Average Time Analyses of Simplified Putnam-Davis Procedures

by

Paul W. Purdom, Jr. and Synthia A. Brown

Computer Science Department

Indiana University

Bloomington, Indiana 47405

TECHNICAL REPORT No. 101

AVERAGE TIME ANALYSES OF SIMPLIFIED PUTNAM-DAVIS PROCEDURES

PAUL W. PURDOM, JR. AND CYNTHIA A. BROWN

APRIL 1981

This material is based upon work supported by the National Science Foundation under Grant NSF MCS 7906110

Average Time Analyses of Simplified Putnam-Davis Procedures

1. Introduction

The Putnam-Davis procedure [4] is a powerful method for solving satisfiability problems. Satisfiability is NP-complete, and the worst case time of the procedure is exponential. In this paper we study the average time. Since the full Putnam-Davis procedure is difficult to analyze, we consider simplified versions of it. After correcting some errors in a paper of Goldberg's, we confirm that the average time of a simplified version of the Putnam-Davis procedure is polynomial over certain NP-complete sets of satisfiability problems, and we give a formula for the degree of the polynomial involved. We also consider the average time of other versions, to illustrate the sensitivity of the analysis to small changes in the recurrence for the average time and to help clarify the contribution of various portions of the procedure to the final results.

The Putnam-Davis procedure, which is used to determine whether a conjuctive normal form (CNF) predicate is satisfiable, may be stated as follows:

Putnam-Davis Procedure

1. [Satisfiable] If there are no clauses the predicate is satisfiable.
2. [Unsatistiable] If any clause is empty the predicate is not satisfiable.
3. [Unit clause] If there is any clause consisting of a single literal, select a variable associated with such a clause. Set the variable so that the clause is true, simplify the predicate

(see below), and apply the Procedure. The original predicate is satisfiable iff the simplified predicate is.

4. [Pure literal] If any variable appears only negated or only unnegated, choose it as the next variable. Set it so that its literals are true, simplify the predicate, and apply the Procedure. The original predicate is satisfiable iff the simplified predicate is.
5. [Splitting] Select any variable. Form two predicates by setting the variable to each value and simplifing the resulting predicates. Apply the Procedure to both predicates. The original predicate is satisfiable iff at least one of the simplified predicates is satisfiable.

The procedure ignores any variable that does not appear in the predicate. Simplification of a predicate is performed as follows: drop from the predicate all clauses containing a true literal and drop from each clause any false literals.

Goldberg [5] simplified the Putnam-Davis procedure by omitting the unit clause rule, by modifying the unsatisfiable rule to say that if all the variables have been set without finding a solution then the predicate is unsatisfiable, and by modifying the pure literal rule (see below). He also selected the variables in a fixed order. To simplify slightly, he analyzed the performance of the simplified algorithm on random CNF predicates with t clauses and v variables. An arbitrary literal (either a variable or its negation) occurs in an arbitrary clause with probability p. (Notice that with this model some variables may not occur anywhere in the predicate.) He presented the following recurrence for the average time:

$$A(t,v) = atv + 2 \sum_{i \geq 1} \binom{t}{i} p^i (1-p)^{t-i} A(t-i,v-1) \quad \text{for } t,v \geq 1,$$
$$A(t,0) = 0, \quad A(0,v) = 0. \tag{1}$$

This recurrence was intended [6] to describe the behavior of the version of the simplified algorithm where the variables are always selected in fixed order but the pure literal rule is used to generate only one predicate when the selected literal is pure. Actually, however, Eq. 1 describes an algorithm with one additional "feature": when the selected variable does not appear in the predicate, investigation of the predicate stops without determining whether it is satisfiable.

The algorithm actually stated in Goldberg's paper is the simplified algorithm with the pure literal rule omitted entirely. The recurrence for this algorithm is

$$A(t,v) = atv + 2 \sum_{i \geq 0} \binom{t}{i} p^i (1-p)^{t-i} A(t-i,v-1) \quad \text{for } t,v \geq 1,$$
$$A(t,0) = 0, \quad A(0,v) = 0. \tag{2}$$

This differs from Eq. 1 only by the limit on i.

The recurrence for the algorithm Goldberg intended to analyze is

$$A(t,v) = atv + 2 \sum_{i \geq 1} \binom{t}{i} p^i (1-p)^{t-i} A(t-i,v-1) + (1-p)^{2t} A(t,v-1),$$
$$A(t,0) = 0, \quad A(0,v) = 0. \tag{3}$$

The solution to Eq 1. is bounded by a function that is linear in v and polynominal in t [5]. We will show that the solution to Eq. 2 is linear in t and exponential in v. An examination of the coefficients of the terms in Eq. 3 suggests that its solution is between that of Eq. 1 and Eq. 2. We will show that its solution is essentially the same as that of Eq. 1. A careful comparison of these recurrences helps show why the pure literal rule reduces the time for solving some random sets of CNF predicates from exponential to polynomial average time.

2. Solution of the Recurrences

Goldberg [5] claims that the solution for Eq. 1 is less than cvt^k, where $k = \lceil -1/\lg(1-p) \rceil$ and c is a constant that depends on p. (We use lg for the logarithm base 2.) He gives the proof for the case $p = 1/3$. He intended k to be given by $\lceil \varepsilon - 1/\lg(1-p) \rceil$ for any $\varepsilon > 0$ [7].

To solve Equation (2), define the exponential generating function

$$G_v(z) = \sum_{t \geq 0} A(t,v) \frac{z^t}{t!} . \tag{4}$$

Multiplying Eq. (2) by $z^t/t!$ and summing over t (see Knuth [8, pp.87-88]) gives

$$G_v(z) = avze^z + 2 \sum_{t \geq 0} \sum_{0 \leq i \leq t} \frac{(1-p)^i z^i}{i!} A(i,v-1) \frac{p^{t-i} z^{t-i}}{(t-i)!}$$

$$= avze^z + 2e^{zp} G_{v-1}(z(1-p)) . \tag{5}$$

Iterating Equation (5), we obtain

$$G_v(z) = \sum_{0 \leq i \leq v} 2^i a(v-i) z(1-p)^i \exp\Big[z(1-p)^i + \sum_{0 \leq j \leq i-1} zp(1-p)^j \Big]$$

$$= \sum_{0 \leq i \leq v} a[2(1-p)]^i (v-i) ze^z$$

$$= aze^z \frac{(2p-1)v + [2(1-p)]^{v+1} - 2(1-p)}{(2p-1)^2} . \tag{6}$$

Expanding $G_v(z)$ in a power series gives

$$A(t,v) = at \frac{[2(1-p)]^{v+1} - 2(1-p) + (2p-1)v}{(2p-1)^2} . \tag{7}$$

This solution can be checked by substitution into Equation (2). This result is linear in t and exponential in v.

See [7] for the generating function for Eq. 1, and for the reasons why it is difficult to use generating functions to obtain an explicit solution to Eq. 1. Similar difficulties occur in Eq. 3.

Eq. 3 can be solved using essentially the same technique that Goldberg suggests for Eq. 1. It can be proved by induction that $A(t,v)$ is increasing in v, so

$$A(t,v) \leq atv + 2 \sum_{i \geq 1} \binom{t}{i} p^i (1-p)^{t-i} A(t-i,v) + (1-p)^{2t} A(t,v). \quad (8)$$

Now, $A(t,v) \leq A(t)$, where $A(t)$ is the solution to the recurrence

$$(1-(1-p)^{2t})A(t) \leq bt + 2 \sum_{i \geq 1} \binom{t}{i} p^i (1-p)^{t-i} A(t-i), \quad (9)$$

where $b = av$. The solution to Eq. 9 is linear in b, and hence in v, because Eq. 9 is linear in the A's. We will now show that $A(t) \leq ct^x$, where $x > -1/\lg(1-p)$ and c is independent of t but is a linear function of v and depends on x.

First note that for any given t_0 we can choose c so that $ct^x \geq A(t)$ for $t \leq t_0$. Assume the claim is true for $A(r)$ where $r \leq t-1$. Then

$$(1-(1-p)^{2t})A(t) \leq bt + 2ct^x \sum_{i \geq 0} \binom{t}{i} p^i (1-p)^{t-i} (1 - \tfrac{i}{t})^x. \quad (10)$$

Using the binomial theorem and Stirling numbers of the second kind [8, p.65], the summation can be written as

$$\sum_i \sum_j \sum_k \binom{t}{i} \binom{x}{j} \binom{i}{k} \left\{ {j \atop k} \right\} k!\, p^i (1-p)^{t-i} (-1)^j t^{-j}. \quad (11)$$

Using the formula $\binom{t}{i}\binom{i}{k} = \binom{t}{k}\binom{t-k}{i-k}$, the sum over i can be done using the binomial theorem, giving

$$\sum_j \sum_k \binom{x}{j} \binom{t}{k} \left\{ {j \atop k} \right\} k!\, p^k (-1)^j t^{-j}. \quad (12)$$

The quantity $\binom{t}{k} k!$ can be expanded using Stirling numbers of the first kind [8, p.65]. A change of variables gives

$$\sum_j \sum_k \sum_n \binom{x}{j} \left\{ {j \atop k} \right\} \left[{k \atop j-n} \right] (-1)^{n+k} \, p^k \, t^{-n}. \tag{13}$$

The coefficient of the $n = 0$ term is $(1-p)^x$. The coefficient of t^{-n} in this formula is a power series in p. For $p = 1$ this coefficient is

$$\sum_j \sum_k \binom{x}{j} \left\{ {j \atop k} \right\} \left[{k \atop j-n} \right] (-1)^{n+k} = 0 \tag{14}$$

[8, p.67]. Since the power series converges for $p = 1$, it converges for $0 \leq p \leq 1$. Thus

$$\sum_i \binom{t}{i} \, p^i \, (1-p)^{t-i} \, (1 - \tfrac{i}{t})^x = (1-p)^x + O(t^{-1}). \tag{15}$$

This shows that

$$(1-(1-p)^{2t}) \, A(t) \leq bt + 2ct^x \, (1-p)^x + O(t^{x-1}), \text{ or} \tag{16}$$

$$A(t) \leq \frac{2ct^x \, (1-p)^x}{(1-(1-p)^{2t})} + O(t^{\max(1,x-1)}). \tag{17}$$

To complete the proof the right hand side of Eq. 17 must be less than or equal to ct^x. This is true for $x > -1/\lg(1-p)$ and t sufficiently large, because then $2(1-p)^x < 1$, the quantity $(1-p)^{2t}$ is insignificant, and the difference between ct^x and $2ct^x \, (1-p)^x$ will be larger than the $O(t^{\max(1,x-1)})$ term. Choosing t sufficiently large establishes the value of t_0 for the basis step.

The same proof technique applies to Eq. 1 and gives the same results. It is interesting to notice why this technique does not apply to Eq. 2. Corresponding to Eq. 9 one obtains

$$(1-2(1-p)^t) \, A(t) \leq bt + 2 \sum_{i \geq 1} \binom{t}{i} p^i \, (1-p)^{t-i} \, A(t-i). \tag{18}$$

For $t = 1$ and $p < 1/2$, $(1-2(1-p)^t)$ is negative, so Eq. 18 gives a lower limit rather than an upper limit on $A(1)$.

3. Conclusion

The solutions to the recurrences for the average time of the Putnam-Davis procedure are quite sensitive to changes in the coefficients. Correcting the minor errors in [5], Goldberg's method can be used to show that the Putnam-Davis procedure takes polynomial average time on problems with a fixed value of p. Using a fixed value of p describes sets of problems where the clauses increase in size as v becomes large. If p varies inversely with v then the simplified version of the Putnam-Davis procedure requires exponential time. For such problems, we have shown [2,3] that simple backtracking (which uses Steps 1, 2, and 5 of the Putnam-Davis procedure) takes subexponential time (asymptotically less than e^{cv} for any c) provided t increases more rapidly than v^{α} for $\alpha > 1$. We are now studying the effect of combining backtracking with the pure literal rule.

Using the unit clause rule to change the search order greatly reduces the time required to solve some problems [1,9]. The use of the pure literal rule to control search order is likely to have a similar effect. The analysis of the full pure literal rule, however, is probably difficult. So far no theoretical work has been done on the effect of stopping (in rule 5) at the first solution.

Computation of the average running time on difficult problem sets is a useful way to compare the efficiency of various searching algorithms. Such analyses help identify the features in an algorithm that permit it to solve many difficult problems rapidly.

The research reported herein was supported in part by the National Science Foundation under grant number MCS7906110.

References

1. James R. Bitner and Edward M. Reingold, "Backtrack programming techniques", Comm. ACM 18 (1975) pp. 651-665.
2. Cynthia A. Brown and Paul Walton Purdom, Jr., "An average time analysis of backtracking", SIAM J. Comput., (to appear).
3. Cynthia A. Brown and Paul Walton Purdom, Jr., "How to search efficiently", Indiana University Computer Science Technical Report No. 105 (1981).
4. Martin Davis, George Logemann, and Donald Loveland, "A Machine Program for Theorem Proving", Comm. ACM 5 (1962) pp. 394-397.
5. Allen Goldberg, "Average case complexity of the satisfiability problem", Proceedings Fourth Annual Workshop on Automated Deduction (1979) pp. 1-6.
6. Allen Goldberg, private communication (1981).
7. Allen Goldberg, "On the Complexity of the Satisfiability Problem", Courant Computer Science Report No. 16, New York University (1979).
8. Donald E. Knuth, The art of computer programming, v.1, Addison Wesley, Reading, Mass. (1973).
9. Paul Walton Purdom, Jr. and Cynthia A. Brown, "An analysis of backtracking with search rearrangement", Indiana University Computer Science Department Technical Report No. 89 (1980).