TECHNICAL REPORT NO. 294

Elimination of Infrequent Variables Improves Average Case Performance of Satisfiability Algorithms

by

John Franco

October, 1989

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY

Bloomington, Indiana 47405-4101

Elimination of Infrequent Variables Improves Average Case Performance of Satisfiability Algorithms[*][†]

John Franco
Department of Computer Science,
Indiana University, Bloomington Indiana 47405

October 13, 1989

Abstract

We consider pre-processing a random instance I of CNF Satisfiability in order to remove infrequent variables (those which appear once or twice in an instance) from I. The model used to generate random instances is the popular random-clause-size model with parameters n, the number of clauses, r, the number of Boolean variables from which clauses are composed, and p, the probability that a variable appears in a clause as a positive (or negative) literal. It is shown that exhaustive search over such pre-processed instances runs in polynomial average time over a significantly larger parameter space than has been shown for any other algorithm under the random-clause-size model when $n = r^{\epsilon}$, $\epsilon < 1$, and $pr < \sqrt{\epsilon r \ln(r)}$. Specifically, the results are that random instances of Satisfiability are "easy" in the average case if $n = r^{\epsilon}$, $2/3 > \epsilon > 0$, and $pr < r^{2/3-\epsilon}$; or $n = r^{\epsilon}$, $1 > \epsilon \geq 2/3$, $pr < (1 - \epsilon - \delta)\ln(n)$ for any $\delta > 0$; or $pn \to 0$, $pr < \gamma \ln\ln(n)$ for any $\gamma > 0$.

---

[*]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.

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 for a pair of complementary literals (associated with the same variable) to be present in a clause. It is also possible to generate an empty (or null) clause using this model. If an instance contains a null clause it is trivially unsatisfiable. The preponderance or absence of null clauses in random instances is controlled by the product pr which is half the average number of literals in a clause. From [3] a random instance posseses a null clause with probability tending to 1 if the product $pr < \ln(n)/2$.

In the literature, polynomial average time results for Satisfiability algorithms are known only if $n = r^\epsilon$, $1 \geq \epsilon > 0$, $pr < \sqrt{\ln(n)} \cdot r^{.5-\epsilon}$ [6]; or $n = r^\gamma$, $\gamma > 1$, $pr < (\gamma - 1)\ln(n)/(2\gamma)$ [7]; or $n = \beta r$, β a positive constant, and $pr < f(\beta)$ [4]; or $pr > \sqrt{r \ln(n)}$ [5]. Furthermore, no polynomial average time results are published for the case $\lim_{n,r\to\infty} n/\sqrt{r} = \infty$ and $\ln(n)/2 < pr < \sqrt{r\ln(n)}$; 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 $pr > \sqrt{\omega(r)\ln(n)} \cdot r^{.5-\epsilon}$ 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 no published polynomial average time analysis exists and at least one non-trivial algorithm requires superpolynomial average time. This philosophically uncomfortable situation has recently been improved by a result in [4] which shows that exhaustive search, only if no null clauses are present, has polynomial average time if $n = r^\epsilon$, $1 > \epsilon > .5$, and $pr < (1 - \epsilon - \delta)\ln(n)/(2\epsilon)$ for any $(1-\epsilon)^2 > \delta > 0$. Thus, random instances of Satisfiability are generally easy in the average case when null clauses are present with high probability.

In this paper we extend significantly the parameter space over which polynomial average time results are known. This is accomplished by using substitution rules to eliminate clauses containing infrequent variables: that is, variables occurring only once or twice in an instance. Thus, infrequent variables are also eliminated by applying these rules. The results of this paper show that exhaustive search over the remaining variables is speeded up considerably. The idea seems to be generalizable and may represent the first of a family of such results that will take care of a large portion of the remaining parameter space for which polynomial average time results are not now known. The result of such a generalization, to the extent that it is possible, is apparent from the analysis presented here. Specifically, the results of this paper are that random instances of Satisfiability are "easy" in the average case if $n = r^\epsilon$, $2/3 > \epsilon > 0$, and $pr < r^{2/3-\epsilon}$; or $n = r^\epsilon$, $1 > \epsilon \geq 2/3$, and $pr \leq (1 - \epsilon - \delta)\ln(n)$ for any $\delta > 0$; or $pn \to 0$, and $pr < \gamma \ln\ln(n)$ for any $\gamma > 0$; or $n = \beta r$, β a constant, and $pr < g(\beta)f(\beta)$ where $g(\beta) > 1$ for all β. The first of these results does not depend on the presence of null clauses in an instance and is significant for this reason.

2 The Algorithm

For convenience, we write a clause as a tuple of the literals it contains. For example, the clause $(x \vee y \vee z)$ is written as (x, y, z). Similarly, we write the conjunction of two clauses $C_1 \wedge C_2$ as C_1, C_2.

Let a variable which appears exactly once in an instance I be called a *unit* variable. Let a variable which appears exactly twice in I be called a *double* variable. Let a variable which appears at least three times in I be called a *serious* variable. The table below defines substitutions for clauses in I containing unit and double variables. In the table we use v to denote a positive literal taken from a unit or double variable, $\bar{v}$ a negative literal so taken, and x and y either a positive or negative literal which is not necessarily taken from a unit or double variable.

var type	substitution name	occurrence	replacement
unit	unit elimination	$(v, x, \ldots)$	*true*
		$(\bar{v}, x, \ldots.)$	
double	double elimination	$(v, v, x, \ldots)$	
		$(\bar{v}, \bar{v}, x, \ldots)$	
	trivial elimination	$(v, \bar{v}, x, \ldots)$	
	pure literal rule	$(v, x, \ldots), (v, y, \ldots)$	
		$(\bar{v}, x, \ldots), (\bar{v}, y, \ldots)$	
	resolution	$(v, x, \ldots), (\bar{v}, y, \ldots)$	$(x, \ldots, y, \ldots)$

When we say *apply unit elimination* we mean, according to the table above, look for a clause containing a unit variable v and replace it with the logical value *true*; if no such clause exists do nothing. Similar statements hold for applying any of the other substitution rules listed in the table. It is possible that, after repeated applications of double-variable substitution rules, some double variables will occur only once in I. By *clean up double variables* we mean eliminate all clauses containing double variables that appear once in I.

Consider the following algorithm for solving instances of Satisfiability:

```
$INFREQ(I)$:

1.   If $I$ has a null clause then return "unsatisfiable"
2.   Otherwise,
  a.   repeatedly apply double variable substitution rules in order until opportunities vanish
  b.   clean up all remaining double variables
  c.   repeatedly apply unit elimination until opportunities vanish
  d.   for all truth assignments $t$ to serious variables in $I$, if $t$ satisfies $I$ then return "satisfiable"
3.   Return "unsatisfiable"
```

In step (2d) $INFREQ$ terminates as soon as the first satisfiable truth assignment is discovered. It should be apparent that the size of I is not increased by the application of $INFREQ$ to I. It should also be apparent that all unit and double variables are eliminated from I in steps (2a), (2b), and (2c) of $INFREQ$ (these are the pre-processing steps). Thus, in step (2d), the truth assignment t is an assignment to all variables which appear in the processed I.

Lemma 1 *$INFREQ$ returns "satisfiable" if and only if I is satisfiable.*

Proof:

Suppose $INFREQ$ returns "unsatisfiable". This can happen only if there is no truth assignment t which satisfies all clauses of I in step (2d). But I in step (2d) contains a subset of the given input and clauses formed by resolution. Hence, no t can satisfy the original input.

Now suppose $INFREQ$ returns "satisfiable". Let t be the truth assignment to serious variables that satisfies all clauses in I of step (2d). We extend t to an assignment which satisfies the given instance I. If clause $(v, x, ...)$, alternatively $(\bar{v}, x, ...)$, was replaced by *true* using unit elimination then set v to true, alternatively, $false$, and the clause is satisfied. Similarly for $(v, v, x, ...)$, alternatively $(\bar{v}, \bar{v}, x...)$, using double elimination. A clause eliminated using trivial replacement is trivially satisfied by any truth assignment. If clauses $(v, x, ...)$ and $(v, y, ...)$, alternatively $(\bar{v}, x, ...)$ and $(\bar{v}, y, ...)$ were replaced by *true* using the pure literal rule then set v to *true*, alternatively $false$, and the clauses are satisfied. If clause $(x, ..., y, ...)$ was the result of the resolution of $(v, x, ...)$ and $(\bar{v}, y, ...)$ and if t satisfies one of $x, ...$, alternatively $y, ...$, then set v to $false$, alternatively *true*. If one or both clauses are themselves a result of some resolution then repeat, using the extended t, until reaching original clauses. If a clause was cleaned up, set its double variable to satisfy the clause. By proceeding as above a truth assignment satisfying the original instance is obtained. □

3 The Analysis

To simplify the analysis, we show that the expected number of iterations in step (2d) of $INFREQ$ is bounded by a polynomial in n under several conditions. Since the complexity of each step is polynomially bounded, the average running time of $INFREQ$ must then be polynomially bounded under those conditions as well. The equations and inequalities below up to (3) are taken from [4] and repeated here for convenience.

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 $INFREQ$ 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,

$$\begin{aligned}
\sum_{x=\lfloor 3.82\mu \rfloor+2}^{r} 2^{x-1} \cdot Pr(I_\geq(x)) &\leq \sum_{x=\lfloor 3.82\mu \rfloor+2}^{r} 2^x e^{-(x/\mu-1)^2\mu/3} \\
&= \sum_{x=\lceil 3.82\mu \rceil+1}^{r} e^{x\ln(2)-x^2/(3\mu)+2x/3-\mu/3} \\
&\leq \sum_{x=\lceil 3.82\mu \rceil+1}^{r} 1 \quad \leq r.
\end{aligned}$$

Next, we obtain an upper bound on the first sum in (1). The probability that a clause is null is $(1-p)^{2r}$. Hence, the probability that all clauses are not null is $Pr(\bar{I}_\phi) = (1-(1-p)^{2r})^n$. Thus, we write

$$\begin{aligned}
\sum_{x=1}^{\lfloor 3.82\mu \rfloor} 2^x \cdot Pr(\bar{I}_\phi) &= Pr(\bar{I}_\phi) \sum_{x=1}^{\lfloor 3.82\mu \rfloor} 2^x = (1-(1-p)^{2r})^n \sum_{x=1}^{\lfloor 3.82\mu \rfloor} 2^x \qquad (2) \\
&\leq (1-(1-p)^{2r})^n 2^{\lfloor 3.82\mu \rfloor+1} \leq e^{-ne^{-2pr}} 2^{3.82\mu+1} \\
&= e^{-ne^{-2pr}+\ln(2)(3.82\mu+1)}. \qquad (3)
\end{aligned}$$

Now, we compute μ and obtain upper bounds on (3). The probability that a variable is not in a particular clause is the probability that neither literal associated with the variable is in that clause and is equal to $(1-p)^2$. Since clauses are independently chosen, the probability that a variable is not in a given instance is $(1-p)^{2n}$, the probability that a variable appears once in an instance is $2pn(1-p)^{2n-1}$, and the probability that a variable appears twice in an instance is $\binom{2n}{2}p^2(1-p)^{2n-2}$. Therefore, the probability that a variable is a serious variable is $1-(1-p)^{2n}-2pn(1-p)^{2n-1}-n(2n-1)p^2(1-p)^{2n-2}$ which may be reduced to $1-(1-p)^{2n}(1+2pn/(1-p)+2(pn)^2(1-1/n)/(1-p)^2)$.

Theorem 1 *INFREQ runs in polynomial average time if $n = r^\epsilon$, $\epsilon \le 2/3$, and $pr \le r^{2/3-\epsilon}$, or if $n = r^\epsilon$, $1 > \epsilon > 2/3$, and $pr < (1-\epsilon-\delta)\ln(n)$, for any $\delta > 0$, or if $pn \to 0$ and $pr < \gamma \ln\ln(n)$ for any $\gamma > 0$.*

Proof:

If $n = r^\epsilon$, $\epsilon < 2/3$, and $pr \le r^{2/3-\epsilon}$ then $pn = pr \cdot r^{\epsilon-1} \le r^{2/3-\epsilon+\epsilon-1} = r^{-1/3} \to 0$. If $1 > \epsilon \ge 2/3$ and $pr \le \ln(n)$ then $pn = pr \cdot r^{\epsilon-1} \le \epsilon\ln(r)r^{\epsilon-1} \to 0$. So, we assume $pn \to 0$. Then $1-(1-p)^{2n}(1+2pn/(1-p)+2(pn)^2(1-1/n)/(1-p)^2) = (4/3)(np)^3 + O((np)^4)$. From now on we ignore the small term for simplicity. Since variables are placed independently in clauses, the number of serious variables in an instance is binomially distributed with parameters r and $(4/3)(np)^3$. Thus, the mean number of serious variables in an instance is $\mu = (4/3)(np)^3 r$. Substituting into (3) gives $e^{-ne^{-2pr}+\ln(2)(5.093(np)^3r+1)}$ which is polynomially bounded if $e^{-ne^{-2pr}+\ln(2)(5.093(np)^3r)}$ is. Therefore, we require

$$-e^{-2pr} + 3.53(pn)^2(pr) \quad \le \quad \ln(n)/n. \tag{4}$$

Let $n = r^\epsilon$, $1 > \epsilon > 0$. Then, after rearranging, (4) becomes

$$\begin{aligned} 3.53(pr)^3 r^{2(\epsilon-1)} &\le e^{-2pr} + \epsilon r^{-\epsilon}\ln(r) \iff \\ 3.53(pr)^3 r^{3\epsilon-2} &\le r^\epsilon e^{-2pr} + \epsilon\ln(r) \end{aligned} \tag{5}$$

Let $\epsilon \le 2/3$ and $pr = r^\alpha$, α a constant. Then (5) becomes

$$3.53 r^{3\alpha+3\epsilon-2} \quad \le \quad e^{-2r^\alpha} r^\epsilon + \epsilon\ln(r). \tag{6}$$

Clearly, (6) is satisfied if $3\alpha + 3\epsilon - 2 \leq 0$ or $\alpha \leq 2/3 - \epsilon$. Thus, if $\epsilon \leq 2/3$ and $pr < r^{2/3-\epsilon}$ then (2) is polynomially bounded.

Now let $1 > \epsilon > 2/3$ and $pr = \alpha \ln(r)$. Then (5) becomes

$$3.53(\alpha \ln(r))^3 r^{3\epsilon-2} \quad \leq \quad r^{\epsilon-2\alpha} + \epsilon \ln(r). \tag{7}$$

Inequality (7) is satisfied if $3\epsilon - 2 \leq \epsilon - 2\alpha - 2\delta$ for any positive constant δ and this is satisfied if $\alpha \leq 1 - \epsilon - \delta$. Thus, if $1 > \epsilon > 2/3$ and $pr \leq (1 - \epsilon - \delta)\ln(r)$ then (2) is polynomially bounded.

The remaining case, $pn \to 0$ and $pr < \gamma \ln\ln(n)$, is straightforward. $\square$

We make two remarks about the proof of Theorem 1. First, in (6) the term $e^{-2r^{\alpha}} r^{\epsilon}$ is due to the presence of null clauses in I. But this term is ignored when determining that $\alpha \leq 2/3 - \epsilon$ in the sentence following (6). Thus, the polynomial average time result for $n = r^{\epsilon}$, $\epsilon < 2/3$, is *not* due to the presence of null clauses in I.

The second remark concerns the scope of infrequent variables. From the paragraph preceeding Theorem 1 it should be evident that, if $pn \to 0$ and only unit variables are eliminated in $INFREQ$, then $\mu = \theta((np)^2 r)$ and, up to constant factors, (6) becomes

$$r^{2\alpha+2\epsilon-1} \leq e^{-2r^{\alpha}} r^{\epsilon} + \epsilon \ln(r)$$

which is satisfied if $\alpha < .5 - \epsilon$. If we could use substitution rules to eliminate triple variables, those which appear three times in an instance, then $\mu = \theta((np)^4 r)$ and $\alpha < (3/4) - \epsilon$. If we could eliminate all variables occurring i or fewer times in I then we would have $\mu = \theta((np)^{(i-1)} r)$ and polynomial average time if $pr < r^{i/(i+1)-\epsilon}$, $\epsilon < i/(i+1)$. Clearly i does not have to be very large to make a major impact on the parameter space supporting polynomial average time. Unfortunately, trying to eliminate even triple variables can cause an exponential explosion of the size of I. In this event the assumption that the complexity of each step of $INFREQ$ is polynomially bounded is not valid. We ask: are explosions so infrequent that they do not significantly affect average time performance? An affirmative answer would have a major impact on polynomial average time results under the random-clause-size model. We leave investigation of this question for a future paper.

The next theorem shows where $INFREQ$ runs in polynomial average time when $n = \beta r$, β a positive constant.

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

Proof:

Since $p < 1$, $1/(1-p) > 1$ and $1/(1-p)^2 > 1$. Then

$$\mu = (1-(1-p)^{2n}(1+2pn/(1-p)+2(pn)^2/(1-p)^2))r \le (1-(1-p)^{2n}(1+2pn+2(pn)^2))r.$$

Thus, (3) is polynomially bounded if

$$\begin{aligned} -ne^{-2pr} + ln(2)(3.82(1-(1-p)^{2n}(1+2pn+2(pn)^2)))r &\le \ln(n) \iff \\ -\beta e^{-2pr} + 2.64(1-(1-p)^{2\beta r}(1+2\beta pr+2(\beta pr)^2)) &\le \ln(n)/r. \end{aligned} \tag{8}$$

The theorem follows. □

According to Theorem 2, $INFREQ$ has polynomial average time if $2pr < \ln(\beta) - \ln(2.64)$ (this is fairly tight if β is large). If $\beta = 1$ then $INFREQ$ has polynomial average time if $pr < .59$.

4 Conclusions

We have investigated a simple strategy for solving instances of CNF Satisfiability with respect to average case performance. The important idea is the elimination of infrequent variables before applying, in this case, exhaustive search. We have shown that this strategy is superior in average case performance to all other algorithms analyzed under the random-clause-size model when $pr < \sqrt{\epsilon r \ln(r)}$, $n < r^\epsilon$, and $\epsilon < 2/3$. The strategy may be generalizable, to some extent, and the analysis seems to suggest the outcome of an investigation of such a generalization.

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] **Franco, J.,** "On the occurrence of null clauses in random instances of Satisfiability", Technical Report No. 291, Computer Science Department, Indiana University (1989).

[5] **Iwama, K.,** "CNF Satisfiability test by counting and polynomial average time", *SIAM J. Comput.*, Vol. 18, No. 2, (1989) pp. 385-391.

[6] **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.

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