TECHNICAL REPORT NO. 291

On the Occurrence of Null Clauses in Random Instances of Satisfiability

by

John Franco

October, 1989

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY

Bloomington, Indiana 47405-4101

On the Occurrence of Null Clauses in Random Instances of Satisfiability*†

John Franco

Department of Computer Science,

Indiana University, Bloomington Indiana 47405

October 11, 1989

Abstract

We analyze a popular probabilistic model for generating instances of Satisfiability. According to this model, each literal of a set $L = \{v_1, \bar{v}_1, v_2, \bar{v}_2, ..., v_r, \bar{v}_r\}$ of literals appears independently in each of n clauses with probability p. This model allows null clauses and the frequency of occurrence of such clauses depends on the relationship between the parameters n, r, and p. If an instance contains a null clause it is trivially unsatisfiable. Several papers present polynomial average time results under this model when null clauses are numerous (e.g. [4,5]) but, until now, not all such cases have been covered by average-case efficient algorithms. In fact, a recent paper [2] shows that the average complexity of the pure literal rule is superpolynomial even when most random instances contain a null clause. We show here that a simple strategy based on locating null clauses in a given random input has polynomial average complexity if either $n \leq r^{.5}$, and $pr < \ln(n)/2$; or $n = r^{\epsilon}$, $\epsilon \neq 1$, and $pr < c(\epsilon)\ln(n)/2$; or $n = \beta r$, β a positive constant, and $2.64(1-e^{-2\beta pr}(1+2\beta pr)) < \beta e^{-2pr}$. These are essentially the conditions for which null clauses appear in random instances with probability tending to one [3]. These results are an improvement over some results in the references cited above. The strategy is as follows. Search

*This work was carried out in part at the FAW, Helmholtzstrasse 16, D-7900 Ulm/Donau, Germany.

†This work is based on research supported in part by the Air Force Office of Scientific Research, Grant No. AFOSR 84-0372.

the input for a null clause. If one is found, immediately decide the instance is unsatisfiable. Otherwise, set variables appearing exactly once to satisfy the clauses they occupy and determine satisfiability by exhaustively trying all possible truth assignments to the remaining literals of the input. Because the good average case performance depends completely on the presence of null clauses, we see this work as illuminating properties of the probabilistic model which cause polynomial average time rather than presenting a new algorithm with improved average time behavior.

1 Introduction

The Satisfiability problem is to determine whether there exists a truth assignment to the variables of a given CNF Boolean expression which cause it to have value *true*. If such a truth assignment exists we say the expression is satisfiable, otherwise it is unsatisfiable. The problem is NP-complete so there is no known polynomial time algorithm for solving it. Several papers have been concerned with the analysis of algorithms for Satisfiability that run in polynomial average time. These results depend on an assumed probabilistic input model. One popular model is the "random-clause-size" model which we refer to as $M(n, r, p)$.

Let $L = \{v_1, \bar{v}_1, v_2, \bar{v}_2, ..., v_r, \bar{v}_r\}$ be a set of $2r$ literals. According to the model $M(n, r, p)$, n disjunctions (called clauses) are generated as follows: for each clause C_i, for all literals $l \in L$, put l in C_i with probability p, independently of the placement of other literals and clauses. Notice that it is possible to generate an empty (or null) clause using this model. The preponderance or absence of null clauses in random instances is controlled by the product pr [3]. The theme of this paper centers around the fact that if an instance has a null clause it is trivially unsatisfiable.

We are primarily concerned with the average running time of algorithms when null clauses frequently appear in random instances generated according to $M(n, r, p)$. From [3] a random instance posseses a null clause with probability tending to 1 if the product $pr < \ln(n)/2$. However, in the literature, polynomial average time results for this range of pr are known only if $n = r^{\epsilon}$, $.5 \geq \epsilon > 0$ [4]; or $n = r^{\epsilon}, 1 > \epsilon > .5, pr < \sqrt{\ln(n)}/r^{\epsilon-.5}$ [4]; or $n = r^{\gamma}, \gamma > 1, pr < (\gamma-1)\ln(n)/(2\gamma)$ [5]. Furthermore, no polynomial average time results are known for the case $\lim_{n,r\to\infty} n/\sqrt{r} = \infty$

and $\ln(n)/2 < pr < r^{.5}$; this is a large range of pr where instances usually do not have null clauses. Also of interest is a result in [2] which shows that the pure literal rule requires superpolynomial average time if $n = r^{\epsilon}$, $1 > \epsilon > .5$, and $pr > \sqrt{\omega(r)\ln(n)}/r^{\epsilon-.5}$ where $\omega(r)$ is any growing function of r. This leaves a substantial range of pr for which null clauses exist in random instances with high probability but, up to now, no known polynomial average time algorithm exists and at least one non-trivial algorithm requires superpolynomial average time. This situation has been philosophically uncomfortable because one would expect that such obviously unsatisfiable instances should be "easy" to solve on the average.

In this paper we show that a very simple algorithm for solving Satisfiability has polynomial average time behavior when $pr < \ln(n)/2$ and $n < r^{.5}$; or $pr < (1-\epsilon-\delta)\ln(n)/(2\epsilon)$, and $n = r^{\epsilon}$ for any $1 > \epsilon > .5$, $\delta > 0$; or $pr < (\gamma-1)\ln(n)/(2\gamma)$, and $n = r^{\gamma}$ for any $\gamma > 1$. That is, we present an algorithm which usually runs in polynomial average time when null clauses are present in random instances with high probability. In fact, we show that the good average performance is due *only* to the occurrence of null clauses in random inputs. Thus, the results are more a study of the model itself rather than an analysis of a practical algorithm.

2 The Algorithm

Let a variable which appears exactly once in an instance be called a *unit* variable. Let a variable which appears at least twice in an instance be called a *serious* variable. We consider the following algorithm for solving instances of Satisfiability:

$NULL(I)$:

1. If I has a null clause then return "unsatisfiable"
2. Otherwise,
   a. Set all unit variables to satisfy the clauses they occupy
   b. for all truth assignments t to serious variables in I, if t satisfies I then return "satisfiable"
3. Return "unsatisfiable"

In step (2b) $NULL$ terminates as soon as the first satisfiable truth assignment is discovered. It should be clear that $NULL$ returns "satisfiable" if and only if I is satisfiable.

3 The Analysis

To simplify the analysis, we show that the expected number of *steps* executed in $NULL$ is bounded by a polynomial in n under several conditions. Since the complexity of each step is polynomially bounded, the average running time of $NULL$ must then be polynomially bounded under those conditions as well.

Let $I_{=}(x)$ denote the event that the input contains exactly x serious variables. Let $I_{\geq}(x)$ denote the event that the input contains at least x serious variables. Let I_{ϕ} denote the event that the input contains a null clause. Let $T(n,r,p)$ denote the average number of steps executed by $NULL$ given that instances are generated according to model $M(n,r,p)$. Then, since the number of steps required by exhaustive search on an input with exactly x serious variables is at most 2^x, we can write

$$\begin{aligned}
T(n,r,p) &\leq Pr(I_{\phi}) + \sum_{x=1}^{r} 2^x \cdot Pr(\bar{I}_{\phi} \wedge I_{=}(x)) \\
&\leq 1 + 2Pr(\bar{I}_{\phi} \wedge I_{\geq}(1)) + \sum_{x=1}^{r-1} 2^x \cdot Pr(\bar{I}_{\phi} \wedge I_{\geq}(x+1)) \\
&\leq 3 + \sum_{x=1}^{\lfloor 3.82\mu \rfloor} 2^x \cdot Pr(\bar{I}_{\phi}) + \sum_{x=\lfloor 3.82\mu \rfloor+1}^{r-1} 2^x \cdot Pr(I_{\geq}(x+1)) \\
&= 3 + \sum_{x=1}^{\lfloor 3.82\mu \rfloor} 2^x \cdot Pr(\bar{I}_{\phi}) + \sum_{x=\lfloor 3.82\mu \rfloor+2}^{r} 2^{x-1} \cdot Pr(I_{\geq}(x)) \qquad (1)
\end{aligned}$$

where μ is the mean number of serious variables in an instance.

First, we obtain a bound on the second sum in (1). Since variables are placed independently in clauses, the number of serious variables in an instance is binomially distributed. By the Chernoff bound for binomial distributions [1], $Pr(I_{\geq}((1+\beta)\mu)) < e^{-\beta^2\mu/3}$, $\beta > 0$. Thus,

$$\sum_{x=\lfloor 3.82\mu \rfloor+2}^{r} 2^{x-1} \cdot Pr(I_{\geq}(x)) \leq \sum_{x=\lfloor 3.82\mu \rfloor+2}^{r} 2^x e^{-(x/\mu-1)^2\mu/3}$$

First, suppose $n = r^\epsilon$, $1 > \epsilon > .5$, and $pr \leq (1-\epsilon-\delta)\ln(n)/(2\epsilon)$ for any $(1-\epsilon)^2 > \delta > 0$. Then

$$pn = pr^\epsilon = pr \cdot r^{\epsilon-1} \leq (1-\epsilon-\delta)\ln(n)r^{\epsilon-1}/(2\epsilon).$$

But, $r^{\epsilon-1} = n^{(\epsilon-1)/\epsilon}$. Therefore, $pn \leq (1-\epsilon-\delta)\ln(n)n^{(\epsilon-1)/\epsilon}/(2\epsilon)$. This implies

$$\begin{aligned} \ln(2)(7.64(pn)(pr)+1/n) &\leq \ln(2)(7.64(1-\epsilon-\delta)^2\ln^2(n)n^{(\epsilon-1)/\epsilon}/(2\epsilon)^2 + n^{-1}) \\ &\leq (1-\delta)n^{(\epsilon-1+\delta)/\epsilon} + n^{-1} \end{aligned} \quad (5)$$

for large n. Since $1 > \epsilon > .5$ and $(1-\epsilon)^2 > \delta > 0$, $(\epsilon-1)/\epsilon > -1$ and (5) is less than $n^{(\epsilon-1+\delta)/\epsilon}$ in the limit. But $e^{-2pr} \geq e^{-(1-\epsilon-\delta)\ln(n)/\epsilon} = n^{(\epsilon-1+\delta)/\epsilon}$ so (4) is satisfied.

Now, suppose $n = r^\epsilon$, $.5 \geq \epsilon > 0$, and $pr \leq (1-\delta)\ln(n)/2$. Then, proceeding as above, $\lim_{n\to\infty} \ln(2)(7.64(pn)(pr)+1/n) < 6n^{-1}$. But, in the limit, $6n^{-1} < n^{\delta-1} = e^{-(1-\delta)\ln(n)} \leq e^{-2pr}$ satisfying (4).

The remaining case, $pn \to 0$ and $pr < \gamma$, is straightforward. □

Theorem 2 *NULL runs in polynomial average time if $n/r = \beta$, where β is a constant greater than 0, and $2.64(1-(1-p)^{2\beta r}(1+2\beta pr)) < \beta e^{-2pr}$.*

Proof:

Since $p < 1$, $1/(1-p) > 1$. Then $\mu = (1-(1-p)^{2n}(1+2pn/(1-p)))r \leq (1-(1-p)^{2n}(1+2pn))r$. Thus, (3) is polynomially bounded if

$$\begin{aligned} -ne^{-2pr} + ln(2)(3.82(1-(1-p)^{2n}(1+2pn)))r &\leq \ln(n) \iff \\ -\beta e^{-2pr} + 2.64(1-(1-p)^{2\beta r}(1+2\beta pr)) &\leq \ln(n)/r. \end{aligned} \quad (6)$$

The theorem follows. □

According to Theorem 2, $NULL$ has polynomial average time if $2pr < \ln(\beta) - \ln(2.64)$ (this is fairly tight if β is large). If $\beta = 1$ then $NULL$ has polynomial average time if $pr < .37$.

Theorem 3 *NULL runs in polynomial average time if $n = r^\gamma$, $\gamma > 1$, and $pr < (\gamma-1)\ln(n)/(2\gamma)$.*

Proof:

Sum (2) is bounded from above by $2^r(1-(1-p)^{2r})^n < e^{-ne^{-2pr}+\ln(2)r}$. If $n = r^\gamma$ and $pr < 1$ then $-ne^{-2pr} + \ln(2)r < -r^\gamma e^{-2} + \ln(2)r \to -\infty$. Hence, the sum (2) is polynomially bounded. Now suppose $n = r^\gamma$ and $1 < pr < (\gamma-1)\ln(n)/(2\gamma)$. Then $e^{-ne^{-2pr}+\ln(2)r}$ is polynomially bounded if $r < ne^{-2pr}/\ln(2)$. This is satisfied if $r < r^\gamma e^{-2pr}/\ln(2)$. This is equivalent to $\ln(2) < r^{\gamma-1}e^{-2pr}$. But this is satisfied by the hypothesis because $e^{-2pr} > n^{-1+1/\gamma} = r^{-(\gamma-1)}$. □

5 Acknowledgements

Thanks to Paul W. Purdom for suggesting this problem.

References

[1] **Angluin, D., and Valiant, L. G.,** "Fast probabilistic algorithms for Hamiltonian circuits and matchings", *Journal of Computer and System Sciences*, Vol. 18, (1979) pp.155-193.

[2] **Bugrara, K., Pan, Y., and Purdom, P. W.,** "Exponential average time for the pure literal rule", *SIAM J. Comput.*, Vol. 18, No. 2, (1989) pp. 409-418.

[3] **Franco, J.,** "On the probabilistic performance of algorithms for the Satisfiability problem", *Information Processing Letters*, Vol. 23, (1986) pp. 103-106.

[4] **Purdom, P. W., and Brown, C. A.,** "The pure literal rule and polynomial average time", *SIAM J. Comput.*, Vol. 14, No. 4, (1985) pp. 943-953.

[5] **Purdom, P. W., and Brown, C. A.,** "Polynomial average time Satisfiability problems", *Information Sciences*, Vol. 41, (1987) pp. 23-42.