 >) = 1 \\ P(node(< x_1, \ldots, x_k >), < a_1, \ldots, a_n >) = \\ \quad \frac{1}{k} P(x_{a_1}, < a_2, \ldots, a_n >) \end{cases}$$

$P(t, c)$ is the probability of drawing a path c from a tree t using the following method. We start with the root. Each new node is drawn uniformly from the current node's children.

Let X be a random variable on $(\Omega(t), P)$ which associates to each path $c \in \Omega(t)$ the value $f(t, c)/P(t, c)$. By definition,

$$E[X] = \sum_{c \in \Omega(t)} f(t, c) \qquad (1)$$

Let I be a search problem instance and S be the set of solutions of I. Let t be the search tree of a counting algorithm with input I. We suppose that a leaf of t can find multiple solutions. If the label of each leaf is the size of the associated packet of solutions, we can use a sample of paths for making an unbiased estimate of $|S|$.

Let $< c_1, \ldots, c_N >$ be a sample of paths generated by N independent drawings with respect of probability P, we have :

$$|S| \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(t, c_i)}{P(t, c_i)} \qquad (2)$$

Labeling all the leaves with value 1, it is possible to obtain an estimate of the number of leaves in the search tree. Let $\eta(t)$ be the number of leaves of a tree t. We have :

$$\eta(t) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{1}{P(t, c_i)} \qquad (3)$$

If t is a k-ary tree (each internal node having exactly k children), there is a relation betwen the number of leaves $\eta(t)$ and the number of nodes $|t|$:

$$|t| = \frac{1}{k-1}(k\eta(t) - 1) \qquad (4)$$

If t is a neuronal tree (each internal node having at least 2 children), we only have :

$$\eta(t) < |t| < 2\eta(t) \qquad (5)$$

With the relations 4, 5 and 3 we can obtain a piece of information about the size of a tree without exhaustively exploring it.

From a practical point of view, the quality of the estimates depends on both the distribution of the random variable X and the sample size. Broadly speaking, we do not know the distribution of X. However, we can obtain, with a given error probability, a *lower bound* of $E[X]$.

Proposition

Let $\mu_1, \ldots, \mu_n$ be a sequence of independent unbiased estimates of $E[X]$ and a real $\lambda > 0$. We have :

$$P(\frac{1}{\lambda} Min\{\mu_1, \ldots, \mu_n\} > E[X]) \leq \frac{1}{\lambda^n} \qquad (6)$$

The inequality 6 is given as a direct consequence of the following well known lemme : if Y is a positive random variable, $\lambda > 0$ a real and y a sample value of Y then $P(y > \lambda E[Y]) \leq \frac{1}{\lambda}$.

Method implementation

General framework

The two problems that we consider in the following can be specified as constraint satisfaction problems (CSP). A CSP instance is a 3-tuple $< V, D, R >$, where $V = \{v_1, \ldots, v_n\}$ is a set of variables, $D = \{d_1, \ldots, d_n\}$ the associated set of domains and R a set of contraints defining relations betwen the values assigned to the variables (Tsang 1993). A label is a couple (v_i, x), $v_i \in V$, $x \in d_i$, that stand for the assignment of the value x to v_i. A set of labels $\{(v_1, x_1), \ldots, (v_n, x_n)\}$ is a solution of $< V, D, R >$ iff it satisfies all the constaints of R.

The figure 1 describes the 'forward checking' algorithm that can be used for counting the solutions of a CSP instance. This algorithm explores a search tree where each internal node is associated with a search context $< F, A, D, R >$. F is a set of free variables, A is a set of labels, D is the set of domains of the free variables and R is a set of unsolved constraints. The function *Propage* simplifies the current context, notably by removing from the domains of D some values that are incompatible with the labels of A, according to the constraints.

The figure 2 describes a stochastic algorithm which draws a path Ψ from the search tree of the 'forward

```
Algorithm $Count(<V, D, R>: context)$
 Begin
  Return $Explore(<V, \{\}, D, R>)$;
 End

Function $Explore(<F, A, D, R>: context) : integer$
 Begin
  If $\{\} \in D$ or one of the constraints is violated
   Return (0);
  Else If all the contraints are satisfied
   Return $(\prod_{d \in D} |d|)$;
  Else
   Choose $v_i \in F$;
   Return $\sum_{x \in d_i} Explore(Propage(c(x, v_i)))$
   where $c(x, v_i) = <F \backslash \{v_i\}, A \cup \{(v_i, x)\}, D \backslash \{d_i\}, R>$
  End If
 End
```

Figure 1: Forward checking

```
Algorithm $Draw(<V, D, R>: context)$
 Begin
  Return $Explore(<V, \{\}, D, R>)$;
 End

Function $Explore(<F, A, D, R>: context)$ :
$(probability, integer)$
 Begin
  If $\{\} \in D$ or one of the constraints is violated
   Return (1,0);
  Else If all the contraints are satisfied
   Return $(1, \prod_{d \in D} |d|)$;
  Else
   Choose $v_i \in F$;
   Draw $x$ uniformly from $d_i$
   Let $(p, k) = Explore(Propage(c(x, v_i)))$
   where $c(x, v_i) = <F \backslash \{v_i\}, A \cup \{(v_i, x)\}, D \backslash \{d_i\}, R>$
   Return $(\frac{1}{|d_i|} p, k)$;
  End If
 End
```

Figure 2: Stochastic forward checking

checking' algorithm of figure 1 and returns a couple (p, k) where

- p is the probability of Ψ,
- k is the number of solutions associated with Ψ.

Application to SAT

A SAT instance is a special case of CSP instance $< V, D, R >$ where V is a set of boolean variables, D a set of boolean domains and R a set of clauses. Each clause is a disjonction of literals.

We will use a variant of the Davis and Putman (D&P) algorithm (Davis & Putman 1960; Franco & Paull 1983), which is the general algorithm described in figure 1 with a specific propagation procedure : For each clause with only one literal v_i (resp. $\neg v_i$),

- the label $(v_i, 1)$ (resp. $(v_i, 0)$) is added to the set A,
- the clauses that contain v_i (resp. $\neg v_i$) are removed from the set of clauses R,
- the literal $\neg v_i$ (resp. v_i) is removed from each clause of R where it occurs.

With a good heuristic for the choice of variables, the generation of a solution by drawing from the search tree can be much more probable than the generation of a solution by directly drawing assignment to the variables. As an example, let us consider a r-SAT instance, that is a SAT instance whose clauses have r literals (r-SAT is NP-Complete for $r > 2$). Whatever the heuristic, all the solutions are accessible in the search tree. They are formed in packets associated with leaves of the tree. Each packet is characterized by the assignation of some variables.

For each real x, we write $\lceil x \rceil$ the smaller integer upper or equal to x.

Let $2^{k_1}, \ldots, 2^{k_m}$ be the sizes of packets of solutions, ranked in an abitrary order.

Let P_1 be the probability of generating a solution by directly drawing assignments to the variables. Let P_2 be the probability of generating a solution by drawing a path with the stochastic forward checking algotithm. Let N be the number of solutions.

Proposition

If the heuristic always chooses one of the variables that are in the smaller clauses, we have

$$\frac{P_2}{P_1} \geq 2^{\frac{n - log_2(N)}{r} - 1} \qquad (7)$$

Proof

$$P_1 = \frac{N}{2^n} = \frac{2^{k_1} + \ldots + 2^{k_m}}{2^n} = 2^{k_1 - n} + \ldots + 2^{k_m - n}$$

The stochastic forward checking algorithm makes two kinds of variable assignments : propagation assignments (made by the propagation procedure) and random assignments. Each path Ψ ending to a packet of 2^k solutions is associated with $n - k$ assignments. If the heuristic always chooses a variable that occur in a clause of minimal size, there is at least a propagation assignment for $r - 1$ random assignments. Hence Ψ has at most $\lceil \frac{r-1}{r}(n-k) \rceil$ internal nodes.

So, for each path Ψ ending to a packet of 2^k solutions, we have

$$P(\Psi) \geq \frac{1}{2^{\lceil \frac{r-1}{r}(n-k) \rceil}} \geq \frac{1}{2^{1 + \frac{r-1}{r}(n-k)}} = \frac{1}{2} 2^{\frac{r-1}{r}(n-k)}$$

where $P(\Psi)$ is the probability for drawing Ψ.

hence

$$P_2 \geq \frac{1}{2}(2^{\frac{r-1}{r}(k_1-n)} + \ldots + 2^{\frac{r-1}{r}(k_m-n)})$$

yet

$$2^{\frac{r-1}{r}(k_i-n)} = 2^{k_i-n}2^{\frac{1}{r}(n-k_i)}$$

hence

$$P_2 \geq \frac{1}{2}(2^{k_1-n} + \ldots + 2^{k_m-n})2^{\frac{1}{r}Min\{n-k_i, i\in 1..m\}}$$

moreover

$$Max\{k_i, i \in 1..m\} \leq log_2(N)$$

and

$$\frac{P_2}{P_1} \geq \frac{1}{2}(2^{\frac{1}{r}Min\{n-k_i, i\in 1..m\}})$$

hence

$$\frac{P_2}{P_1} \geq 2^{\frac{n-log_2(N)}{r}-1}$$

Application to the Queens problem

A configuration for the n Queens problem is a n-uple $< x_0, \ldots, x_{n-1} >$, $x_i \in 0..n-1$, $i \in 0..n-1$. The component x_i is the position of a queen on the row of rank i on a n by n chessboard. This representation implies that there is one queen on each row. A solution is a configuration such that there is at most one queen on each line and each diagonal.

$< x_0, \ldots, x_{n-1} >$ is a solution if and only if

$$\begin{cases} \forall i,j \in 0..n-1, x_i \neq x_j \\ \forall i,j \in 0..n-1, |i-j| \neq |x_i - x_j| \end{cases}$$

The propagation function removes from the domains of free variables the values that are inconsistent with the current assignment. The heuristic we used consists to choose, for each new assignment, a variable among the ones whose domain size is minimal.

As with the previous algorithm, because of domains reduction, generating a solution by drawing a path from the search tree is more probable than generating a solution by drawing assignments to the variables.

Experimental results

Random 3-SAT instances

The results presented in this section have been obtained with 3-SAT instances generated by independent drawings of clauses performed under uniform conditions. The clauses with duplicates literals were not allowed.

The figure 3 shows the average number of the search tree nodes, according to the number of clauses, for 3-SAT instances of 50 variables. The used algorithm is a Davis & Putnam with an heuristic decribed in (Andre & Dubois 1992). We note that instances of roughly 60 clauses are the hardest for this algorithm. These instances are not interesting because we can estimate their number of solutions by randomly drawing assignments to the variables. In contrast, instances of 100

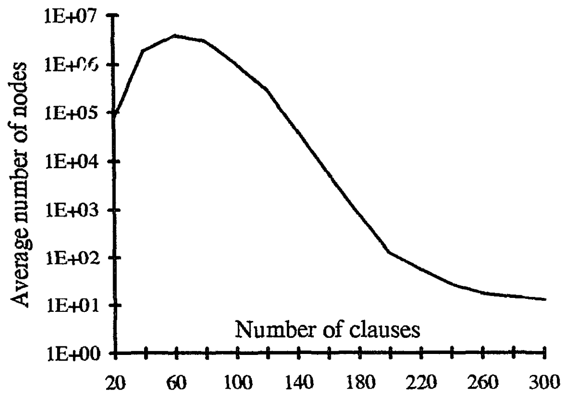

Figure 3: Random 3-SAT instances, 50 variables : average number of nodes in the search tree according to the number of clauses.

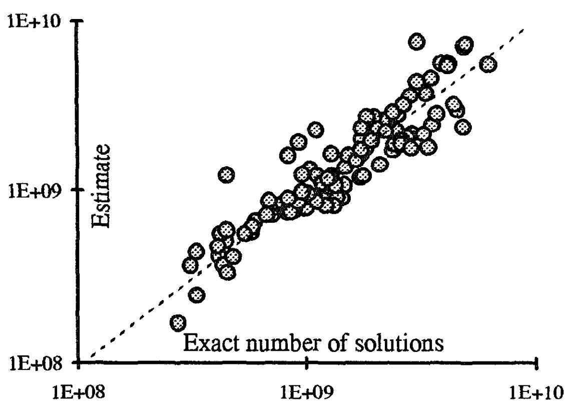

Figure 4: Random 3-SAT instances, 50 variables, 100 clauses : correlation between estimates and exact numbers of solutions.

clauses are hard for both exhaustive exploration of the search tree and drawing in configurations space.

The figure 4 gives the results of estimations on random 3-SAT instances of 50 variables and 100 clauses. Each estimation was computed from a sample of 1000 paths, that is on average 1.5% of internal nodes of the search tree. Each point is associated with an instance. The x-coordinate of a point is the exact number of solutions of the associated instance. Its y-coordinate is the estimated number of solutions. The concentration of the points around the first diagonal (doted line) shows the quality of the estimations.

Structured SAT instances

In this section, we present some results about SAT instances associated with graph colouring problems. The

Instance	Number of solutions	Average of estimates	Rel. std. deviation
Btree(4)	$4.92 \cdot 10^4$	$4.82 \cdot 10^4$	5.8%
Btree(5)	$3.22 \cdot 10^9$	$3.03 \cdot 10^9$	21%
Btree(6)	$1.38 \cdot 10^{19}$	$8.83 \cdot 10^{18}$	40%
Btree(7)	$2.55 \cdot 10^{38}$	$9.37 \cdot 10^{37}$	80%
Ramsey(6)	$1.10 \cdot 10^6$	$1.13 \cdot 10^6$	19%
Ramsey(7)	$1.10 \cdot 10^8$	$1.03 \cdot 10^8$	42%

Table 1: Estimates of number of solutions for structured SAT instances.

aim of the first problem is to colour each node of a complete binary tree with 3 colours, such as nodes that are connected by an edge do not have the same colour. To specify the constaints related to a tree of deep n, we built a SAT instance called *Btree*(n) as follow : With each node are associated a set of three variables, one for each possible colour, and a set of clauses specifying that exactly one of these variables has the value 1. With each edge is associated a set of clauses specifiying that the two connected nodes have different colours.

We chose these instances on the one hand because we can easily compute their number of solutions, and on the other hand because it would be relatively difficult to count their solutions by a D&P approach without refinement. Because of constraints related to the unicity of the colour of each node, the D&P program must assign all the variables (directly or by propagation), to generate each solution. It can generate only one solution at the same time.

The second problem we experimented with is called *Ramsey*(3, 3, 3, 2). The aim is to colour the edges of a complete graph in such a way that there is not any monochromatic triangles. For a graph of n vertices, we built a SAT instance called *Ramsey*(n) with the same approach than the previous problem.

The table 1 gives, for each instance, the exact number of solutions, the average and the relative standard deviation of 30 estimates on samples of 1000 paths.

Queens problem

The table 2 presents the results of a series of 30 measures on some instances of the Queens problem. The estimates were obtained from samples of 1000 paths. For each instance, we give the exact number of solutions, the average and the relative standard deviations of estimates and the ratio of the number of explorated nodes to the total number of nodes in the search tree.

The table 3 gives the lower bounds, with an error probability lower than 10^{-30}, for the number of solutions of Queens problem instances for which we do not know any reference of usable counting method.

Size of inst.	Nb. of sol.	Av. of est.	Rel. std. dev.	expl. nodes
12	$1.42 \cdot 10^4$	$1.42 \cdot 10^4$	3.5%	$3.2 \cdot 10^{-1}$
13	$7.37 \cdot 10^4$	$7.32 \cdot 10^4$	4.3%	$6.8 \cdot 10^{-2}$
14	$3.66 \cdot 10^5$	$3.65 \cdot 10^5$	4.7%	$1.3 \cdot 10^{-2}$
15	$2.28 \cdot 10^6$	$2.28 \cdot 10^6$	4.4%	$2.4 \cdot 10^{-3}$
16	$1.48 \cdot 10^7$	$1.48 \cdot 10^7$	4.7%	$3.9 \cdot 10^{-4}$
17	$9.58 \cdot 10^7$	$9.41 \cdot 10^7$	5.3%	$6.7 \cdot 10^{-5}$
18	$6.66 \cdot 10^8$	$6.63 \cdot 10^8$	3.8%	$1.0 \cdot 10^{-5}$

Table 2: Estimates of number of solutions for Queens problem instances.

Size of instance	Probable lower bound
20	$2 \cdot 10^9$
40	$4 \cdot 10^{30}$
60	$2 \cdot 10^{56}$
80	$1 \cdot 10^{85}$
100	$5 \cdot 10^{115}$

Table 3: Lower bounds of number of solutions for Queens problem instances, with error probability lower than 10^{-30}.

Conclusion

We proposed a stochastic approach for making unbiased estimates of the number of solutions CSP instances. Our method uses the set of paths that link the root of a search tree to its leaves as sample space. This allows us to take advantage of the efficiency of the 'forward checking' algorithm.

We can always compute a lower bound for the number of solutions with a given error probability. Actually, we do not know how to compute a reliable upper bound. Despite this limitation, the experimental results seem very promising.

The method can be used for estimating the size of a search tree without exploring it exhaustively, with the same limitation as the estimation of the number of solutions. This can be useful, for instance, to evaluate the time we need to compute the exact number of solutions of a problem instance with a given algorithm and a given heuristic.

References

Andre, P., and Dubois, O. 1992. Utilisation de l'espérance du nombre de solutions afin d'optimiser la résolution d'un système sat. *C. R. Acad. Sci. Paris* 217–220.

Bailleux, O., and Chabrier, J.-J. 1995. Measures on sat landscapes by statistical exploration of search trees. Workshop on Studying and solving really hard problems at CP95.

Billonnet, A., and Sutter, A. 1992. An efficient algorithm for the 3-satisfiability problem. *Operation Research Letters* 29–36.

Dalal, M., and Etherington, D. W. 1992. A hierarchy of tractable satisfiability problems. *Information Processing Letters* 44:173–180.

Davis, and Putman. 1960. A computing procedure for quantification theory. *J. ACM* 7:201–215.

Dubois, O., and Carlier, J. 1991. Probabilistic approach to the satisfiability problem. *Theoretical Computer Science* 65–75.

Dubois, O. 1991. Counting the number of solutions for instances of satisfiability. *Theoretical Computer Science* 81:49–64.

Franco, J., and Paull, M. 1983. Probabilistic analysis of the davis putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics* 5:77–87.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability.* W. H. Freeman and Compagny.

Gent, P. I., and Walsh, T. 1994. The sat phase transition. In *Proceedings of ECAI 94.* John Wiley and Sons, Ltd.

Lozinskii, E. L. 1992. Counting propositional models. *Information Processing Letters* 41:327–332.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of sat problems. In *Proceedings of AAAI 92*, 459–465.

Parshina, N. A. 1992. Satifiability problem : some polynomial classes of conjunctive normal forms. *Kibernetica i Sistemnyi Analiz* 1:165–170.

Robinson. 1963. Theorem proving on computer. *J. ACM* 10:163–174.

Selman, B., and Kautz, H. 1993. An empirical study of greedy local search algorithms for satisfiability testing. In *Proceedings of the 11th National Conference on Artificial Intelligence.*

Selman, B.; Levesque, H.; and Mitchell, G. 1992. A new method for solving hard satisfiability problems. In *Proceedings of AAAI 92, San Jose, CA*, 440–446.

Simon, J.-C.; Carlier, J.; Dubois, O.; and Mouline, O. 1986. Statistical distribution of sat problem solutions, application to expert-systems. *C. R. Acad. Sci. Paris* 217–220.

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

Valliant, L. 1979. The complexity of enumeration and reliability problems. *SIAM J. Comput.* 8(3):410–421.