Probabilistic Performance of a Heuristic for the Satisfiability Problem

By

John Franco and Yuan Chuan Ho
Computer Science Department
Indiana University
Bloomington, IN 47405

TECHNICAL REPORT NO. 193

Probabilistic Performance of a Heuristic for the Satisfiability Problem

by

John Franco and Yuan Chuan Ho
Indiana University

May, 1986

KEYWORDS: Satisfiability, Average Analysis, Probalistic Analysis, Davis-Putnam, NP-complete

This material is based on work supported by the Air Fouce Office of Scientific Research under Grant No. AFOSR-84-0372.

This report to appear in Discrete Applied Math.

ABSTRACT

An algorithm for the Satisfiability problem is presented and its probabilistic behavior is analysed when combined with two other algorithms studied earlier. The analysis is based on an instance distribution which is parameterized to simulate a variety of sample characteristics. The algorithm dynamically assigns values to literals appearing in a given instance until a satisfying assignment is found or the algorithm "gives up" without determining whether or not a solution exists. It is shown that if n clauses are constructed independently from r boolean variables where the probability that a variable appears in a clause as a positive literal is p and as a negative literal is p then almost all randomly generated instances of Satisfiability are solved in polynomial time if $p < .4\ln(n)/r$ or $p > \ln(n)/r$ or $p = c\ln(n)/r$, $.4 < c < 1$ and $\lim_{n,r\to\infty} n^{1-c}/r^{1-\epsilon} < \infty$ for any $\epsilon > 0$. It is also shown that if $p = c\ln(n)/r$, $.4 < c < 1$ and $\lim_{n,r\to\infty} n^{1-c}/r = \infty$ then almost all randomly generated instances of SAT have no solution. Thus the combined algorithm is very effective in the probabilistic sense on instances of SAT that have solutions.

1. Introduction

The Satisfiability problem (SAT) is the problem of determining whether a given collection I of disjunctions (clauses) of boolean literals can all be satisfied (have value *true*) by some consistent assignment of truth values to the literals of I (truth assignment). SAT is NP-complete so there is no known worst case efficient algorithm for solving this problem.

However, numerous algorithms for SAT have been shown to solve random instances of SAT efficiently with high probability under certain conditions. Some of these results are based on a parameterized input distribution which we denote by $J(n,r,p)$. According to this distribution a random instance I of SAT consists of n clauses constructed independently from a set V of r variables as follows: for each $v \in V$ and for all $1 \leq i \leq n$ place v into the i^{th} clause of I as a positive literal (that is, v) with probability p, as a negative literal (that is, $\bar{v}$) with probability p and leave v and $\bar{v}$ out of the i^{th} clause with probability $1-2p$. In this paper, both p and n are functions of r but, for the sake of simplicity, we write p and n instead of $p(r)$ and $n(r)$. In [10], [12] and [13] the average running time of several algorithms for SAT is obtained under $J(n,r,p)$. The conditions under which at least one of those algorithms runs in polynomial average time are as follows:

1) $\lim_{r\to\infty} rp = 0,\ \ n \geq r\ln(2)/-\ln((r+1)p)$.
2) $\lim_{r\to\infty} rp = \infty,\ \ \lim_{r\to\infty} p = 0,\ \ n \geq \ln(2)e^{2rp}/ep$.
3) $\lim_{r\to\infty} p = 0,\ \ np \leq \sqrt{\frac{c\ln(r)}{r}}$, c constant.
4) $\lim_{r\to\infty} 1/p =$ polynomial(r), $np \leq r^{cp}$, c constant.
5) $n \leq c\ln(r)$, c constant.

In [6] it was shown that two trivial, polynomial time algorithms nearly always solve random instances of SAT generated according to $J(n,r,p)$ under conditions which subsume 1), 2) and 4) above. Specifically, consider the following two algorithms:

```
$A_1(I)$ :
    Construct a random truth assignment $t$ to the variables of $I$
    Check whether $t$ satisfies $I$
    If $t$ satisfies $I$ then return($t$)
    Else return("give up")
```

```
$A_2(I)$ :
    For all clauses $c \in I$
        If $c$ contains no literals then return('no solution possible')
    Return("give up")
```

In A_1 a random truth assignment is found by choosing the value *true* for each variable with probability 1/2 (consequently the value *false* with probability 1/2) independently of the assignment of values to other variables. Clearly, generating and checking a truth assignment can be accomplished in polynomial time and if a truth assignment t is returned by $A_1(I)$ then t satsifies I. Clearly, A_2 runs in polynomial time. Since no truth assignment can satisfy a null clause, if $A_2(I)$ returns the expression "no solution possible" then I is not satisfiable. Thus the collection A_1 and A_2 solves instance I of SAT in polynomial time if and only if both do not "give up". In [6] it was shown that A_1 gives up with probability tending to 0 under $J(n,r,p)$ if $p \geq \ln(n)/r$. It was also shown in [6] that A_2 gives up with probability tending to 0 under $J(n,r,p)$ if 1) $p \leq .4\ln(n)/r$ and $n < 2^r$ or 2) $p \leq \ln(n)/(2r)$ and n and r are polynomially related. If $n \geq 2^r$ then exhaustive search will solve instances of SAT in polynomial time so we won't consider this case here. Thus A_1 and A_2 collectively are a probabilistically effective method for solving SAT under $J(n,r,p)$ when $p \geq \ln(n)/r$ or $p \leq \ln(n)/(2r)$ and n and r are polynomially related or $p \leq .4\ln(n)/r$. One interpretation of this result is that random instances of SAT generated according to $J(n,r,p)$ are trivial over the range of p indicated in the previous sentence. Note that condition 3) above becomes

3a) $\lim_{r\to\infty} p = 0, \quad n\ln(n) \leq c\sqrt{r\ln(r)}, \quad c$ constant

in the range $.4\ln(n)/r < p < \ln(n)/r$ and 3a) subsumes 5) over that range of p.

Figure 1 shows the relationships between p, r and n for which random instances of SAT generated according to $J(n,r,p)$ are known to be solved in polynomial time with probability tending to 1 by some previously analyzed algorithm. Also shown in this figure is a line marked "SAT BOUNDARY" which divides the parameter space into two regions such that if the parameters are set to values that correspond to a point to the left of the line then almost all instances generated are unsatisfiable (more explanation will be given at the end of section 4). The unbounded region bordered by lines **I** on the left and **II** on the right corresponds to parameter settings that generate instances which are not solved in polynomial time, almost always, by any previously considered algorithm if $n\ln(n) > c\sqrt{r\ln(r)}$, c constant. In this paper we investigate the question: how hard are the instances in this region? That is, how hard are the instances generated when $.4\ln(n)/r < p < \ln(n)/r$ and $n\ln(n) > c\sqrt{r\ln(r)}$? Are the instances in this range of p and n solved trivially in some other sense? Or, are these instances so hard that no alogrithm which performs well in some probabilistic sense on these instances exists? Or, are there three regions of values for p such that in one region trivial instances are predominantly generated, in the second region non-trivial instances are generated but these can be solved in probabilistic polynomial time by non-trivial algorithms, and in the third region hard instances are predominantly generated? These questions are answered, in part, by the results presented in this paper. We consider the probabilistic performance, under $J(n,r,p)$ and in the range $.4\ln(n)/r < p < \ln(n)/r$, of an algorithm based on the Davis-Putnam Procedure.

The Davis-Putnam Procedure (DPP) [5] is a well known, much studied method for solving instances of SAT and is the basis of most algorithms for SAT. During execution of DPP truth values are assigned to variables sequentially. Each assignment results in some satisfied clauses and some falsified literals within clauses that are not satisfied. A clause which is not satisfied by the current partial assignment and contains exactly one literal that has not been falsified is called a unit clause. An unassigned literal whose complement does not appear in any unsatisfied clause is called a pure literal. In expressing DPP it is convenient to regard clauses to be sets of non-falsified literals and instances to be multisets of clauses. Also, if v is a literal (positive or negative) it is convenient to use the notation $comp(v)$ to mean the literal which is complementary to v. Let $L = \{v_1, v_2, \ldots, v_r, \bar{v}_1, \bar{v}_2, \ldots, \bar{v}_r\}$ be a set of $2r$ literals from which clauses are composed initially and let I be a collection of clauses. The Davis-Putnam procedure is stated as follows:

```
$DPP(I, L)$:
    If $I = \phi$ then return("satisfiable")
    If $\exists c \in I$ such that $c = \phi$ then return("unsatisfiable")
    While there is a unit clause $\{v\}$ in $I$
        $I \leftarrow \{c - \{comp(v)\} : c \in I \text{ and } v \notin c\}$
        $L \leftarrow L - \{v, comp(v)\}$
    While there is a pure literal $v$ in $I$
        $I \leftarrow \{c : c \in I \text{ and } v \notin c\}$
        $L \leftarrow L - \{v, comp(v)\}$
    Choose a literal $v$ from $L$
    $I_1 \leftarrow \{c - \{comp(v)\} : c \in I \text{ and } v \notin c\}$
    $I_2 \leftarrow \{c - \{v\} : c \in I \text{ and } comp(v) \notin c\}$
    $L \leftarrow L - \{v, comp(v)\}$
    If $DPP(I_1, L)$="satisfiable" or $DPP(I_2, L)$="satisfiable"
        Then return("satisfiable") else return("unsatisfiable")
```

The algorithm we analyze in this paper is the following

```
$A_3(I)$:
    While $I \neq \phi$ and $\forall c \in I,\ c \neq \phi$
        If there is a unit clause $\{u\} \in I$ then $v \leftarrow u$
        Else choose a literal $v$ randomly from $L$
        $I \leftarrow \{c - \{comp(v)\} : c \in I$ and $v \notin c\}$
        $L \leftarrow L - \{v, comp(v)\}$
    If $I = \phi$ then return("satisfiable")
    Else return("give up")
```

Implicit in both DPP and A_3 is the assignment of value *true* to literal v and, therefore, the assignment of *false* to $comp(v)$. DPP and A_3 differ in that literals are never given more than one value during execution of A_3 but literals may be assigned the values *true* and *false* at different points during execution of DPP; that is, DPP contains a backtracking component whereas A_3 does not. Thus, A_3 is a polynomial time algorithm whereas DPP requires exponential time on some inputs [9]. However, A_3 is not guaranteed to find a truth assignment (implicitly) which satisfies a given instance of SAT if one exists. But, if A_3 does not "give up" then the truth assignment found implicitly by A_3 satisfies the instance input to A_3. Another difference between A_3 and DPP is that A_3 does not regard pure literals as special.

In this paper we show that $A_1(I)$, $A_2(I)$ and $A_3(I)$ run concurrently on a random instance I of SAT generated according to $J(n,r,p)$ "give up" with probability tending to 0 if $n < 2^r$ and either

1) $p \leq .4\ln(n)/r$ or
2) $p \geq \ln(n)/r$ or
3) $p = c\ln(n)/r$, $.4 < c < 1$ and $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r = 0$.

2. Other Probabilistic Results for SAT

In addition to the results stated in the introduction, a number of probabilistic results on algorithms for SAT have been obtained under a constant-clause-size model which we refer to as $M(n,r,k)$. Under $M(n,r,k)$ a random instance of SAT contains n clauses selected uniformly, independently and with replacement from $Q_k(r)$ where $Q_k(r)$ is the set of all possible clauses containing exactly k literals taken from r variables and their complements such that no pair of literals in the same clause is complementary. It can easily be shown that if $\lim_{n,r\to\infty} n/r < -\ln(2)/\ln(1-2^{-k})$ then the expected number of satisfying truth assignments is greater than B^n where $B > 1$ and if $\lim_{n,r\to\infty} n/r > -\ln(2)/\ln(1-2^{-k})$ then the expected number of satisfying truth assignments is less than B^n where $B < 1$. Therefore, since k is independent of n and r, the case $\lim_{n,r\to\infty} n/r = a$, where a is a constant, is important in M. Note that $\lim_{n,r\to\infty} n/r = -\ln(2)/\ln(1-2^{-k}) \approx 2^k \ln(2)$ represents a "flip point" in that if the ratio of n to r is greater than the flip point then instances are nearly always unsatisfiable and if the ratio of n to r is less than the flip point then the average number of satisfying truth assignments per instance is exponential in r. If $k = 3$ (then the problem becomes the 3-Satisfiability problem which is still NP-complete) the flip point is at $n/r = 5.19$.

In [3] it is shown that A_3 finds solutions to random instances of SAT under $M(n,r,k)$ with probability bounded from below by a constant if $\lim_{n,r\to\infty} n/r < 2^{k-1}((k-1)/(k-2))^{k-2}/k$. A_3 may be improved (and generalized) if the chosen literal is taken from a clause in I containing the smallest number of literals of all clauses in I instead of randomly if there are no unit clauses in I. The resulting generalization is shown in [3] to find solutions to random instances of SAT under $M(n,r,k)$ with probability bounded from below by a constant if $\lim_{n,r\to\infty} n/r < 3.08 * 2^{k-2}((k-1)/(k-2))^{k-2}/(k+1) - .75$ for $4 \le k \le 40$ and with probability tending to 1 if $\lim_{n,r\to\infty} n/r < 1.845 * 2^{k-2}((k-1)/(k-2))^{k-2}/(k+1) - .75$ for $4 \le k \le 40$. Algorithm A_3 may also be improved by choosing a variable randomly (when there is no unit clause in I) instead of a literal and "assigning" to it the value which satisfies most clauses. In [2] it was shown that this improvement allowed A_3 to find solutions to random instances of SAT under $M(n,r,3)$ with probability bounded from below by a constant if $\lim_{n,r\to\infty} n/r < 2.9$. Without the improvement A_3 has the same kind of performance if $\lim_{n,r\to\infty} n/r < 2.66$ (also in [2]).

Finally, in [11] it is shown that the expected number of branches in analytic tableaux analysis in propositional calculus is exponential in the number of occurrences of the connectives *and* and *or* when instances are generated equally likely and are such that *and*, *or* and *not* are the only connectives and negation is applied only to atomic formulas.

3. Analysis of A_3 under $J(n, r, p)$

In this section it is shown that if instances of SAT are generated according to $J(n, r, p)$ and $n < 2^r$ then the probability that A_1, A_2 and A_3 "give up" tends to 0 as $n, r \to \infty$ if

1) $p > \ln(n)/r$ or
2) $p \leq .4\ln(n)/r$ or
3) $p = c\ln(n)/r$, $.4 < c < 1$, and $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r = 0$.

We already know that A_2 "gives up" with probability tending to 0 if $p \leq .4\ln(n)/r$ and $n < 2^r$. We also know that A_1 "gives up" with probability tending to 0 if $p \geq \ln(n)/r$. Therefore, we need only find a similar result for A_3 in the range $p = c\ln(n)/r$, $.4 < c < 1$, and and $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r = 0$. The following two paragraphs give a rough idea of how the analysis proceeds.

At the start of each iteration of A_3 there is a collection I of clauses to be processed. During each iteration of algorithm A_3 a literal is chosen, clauses in I containing that literal are removed from I and occurrences of the literal which is complementary to the chosen literal are removed from clauses in I. Let $C_i(j)$ be the collection of clauses in I containing exactly i literals at the start of the $j + 1^{st}$ iteration. After the $j + 1^{st}$ literal is chosen there is a flow of clauses into $C_i(j+1)$ and out of $C_i(j)$. The outward flow is the collection of clauses that had contained i literals prior to the $j + 1^{st}$ iteration but either the chosen literal or its complement was one of them. The inward flow is the collection of clauses that had contained $i+1$ literals prior to the $j+1^{st}$ iteration but a literal complementary to the chosen literal was one of them. Clauses in $C_i(j)$ that are also in $C_i(j+1)$ are not included in the flow to $C_i(j+1)$. Since each clause can have a maximum of r literals, there is no inward flow of clauses to $C_r(j)$ for any j. Algorithm A_3 "gives up" only if some clause in I becomes null at some iteration of A_3 or the given instance contains a null clause. A clause c will become null only if c is a unit clause, there is another unit clause in I which contains the literal that is complementary to c and that literal is chosen on some iteration. Thus, if A_3 is to "give up" with low probability, the probability that a pair of unit clauses is complementary must be low for any j and the probability that a null clause exists in the given instance must be low. The latter probability can easily be calculated and shown to be low if $p > \ln(n)/(2r)$. The former probability is low if the average flow of clauses into $C_1(j)$ is less than 1 for all $0 \leq j \leq r$. This is because there is a flow out of $C_1(j)$ of at least 1 whenever there is a clause in $C_1(j)$ so the average number of unit clauses in $C_1(j)$ will be bounded by a constant if the average flow into $C_1(l)$, $0 \leq l \leq j-1$, is less than one clause per iteration. If the average number of clauses in $C_1(j)$ is bounded by a constant then the probability that a complementary pair of unit clauses appears in $C_1(j)$ is bounded from above by a constant. This constant can be made arbitrarily small by appropriately reducing the flow into $C_1(j)$ for all j. Note that if the average number of clauses in $C_1(j)$ is bounded by a constant then the average flow of clauses out of $C_1(j)$ will be very close to one clause per iteration when at least one clause is in $C_1(j)$. Thus, if the average flow into $C_1(j)$ is greater than one clause per iteration for αr iterations where $\alpha > 0$ then the average number of clauses in $C_1(j)$ will increase and the liklihood that at least one complementary pair of unit clauses exists will become high.

We proceed with the analysis of A_3 by developing a set of flow equations for $C_i(j)$ for all $1 \leq i \leq r$ and $0 \leq j \leq r$, solving them, and finding the conditions which guarantee that the average flow into $C_1(j)$ is small for all j provided $.4\ln(n)/r < p < \ln(n)/r$. By making use of some results from queueing theory these are then shown to be the conditions under which A_3 "gives up" with probability bounded from above by a term tending to 0 as $n, r \rightarrow \infty$. The flow equations are based on the following theorem.

Theorem 1:

Given $|C_i(j)| = n_i(j)$, for all $1 \leq i \leq r - j$, the clauses in $C_i(j)$ are distributed according to $M(n_i(j), r-j, i)$ independently of the clauses in $C_l(j)$, $l \neq i$.

Proof:

This is certainly true for the case $j = 0$. Suppose it is true for all $0 \leq j \leq m$. There are two ways the $m + 1^{st}$ literal is chosen in A_3: randomly from $C_1(m)$ if $|C_1(m)| \neq \Phi$ or randomly from the set of unassigned literals. Consider the second case. By hypothesis, if h_i clauses of $C_i(m)$ contain the chosen literal or its complement, the remaining $n_i(m) - h_i$ clauses of $C_i(m)$ are distributed according to $M(n_i(m) - h_i, r - m - 1, i)$. Also, if g_{i+1} clauses of $C_{i+1}(m)$ contain the complement of the chosen literal, stripping the complement of the chosen literal from those clauses results in a set of g_{i+1} clauses distributed according to $M(g_{i+1}, r - m - 1, i)$. Combining the second set of clauses with the remainder of the first set of clauses results in a set of $n(m) - h_i + g_i$ clauses distributed according to $M(n_i(m) - h_i + g_i, r - m - 1, i) = M(n_i(m+1), r - (m+1), i)$. Now consider the case that a literal appearing in a unit clause is chosen randomly from the set of all such literals. There is one unit clause c which contains this literal. The remaining unit clauses are independent of c and therefore the chosen literal. Futhermore, all clauses in $C_i(m)$, $m > 1$, are independent of c. Hence the previous argument applies. This establishes the result.

We can now develop a set of recurrence relations for the expected number of clauses in $C_i(j)$ for all $1 \leq i, j \leq r$. From the solution to these recurrence relations we will obtain an expression for the expected flow of clauses into $C_1(j)$ for all $1 \leq j \leq r$. Then we will find the conditions which guarantee that this expectation is small enough in the limit. Let

$E\{n_i(j)\}$ be the average number of clauses in $C_i(j)$ at the start of the $j+1^{st}$ iteration of A_3. Let $E\{z_i(j)\}$ be the average number of clauses that flow out from $C_i(j)$ as a result of choosing the $j+1^{st}$ literal. Let $E\{w_i(j)\}$ be the average flow of clauses into $C_i(j+1)$ as a result of choosing the $j+1^{st}$ literal. Then

$$E\{n_i(j+1)\} = E\{n_i(j)\} + E\{w_i(j)\} - E\{z_i(j)\}. \tag{1}$$

Let

$$E\{E\{z_i(j)|n_i(j)\}\} = \sum_{l=0}^{\infty} E\{z_i(j)|n_i(j)=l\} pr(n_i(j)=l).$$

Then, for all $2 \leq i \leq r$

$$E\{z_i(j)\} = E\{E\{z_i(j)|n_i(j)\}\}$$

$$= \sum_{l=0}^{\infty} \frac{i*l}{r-j} pr(n_i(j)=l) = \frac{i*E\{n_i(j)\}}{r-j}$$

because of theorem 1. Also, for all $1 \leq i < r$

$$E\{w_i(j)\} = E\{E\{w_i(j)|n_{i+1}(j)\}\}$$

$$= \sum_{l=0}^{\infty} \frac{(i+1)*l}{2(r-j)} pr(n_{i+1}(j)=l) = \frac{(i+1)*E\{n_{i+1}(j)\}}{2(r-j)}$$

and

$$E\{w_r(j)\} = 0.$$

Therefore (1), for $2 \leq i \leq r$, can be written

$$E\{n_i(j+1)\} = E\{n_i(j)\} + \frac{(i+1)*E\{n_{i+1}(j)\}}{2(r-j)} - \frac{i*E\{n_i(j)\}}{r-j} \tag{2}$$

and

$$E\{n_r(j+1)\} = E\{n_r(j)\} - \frac{r*E\{n_r(j)\}}{r-j}. \tag{3}$$

In order to solve equations (2) and (3) we need the boundary conditions $E\{n_i(0)\}$ for $2 \leq i \leq r$.

Lemma 1:

$$E\{n_i(0)\} = \binom{r}{i}(2p)^i(1-2p)^{r-i}n \qquad i=1,2,\ldots,n$$

Proof:

Since $2p$ is the probability that a literal associated with a particular variable is contained in a particular clause of a given instance I and since literals are placed in clauses independently, the probability that a clause of I has exactly i literals is $\binom{r}{i}(2p)^i(1-2p)^{r-i}$. The desired expectation is the product of the number of clauses in an instance, n, and that probability.

The required bounds on solutions to (2) and (3) with the boundary conditions given in lemma 1 are given in theorem 2. In theorem 2 we use the convention that $\binom{x}{w} = 0$ if $w > x$.

Theorem 2:

For all $.4\ln(n)/r < p < \ln(n)/r$, $0 \le j \le r$, $2 \le i \le r$

$$E\{n_i(j)\} = \binom{r-j}{i}(2p)^i(1-p)^j(1-2p)^{r-i-j}n.$$

Proof:

The proof is by induction. The hypothesis is true for $j = 0$ since $E\{n_i(0)\} = \binom{r}{i}(2p)^i(1-2p)^{r-i}n$ and for $i = r$ since $E\{n_r(j)\} = 0$ for all $1 \le j$. The hypothesis is also true for $i > r - j$ since $E\{n_i(j)\} = 0$ in that range. Now suppose the hypothesis is true for all $a \le i \le r$, $0 \le j \le b$ and $a + 1 \le i \le r$, $b < j \le r - a$ where a is any integer greater than or equal to 2 and less than or equal to r and b is any integer greater than or equal to 0 and less than or equal to $r - a$. We show that this implies it is also true for $i = a$ and $j = b + 1$. By applying the hypothesis to equation (2) we have

$$
\begin{aligned}
E\{n_a(b+1)\} &= \binom{r-b}{a}(2p)^a(1-p)^b(1-2p)^{r-a-b}\left(1-\frac{a}{r-b}\right)n \\
&\quad + \binom{r-b}{a+1}(2p)^{a+1}(1-p)^b(1-2p)^{r-a-b-1}n \\
&= \binom{r-b-1}{a}(2p)^a(1-p)^b(1-2p)^{r-a-b}n \\
&\quad + p\binom{r-b-1}{a}(2p)^a(1-p)^b(1-2p)^{r-a-b-1}n \\
&= \binom{r-b-1}{a}(2p)^a(1-p)^b(1-2p)^{r-a-b-1}\left((1-2p)+p\right)n \\
&= \binom{r-b-1}{a}(2p)^a(1-p)^{b+1}(1-2p)^{r-a-b-1}n.
\end{aligned}
$$

The main result is stated as follows.

Theorem 3:

> The probability that A_1, A_2 and A_3 "give up" when concurrently applied to a random instance of SAT generated according to $J(n,r,p)$ tends to 0 as $n, r \to \infty$ if $n < 2^r$ and either of the following three conditions hold:
>
> 1. $p \leq .4\ln(n)/r$
> 2. $p \geq \ln(n)/r$
> 3. $p = c\ln(n)/r$, $.4 < c < 1$, and $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r = 0$.

Proof:

> Since A_2 does not "give up" if at least one null clause is present in the given instance of SAT and since A_1 and A_2 perform as needed if $p \geq \ln(n)/r$ and $p \leq .4\ln(n)/r$ we need only show that A_3 "gives up" on some iteration with probability tending to 0 if $.4\ln(n)/r < p < \ln(n)/r$ and $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r = 0$. From theorem 2 we have
>
> $$E\{w_1(j)\} = \frac{E\{n_2(j)\}}{r-j} = 2(r-j-1)p^2(1-p)^j(1-2p)^{r-2-j}n. \tag{4}$$
>
> By setting the derivative of (4) to zero we find that $E\{w_1(j)\}$ has a maximum at $j = j_o = r-1-1/(\ln(1-p)-\ln(1-2p))$. Substituting j_o for j in (4) gives
>
> $$E\{w_1(j_o)\} = \frac{2p^2(1-p)^{r-1-\frac{1}{\ln(1-p)-\ln(1-2p)}}(1-2p)^{\frac{1}{\ln(1-p)-\ln(1-2p)}-1}}{\ln(1-p)-\ln(1-2p)}n. \tag{5}$$

Since $\ln(1-p) - \ln(1-2p) = p + O(p^2)$, $p = c\ln(n)/r$, $.4 < c < 1$ and n is not exponential in r we can write (5) as

$$E\{w_1(j_o)\} =$$

$$2c\ln(n)\left(1 - \frac{c\ln(n)}{r}\right)^{r\left(1-\frac{1}{c\ln(n)(1+O(p))}\right)}\left(1 - \frac{2c\ln(n)}{r}\right)^{\frac{r}{c\ln(n)(1+O(p))}-1}\frac{n}{r}.$$

If $\lim_{n,r\to\infty}\ln(r)\ln^2(n)n^{1-c}/r = 0$ we have

$$\lim_{n,r\to\infty}\ln(r)\ln(n)E\{w_1(j_o)\} = 0. \tag{6}$$

The probability that a complementary pair of unit clauses appears during execution of A_3 is less than the sum over all j of the probabilities that a complementary pair appears during iteration j. The probability that a complementary pair appears during iteration j is less than the expected number of complementary pairs generated at iteration j. Therefore the probability that A_3 "gives up" is

$$\leq \sum_{j=0}^{r-1}\frac{E\{w_1^2(j+1)\}/2 + E\{n_1(j) * w_1(j+1)\}}{2(r-j)}. \tag{7}$$

First consider the $E\{n_1(j) * w_1(j+1)\}$ terms. Let ρ be the list of literals chosen during execution of A_3. Let $N_\rho(j)$ be the collection of clauses in the original instance of SAT which become unit clauses and from which literals are chosen because they become unit clauses during the first j literal choices of ρ. Clauses in the original instance of SAT that correspond to clauses that flow into $C_1(j)$ contain an unchosen literal, the complement of the j^{th} chosen literal and possibly some literals which are all complementary to chosen literals. Therefore, the probability, denoted $g_\rho(r,p,j)$, that a particular clause of the original instance of SAT, which is not in $N_\rho(j)$, flows into $C_1(j)$ given ρ is

$$g_\rho(r,p,j) = 2p^2(r-j)(1-2p)^{r-j}(1-p)^j. \tag{8}$$

The number of such clauses is binomially distributed with parameters $n - |N_\rho(j)|$ and $g_\rho(r,p,j)$ since all clauses are constructed independently. Using the Chernoff bound for binomial distributions [14], and realizing that the bound is maximum if $|N_\rho(j)| = 0$, the probability that the number of clauses flowing into $C_1(j)$ is greater than $\ln(n)$ is less than $n^{-\ln(n)}$ if $\lim_{n,r\to\infty}\ln(r)\ln^2(n)n^{1-c}/r = 0$. Therefore

$$E\{n_1(j) * w_1(j+1)\} \leq \ln(n)E\{n_1(j)\} + \frac{n^2}{n^{\ln(n)}}. \tag{9}$$

Proceeding in the same way for the $E\{w_1^2(j)\}$ terms results in

$$E\{w_1^2(j)\} \leq \ln(n)E\{w_1(j)\} + \frac{n^2}{n^{\ln(n)}}. \tag{10}$$

We only need to find $E\{n_1(j)\}$. Since $w_1(0) = 0$, $w_1(r-1) = 0$ and at least one unit clause is removed on any iteration that unit clauses are present, we may use the following inequality and equation for a work conserving, non-preemptive single server queueing system in which each unit of time is an iteration of A_3:

$$E^*\{W\} \leq E^*\{n_1\} + 1 \tag{11}$$

$$E^*\{n_1\} = E^*\{w_1\} * E^*\{W\} \tag{12}$$

Inequality (12) is Little's law and (11) comes from [4]. In this queueing system "serviced jobs" are unit clauses that are removed. The maximum residual "service time" observed by a clause when it becomes a unit clause (enters the system) is one iteration. In (11) and (12) $E^*\{W\}$ is the average number of iterations that a unit clause waits before it is removed if the expected number of unit clauses present on any iteration were equal to the maximum expected number of unit clauses present over all j, $0 \leq j \leq r-1$. $E^*\{n_1\}$ is the maximum expected number of unit clauses present over all j, $0 \leq j \leq r-1$. Finally, $E^*\{w_1\}$ is the maximum expected flow into the collection of unit clauses over all j, $0 \leq j \leq r-1$. Both (11) and (12) are independent of the order in which unit clauses are removed and distribution of $n_1(j)$ and $w_1(j)$. Both (11) and (12) are valid only if $E^*\{w_1\} < 1$ and this is the case if $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r = 0$ because of (6). Combining (11) and (12) gives

$$E^*\{n_1\} \leq \frac{E^*\{w_1\}}{1 - E^*\{w_1\}}. \tag{13}$$

Combining (7), (9), (10) and (13) we get that, in the limit, (7) is less than

$$K\ln(n)E^*\{w_1\}\sum_{j=0}^{r-1}\frac{1}{r-j} \leq K\ln(n)\ln(r)E^*\{w_1\} \tag{14}$$

where K is some constant greater than zero. From (6) we have that (14) tends to 0 as $n, r \to \infty$. Thus, the probability that A_3 "gives up" when $p = c\ln(n)/r$, $.4 < c < 1$ and $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r = 0$ tends to 0 as $n, r \to \infty$.

4. Where A_3 Fails

In the previous section it was shown that A_3 "gives up" with probability tending to 0 if $p = c\ln(n)/r$, $.4 < c < 1$, and $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r = 0$. In this section we show that this result is tight to within the factor $\ln(r)\ln(n)$. That is, we show that A_3 "gives up" with probability tending to 1 if $p = c\ln(n)/r$, $.4 < c < 1$, $\lim_{n,r\to\infty} \ln(n)n^{1-c}/r > e(1+\gamma)/c$, $\gamma > 0$, and $n < 2^{r^{1-\delta}}$ for any $\delta > 0$. We also show that if $p = c\ln(n)/r$, $.4 < c < 1$, and $\lim_{n,r\to\infty} n^{1-c}/r = \infty$ then the probability that a random instance is unsatisfiable tends to 1. Thus A_3 does not give up with probability tending to 1 over almost all relationships between n and r that admit satisfiable instances when $p = c\ln(n)/r$, $.4 < c < 1$.

Recall that

$$E\{w_1(j)\} = 2(r-j-1)p^2(1-p)^j(1-2p)^{r-2-j}n$$

and $E\{w_1(j)\}$ is maximum at $j = j_o = r-1-r/(c\ln(n))$ (the lower order terms are removed to avoid unnecessary clutter). Consider the interval $j_o - r/(2c\ln(n)) < j < j_o + r/(2c\ln(n))$ which we refer to as J_o. It is straightforward to verify that $E\{w_1(j)\} > 1+\gamma$ for any $\gamma > 0$ at the endpoints of J_o if $\lim_{n,r\to\infty} \ln(n)n^{1-c}/r > e^2(1+\gamma)/c$. Since $E\{w_1(j)\}$ has one maximum (at $j = j_o$), if $E\{w_1(j)\} > 1+\gamma$ at both endpoints of J_o then $E\{w_1(j)\} > 1+\gamma$ at all points internal to J_o. Then, for at least $r/(c\ln(n))$ iterations the average flow into the set of unit clauses is greater than $1+\gamma$. Suppose that, on at least $r^{\delta/2}$ of those iterations, more than one clause is eliminated from the set of unit clauses. Since unit clauses are independent, the probability that a set of at least two unit clauses eliminated on some iteration of A_3 contains a complementary pair of literals is at least $1/2$. Therefore the probability that at least one set of unit clauses contains a complementary pair of unit clauses (A_3 gives up in this case) is at least $1-(1/2)^{r^{\delta/2}}$ which tends to 1. Now suppose that more than one clause is eliminated from the set of unit clauses on no more than $r^{\delta/2}$ iterations of A_3. Using the Chernoff bound for binomial distributions, again, we find that the probability that more than $r^{\delta/4}$ unit clauses are eliminated on any iteration is less than $e^{-r^{\delta/2}}$. Then the probability that more than one clause is eliminated from the set of unit clauses on at least one of $r^{\delta/2}$ iterations is less than $r^{\delta/2}e^{-r^{\delta/2}}$ which tends to 0. Thus, the average number of clauses eliminated during the iterations corresponding to J_o is less than $r^{\delta/2}$ with probability tending to 1. This implies that the average number of clauses remaining at the $j = j_u = r-1-r/(2c\ln(n))$ iteration is greater than $r/\ln(n) - r^{\delta/2}$. It is straightforward to show that this and the assumption that $n < 2^{r^{\delta/2}}$ imply the number of clauses in the set of unit clauses is at least $(r/\ln(n))^{3/4}$ in the limit at $j = j_u$ with probability tending to 1. The number of variables not assigned values at that iteration is $r/2\ln(n)$. Since all unit clauses are independent we may use the result of [1] which says that the probability that an instance of 1-SAT (one literal per clause) containing $\hat{n}$

clauses composed from $\hat{r}$ variables is unsatisfiable with probability tending to 1 if $\hat{n} > \sqrt{\hat{r}}$ and conclude that at the $j = j_u$ iteration there is at least one pair of complementary unit clauses (A_3 gives up in this case, also) with probability tending to 1. This argument shows

Theorem 4:

> Let c, γ and δ be any constants such that $.4 < c < 1, \gamma > 0$ and $\delta > 0$. If $p = c\ln(n)/r$, $n < 2^{r^{1-\delta}}$, and $\lim_{n,r\to\infty} \ln(n)n^{1-c}/r > e(1+\gamma)/c$ then A_3 "gives up" with probability tending to 1 as $n, r \to \infty$.

The next theorem gives a condition under which random instances of SAT are unsatisfiable with probability tending to 1.

Theorem 5:

> Let c be a constant such that $.4 < c < 1$. If $p = c\ln(n)/r$ and $\lim_{n,r\to\infty} n^{1-c}/r = \infty$ then an instance of SAT generated according to $J(n, r, p)$ is unsatisfiable with probability tending to 1.

Proof:

> Let t be a random truth assignment to the variables of V. The probability that a random clause has value *true* under t is the probability that at least one literal in the clause is made *true* by t. This is one minus the probability that none of the literals is made true. The probability that none of the literals is made *true* by t is $(1-p)^r$. Hence, the probability that a random clause has value *true* under t is $1-(1-p)^r$. The probability that n clauses chosen independently have value *true* is $(1-(1-p)^r)^n$. Therefore, the average number of truth assignments satisfying all clauses is $2^r(1-(1-p)^r)^n$. Since $p = c\ln(n)/r$ this is $e^{r\ln(2)}e^{-n^{1-c}}$ as $n, r \to \infty$. But this expression tends to zero if $\lim_{n,r\to\infty} n^{1-c}/r = \infty$. Since the average number of satisfying truth assignments is an upper bound for the probability that there is at least one satisfying truth assignment, the theorem is proved.

From theorems 3 and 5 we have that if $p = c\ln(n)/r$, $.4 < c < 1$ and $n < 2^r$ then a random instance of SAT generated according to $J(n, r, p)$ almost never has a solution if $\lim_{n,r\to\infty} n^{1-c}/r = \infty$ but a solution will almost always be found by A_3 if, for any $\epsilon > 0$, $\lim_{n,r\to\infty} n^{1-c}/r^{1-\epsilon} < \infty$. These conditions define the line called "SAT BOUNDARY" in Figure 1 such that points to the right of that line ($\lim_{n,r\to\infty} n^{1-c}/r^{1-\epsilon} < \infty$) correspond to sets of parameter values that result either in almost all random instances being satisfiable or containing a null clause, and all points to the left of that line correspond to values leading to the generation of unsatisfiable instances, almost always. Since A_1 nearly always finds a solution when $p > \ln(n)/r$ and since almost all random instances are unsatisfiable when $p < .4\ln(n)/r$, the collection of algorithms mentioned here has been shown to be very effective in finding solutions to random instances of SAT when at least one solution exists.

5. Conclusion

We have shown that the combination of algorithms A_1, A_2 and A_3 solve random instances of SAT generated according to $J(n, r, p)$ with probability tending to 1 as $n, r \to \infty$ as long as $p \leq .4\ln(n)/r$ or $p \geq \ln(n)/r$ or $p = c\ln(n)/r$, $.4 < c < 1$ and $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r = 0$. The region of the parameter space now known, but not previously known, to correspond to instances which can be solved in polynomial time almost always is shown shaded in Figure 1. Noting the relationship between the shaded area, the area to the right of **II** and the "SAT BOUNDARY" in Figure 1, we see that no algorithm for finding a *satisfying* truth assignment, when one exists, can perform much better than A_3. The only region of the input model parameters not yet known to be covered by a fast algorithm with good probabilistic performance is $p = c\ln(n)/r$, $.4 < c < 1$ and $\lim_{n,r\to\infty} \ln(r)\ln^2(n)n^{1-c}/r > 0$ (bounded by **I** and the "SAT BOUNDARY" line in Figure 1). Instances generated in this region of the parameter space are almost always unsatisfiable.

The result presented here represents an interesting improvement over previous results. According to the best previous result, bactracking with the pure literal rule runs in polynomial average time when $n\ln(n) \leq \alpha\sqrt{r\ln(r)}$, α constant, in the range $.4\ln(n)/r < p < \ln(n)/r$. This means np, the average number of clauses containing a variable randomly chosen from the given set of r variables, is less than a constant times $\sqrt{\ln(r)/r}$. Since the number of clauses containing a particular variable is binomially distributed, we may use the chernoff bound for binomial distributions and find that the probability that a variable appears in more than one clause is less than $e^{-\alpha\sqrt{r/\ln(r)}}$. Since variables appear in clauses independently, the probability that all variables are in at most one clause is greater than $(1 - e^{-\alpha\sqrt{r/\ln(r)}})^r$ which tends to 1 as $r \to \infty$. Thus, almost all instances in the range $n\ln(n) \leq c\sqrt{r\ln(r)}$, $.4\ln(n)/r < p < \ln(n)/r$ have no variable present in more than one clause. Such instances are probably not very interesting and can be solved easily by assigning the value *true* to every literal in the instance. On the other hand examples of regions in which almost every random instance of SAT may be solved in polynomial time by the algorithms presented here are $p = .4\ln(n)/r$ and $n = o(r^{5/3})$ or $p = .5\ln(n)/r$ and $n = o(r^2)$ or $p = .75\ln(n)/r$ and $n = o(r^4)$ or $p \geq \ln(n)/r$. In all these examples the average number of clauses containing a particular literal may be increasing with r and n.

Finally, we comment that our result is much stronger than a similar result obtained in [8] since that result required p to be fixed while our result holds even if p tends to 0. To see the difference in the proper perspective notice that if p is fixed then the average number of literals in a clause is $\Theta(r)$. However, our results hold even if the average number of literals in a clause is $o(r)$ or $o(n)$. The most interesting instances have $\Theta(\log(n))$ literals per clause, on the average (these are generated in the vicinity of the "SAT BOUNDARY" line of Figure 1). Our results apply to such instances whereas the results of [8] do not if n and r are polynomially related (in fact, those results apply only to instances which have far more literals per clause and are "very" satisfiable).

6. References

1. Chao, M.T., *Probabilistic Analysis and Performance Measurement of Algorithms for the Satisfiability Problem*, Ph. D. thesis, Case Western Reserve University, Cleveland, Ohio (1985).
2. Chao, M.T. and Franco, J., "Probabilistic analysis of two heuristics for the 3-satisfiability problem," *SIAM J. Comput.* **15** (1986), pp. 1106–1118.
3. Chao, M.T. and Franco, J., "Probabilistic analysis of a generalization of the unit clause literal selection heuristic for the k-satisfiability problem," Tech. Report No. 165, Indiana University (1985).
4. Cooper, R.B., *Introduction to Queueing Theory*, MacMillan, New York (1972), p. 163.
5. Davis, M. and Putnam, H., "A computing procedure for quantification theory," *J.ACM* **7** (1960), pp. 201–215.
6. Franco, J., "On the probabilistic performance of algorithms for the satisfiability problem," *Information Processing Letters* **23** (1986), pp. 103–106.
7. Franco, J., "Probabilistic analysis of the pure literal heuristic for the satisfiability problem," *Annals of Operations Research* **1** (1984), pp. 273–289.
8. Franco, J. and Paull, M., "Probabilistic analysis of the Davis- Putnam Procedure for solving the satisfiability problem," *Discrete Applied Mathematics* **5** (1983), pp. 77–87.
9. Galil, Z., " On the complexity of regular resolution and the Davis-Putnam Procedure," *Theoretical Computer Science* **4** (1977), pp. 23–46.
10. Goldberg, A., Purdom, P.W. and Brown, C.A., "Average time analyses of simplified Davis-Putnam Procedures," *Information Processing Letters* **15** (1982), pp. 72–75.
11. Plotkin, J.M. and Rosenthal, J.W., "On the expected number of branches in analytic tableaux analysis in propositional calculus," *Notices Amer. Math. Soc.* **25** (1978) A-437.
12. Purdom, P.W., "Search rearrangement bactracking and polynomial average time," *Artificial Intelligence* **21** (1983), pp. 117–133.
13. Purdom, P.W. and Brown, C.A., "The pure literal rule and polynomial average time," *SIAM J. Comput.* **14** (1985), pp. 943–953.
14. Purdom, P.W. and Brown, C.A., *The Analysis of Algorithms*, Holt, Rinehart and Winston, New York (1985), p. 465.

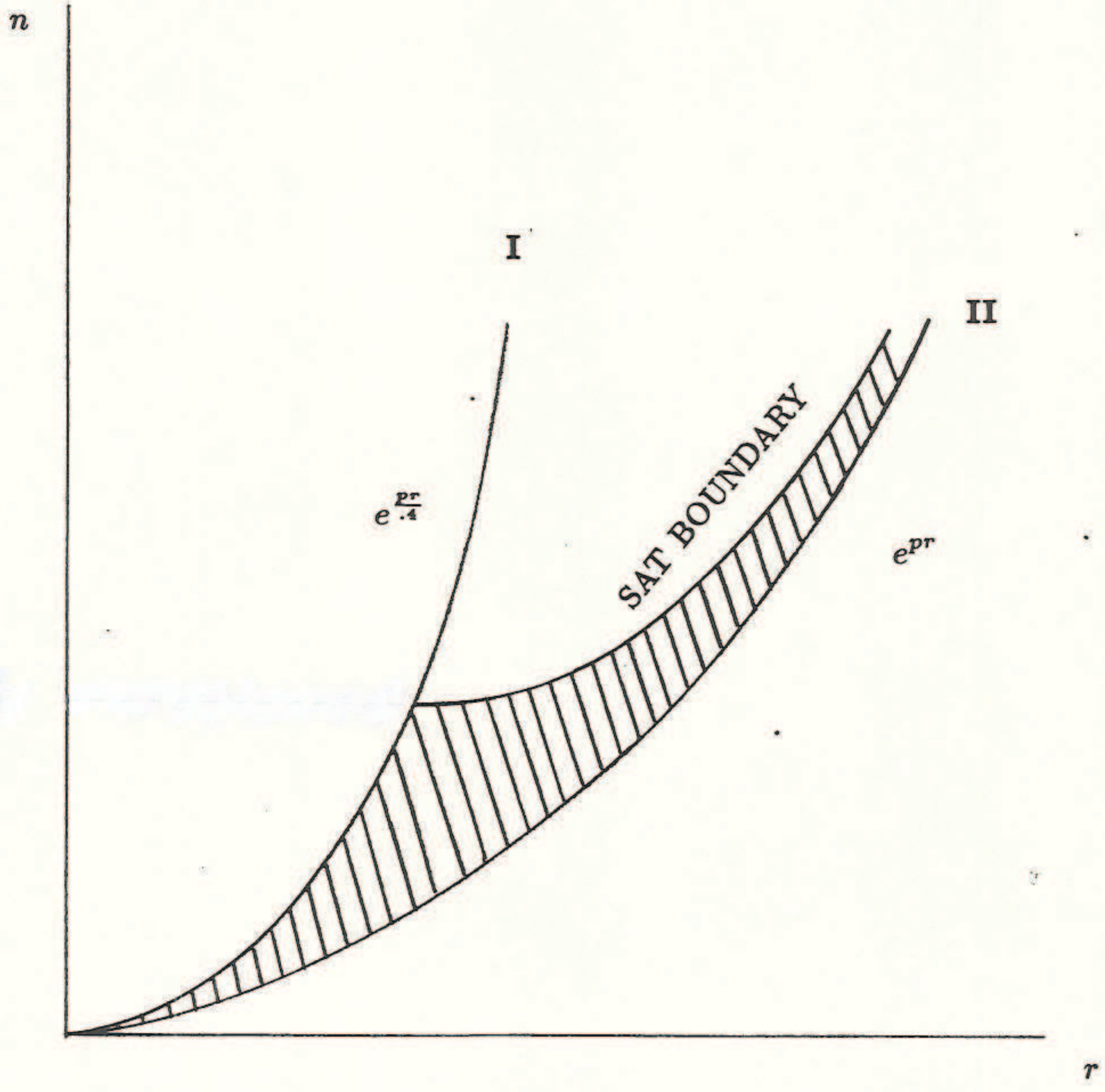
n
I
II
SAT BOUNDARY
$e^{\frac{pr}{.4}}$
e^{pr}
r

Figure 1: The parameter space for which A_1, A_2 and A_3 almost always solve a random instance of SAT generated under $J(n, r, k)$. A_2 works well, in probability, in the region to the left of curve I. A_1 works well, in probability, in the region to the right of II. A_3 works well, in probability, in the hatched region.