EVALUATING SEARCH METHODS ANALYTICALLY*

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

Computer Science Department, Indiana University, Bloomington, Indiana 47405

ABSTRACT

A unified approach to analyzing search algorithms is presented. Each algorithm is characterized by the types of random problems that it can solve rapidly. The results are displayed in a way that clearly indicates the strengths and weaknesses of each algorithm.

1. *Introduction*

Many interesting problems can, at present, best be solved by search methods [7]. In the worst case, searching requires exponential time. Several clever techniques have been developed to speed up searching (see, for example, [1, 3, 6, 8, 12, 13, 14, 15, 20, 21, 22]). While each of these techniques is clearly helpful for some class of problems, it is difficult to evaluate the importance of each method (and of each combination of methods). Analytical studies have been done on several search methods [2, 4, 5, 9, 10, 11, 17]. Each analysis was parameterized in a way that emphasized search problems for which the method under consideration had an interesting behavior. Thus, Goldberg [9, 10] studied the pure literal rule using conjunctive normal form (CNF) predicates with long clauses, while Brown and Purdom [2, 17] studied backtracking using CNF predicates with short clauses. Since each algorithm has its own region of interesting behavior, the results of the analyses are difficult to compare with each other.

In this paper we describe a unified approach to analyzing search algorithms, one that indicates the strengths and weaknesses of each algorithm in a way that makes comparisons straightforward. We first analyze the average time behavior of each algorithm on random CNF problems characterized by v - the number of variables, t - the number of clauses, and p - the probability a given literal is in a clause (so that the average length of a clause is $2pv$). This step is similar to the initial step of previous approaches, which continued by choosing particular functions $p(v)$ and $t(v)$ and studying the resulting asymptotic behavior. We continue by letting $p(v)$ and $t(v)$ be *arbitrary* functions of v , and finding the asymptotic behavior of the algorithm as v approaches infinity. Finally, we find the relation between $p(v)$ and $t(v)$ that characterizes the boundary between exponential and polynomial average time for the algorithm. The results can be displayed with a diagram of $p(v),t(v)$ space which shows the exponential vs. polynomial contour for each algorithm. Fig. 1 shows the results for several basic algorithms. Contours are drawn for ordinary backtracking where all solutions are found, using fixed search order (the results are derived in [18]); searching using the pure literal rule and a fixed search order [10, 18]; pattern of occurrence (an algorithm developed for use with problems that have only a few clauses) [18]; and elimination of unused variables followed by exhaustive search (this paper). Fig. 2. shows the results of using the same approach to obtain a more detailed comparison between ordinary backtracking and simple search rearrangement [16]. From these figures it is clear that the pure literal rule is useful for problems with large clauses (constraints that are easy to satisfy), that backtracking is useful for problems with many short clauses (large numbers of constraints that are not easy to satisfy), and that specialized methods are useful for problems with a small number of clauses (constraints). As more sophisticated algorithms are analyzed, the same approach can be used to identify the types of problems for which they are most suitable. The approach is even more useful for identifying which elementary techniques should be combined to produce powerful general purpose algorithms.

2. *The Model*

We compute the average running time of each algorithm using random CNF predicates. The predicates are formed using v variables, and thus $2v$ literals. A random clause is formed by independently selecting each literal with probability p. For each variable the probability of selecting both the positive and negative literal is p^2 , so tautological clauses are common unless p is small (smaller than $v^{-1/2}$). A random predicate is formed by independently selecting t clauses. Notice that for any $0 < p < 1$ *all* predicates with t clauses are included in the set of model problems. If p is small then predicates with short clauses have a higher probability while if p is large then predicates with long clauses have a higher probability.

3. *Results*

The derivations for the results below are contained in the cited papers. We use the conventions: $a = vp(v)$; $\alpha = (n \ln v)/v$ for some large constant n ; and ϵ is a small positive constant. We say $f(v) \leq g(v)$ when $\lim_{v\to\infty} f(v)/g(v) \leq 1$.

Polynomial average time is used when:

1. Ordinary backtracking (fixed search order) [18]:
   a) $a \leq \ln 2$, $t \geq \frac{(\ln 2 - \alpha)v}{-\ln(1 - e^{-a})}$, or
   b) $a \geq \ln 2$, $t \geq \frac{(\ln 2 - \alpha)vd}{a}$, where

*Research reported herein was supported in part by the National Science Foundation under grant number MCS 7900110.

d is the largest root of $\ln(1+d)+d(\ln(1+1/d)) = 2a$.

2. Simple search rearrangement backtracking [16]
   a) same as 1, or (for example)
   b) $a \geq 3.5$, $t \geq \frac{v \ln 2}{2ea^2} \exp(2a-\alpha)(1+O(\frac{1}{a}))$.
   (See [16] for more details).

3. Simple pure literal rule [10, 18]

   $p \geq \epsilon$.

4. Pattern of occurrence [18]:

   $t \leq (\ln \ln v)/(\ln 3)$.

5. Elimination of unused variables followed by exhaustive search [this paper]:

   $t \leq \alpha/2p$.

The average number of solutions per problem is polynomial when

$$t \geq \frac{(\ln 2 - \alpha)v}{-ln(1-e^{-a})}.$$

4. *Sample Analysis*

To illustrate our method we give a brief analysis of Algorithm 5. The probability that neither literal for a variable occurs in a given predicate is $(1-p)^{2t}$. If i variables occur in a predicate the time for exhaustive search is 2^i. The probability that i particular variables occur, and that the remaining $v-i$ do not, is $[1-(1-p)^{2t}]^i[(1-p)^{2t}]^{v-i}$. The number of ways to choose i variables out of v is $\binom{v}{i}$. Therefore the average running time for Algorithm 5 is

$$\sum_i 2^i\binom{v}{i}[1-(1-p)^{2t}]^i[(1-p)^{2t}]^{v-i} = [2-(1-p)^{2t}]^v .$$

To obtain polynomial time, this average must be no more than v^n for some n. In other words $[2-(1-p)^{2t}]^v \leq v^n$ or $\ln[2-(1-p)^{2t}] \leq \frac{n}{v}\ln v$. Since $\ln[2-(1-p)^{2t}]$ must be small we can use $\ln[2-(1-p)^{2t}] \approx 1-(1-p)^{2t}$, which gives $(1-p)^{2t} \geq 1-\frac{n}{v}\ln v$ or (using $\ln(1-x) \approx x$ for small x)

$$t \leq \frac{n \ln v}{2vp} = \frac{\alpha}{2p}.$$

5. *Conclusions*

For random problem sets where $p(v)$ or $t(v)$ is extremely large or small there are search algorithms that solve the problems in an average time that is polynomial in the size of the problem. The time for these *extreme cases* is also polynomial in the number of variables except when $p(v)$ is large and $t(v)$ is exponential or larger.

Each of the algorithms 2-5 has a region of $p(v),t(v)$ space where it is much better than any of the other algorithms. Algorithm 1 is as good as Algorithm 2 for some regions. A diagram such as Fig. 1 gives a useful display of the strengths and weaknesses of each algorithm.

Addendum:

Recently we did a more careful analysis of the pure literal rule and showed that it leads to polynomial average time when $t \leq n\ln v$ and when $pt \leq (\frac{n\ln v}{v})^{3/2}$ [19]. This is a major improvement over the performances shown in Fig. 1.

REFERENCES

[1] James R. Bitner and Edward M. Reingold, "Backtrack Programming Techniques", *Comm. ACM,* v. 18 (1975) pp. 651-655.

[2] Cynthia A. Brown and Paul Walton Purdom Jr., "An Average Time Analysis of Backtracking", *SIAM J. Comput.* 10 (1981) pp. 583-593.

[3] Martin Davis and Hilary Putnam, "A Computing Procedure for Quantification Theory", *JACM,* v. 7 (1960) pp. 201-215.

[4] John Franco, "Average Analysis of the Pure Literal Heuristic", Case Institute of Technology Report No. CES-81-4 (1981).

[5] John Franco and Marvin Paull, "Probabilistic Analysis of the Davis-Putnam Procedure for Solving the Satisfiability Problem", Case Institute of Technology Report No. CES-81-3 (June 1981).

[6] Eugene C. Freuder, "A Sufficient Condition for Backtrack-Free Search", *JACM,* v. 29, No. 1 (January, 1982) pp. 24-32.

[7] Michael R. Garey and David S. Johnson, *Computers and Intractability,* W.H. Freeman and Co., San Francisco (1979).

[8] John Gaschnig, "Performance Measurement and Analysis of Certain Search Algorithms", Thesis, Carnegie-Mellon (1979).

[9] Allen Goldberg, "Average Case Complexity of the Satisfiability Problem", *Proccedings of the Fourth Workshop on Automatcd Deduction* (1979), pp. 1-6.

[10] Allen Goldberg, Paul Walton Purdom, Jr. and Cynthia A. Brown, "Average Time Analysis of Simplified Putnam-David Procedures", *Info. Proc. Letters* (to appear).

[11] Robert M. Haralick and Gordon L. Elliot, "Increasing Tree Search Efficiency for Constraint Satisfaction Problems", Report from Virginia Polytechnic Institute, 1979.

[12] Robert M. Haralick and Linda G. Shapiro, "The Consistent Labeling Problem", *IEEETPAMI,* v. 1 (1979), pp. 1773-184, v. 2 (1980), pp. 193-203.

[13] Burkhard Monien and Ewald Speckenmeyer, "Three-Satisfiability is Testable in $O(1.62^r)$ Steps", Report No. 3, Theoretical Informatics Series, University of Paderborn (1979).

[14] E. T. Parker, "Computer Investigation of Orthogonal Latin Squares of Order Ten", *Proc. Sym. Appl. Math.*, v. 15 (1963), Amer. Math. Soc., Providence, R.I. p. 73.

[15] Paul Walton Purdom, Jr. "Solving Satisfiability Problems with Less Searching", Indiana University Computer Science Technical Report No. 117 (1981).

[16] Paul Walton Purdom, Jr., "Search Rearrangement Backtracking and Polynomial Average Time", Indiana University Computer Science Technical Report No. 123 (1982).

[17] Paul Walton Purdom, Jr. and Cynthia A. Brown, "An Analysis of Backtracking with Search Rearrangement", *SIAM J. Comput.* (to appear).

[18] Paul Walton Purdom, Jr. and Cynthia A. Brown, "Polynomial Average-time Satisfiability Problems", Indiana University Computer Science Technical Report No. 118 (1981).

[19] Paul Walton Purdom, Jr. and Cynthia A. Brown, "The Pure Literal Rule and Polynomial Average Time", Indiana University Computer Science Technical Report No. 128 (to appear).

[20] Paul Purdom, Cynthia Brown and Edward Robertson, "Multi-Level Dynamic Search Rearrangement", *Acta Informatica* v. 15 (1981), pp. 99-114.

[21] Thomas J. Schaefer, "The Complexity of Satisfiability Problems", *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing,* (1978) pp. 216-226.

[22] David Waltz, "Understanding Line Drawings of Scenes with Shadows", in *The Psychology of Computer Vision,* edited by Patrick Henry Winston, McGraw-Hill, New York (1975).

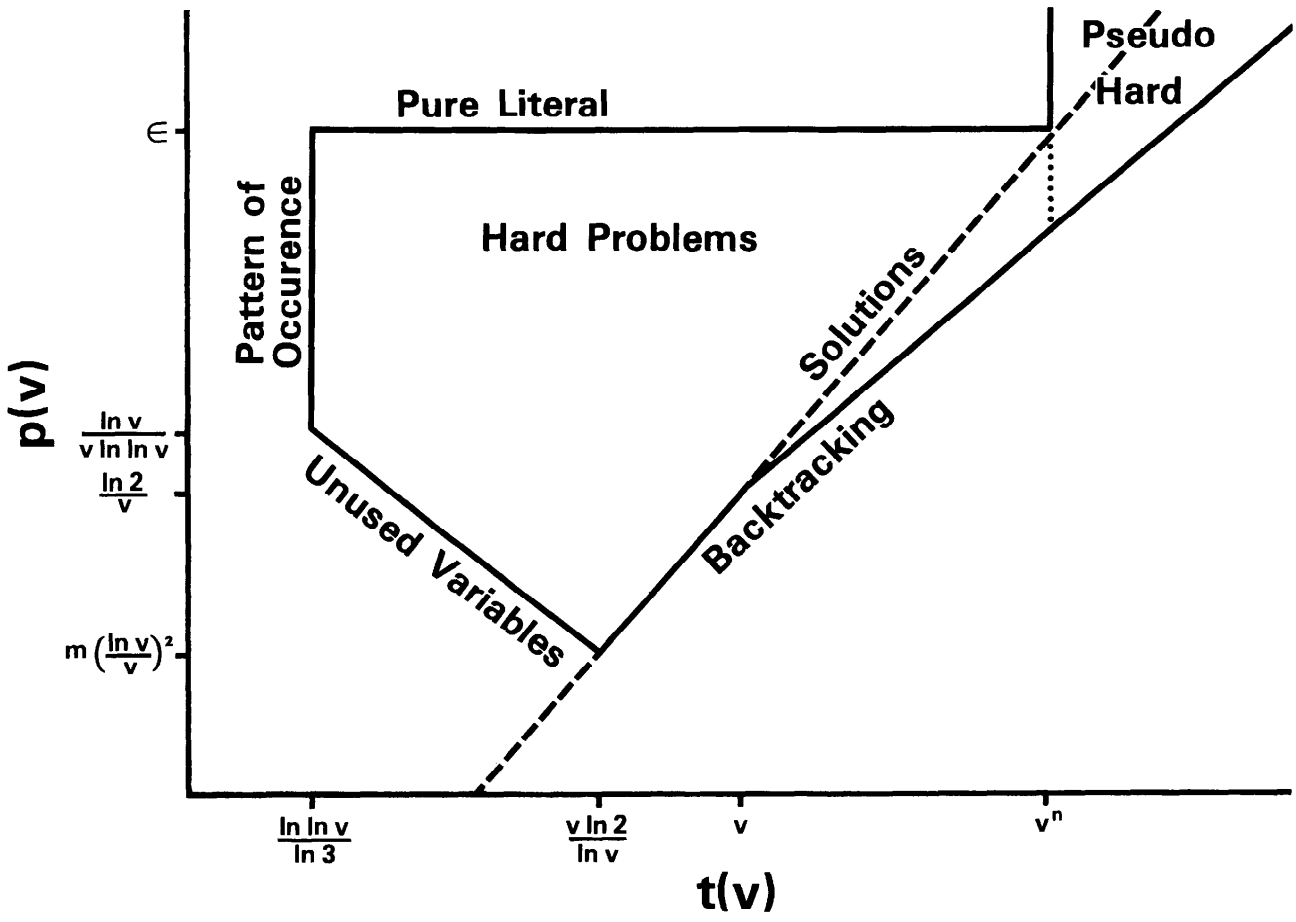

Figure 1. A diagram showing the regions of $p(v),t(v)$ space where random CNF predicates can be solved in polynomial average time. A portion of the contour separating the region of polynomial behavior from the region of exponential behavior is shown for several algorithms. The part of the space where each algorithm performs best is labelled with the name of the algorithm. The central region contains problem sets for which no polynomial average time algorithm is known. In most of this region, the problems have an exponential number of solutions, but below the line marked "solutions" the average number of solutions is polynomial. The region marked "pseudo-hard" contains problem sets for which the analyzed algorithms take average time exponential in the number of variables but polynomial in the problem size (the typical problems are exponentially large there).

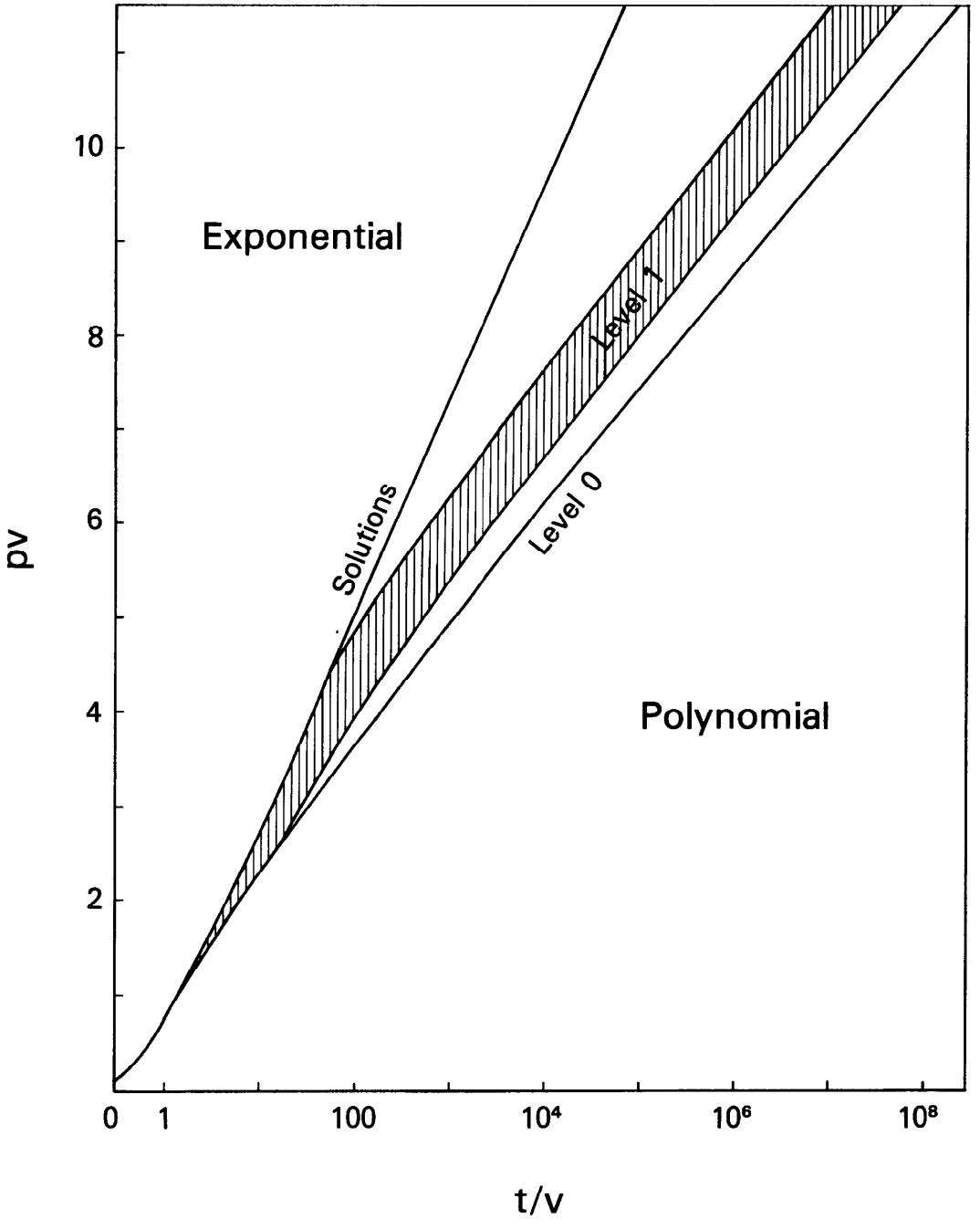

Figure 2. A graph giving more details on the performance of backtracking algorithms. The vertical axis is pv , the average number of literals per clause. The horizontal axis is t/v , the number of clauses per variable. The curve marked *Solutions* separates the region where the average number of solutions per problem is exponential from where it is polynomial. The curve marked *Level 0* separates the region where the average running time of ordinary backtracking is exponential from where it is polynomial. The analysis for simple search rearrangement backtracking produces only limits on its performance. The shaded region marked *Level 1* separates the region where the average running time of simple search rearrangement backtracking is exponential from where it is polynomial.