On the Probabilistic Performance of Algorithms for the Satisfiability Problem

by

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

TECHNICAL REPORT NO. 167

On the Probabilistic Performance of Algorithms for the Satisfiability Problem

by

John Franco

March, 1985

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

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 efficient algorithm for solving this problem.

The Davis-Putnam Procedure (DPP) [4] is a well known, much studied method for solving instances of SAT. The probabilistic analysis of variants of DPP under the assumption of constant-density input distributions such as in [7], [8] and [9] has given the impression that the Davis-Putnam Procedure is intrinsically a very fast method for solving most instances of SAT. This impression is moderated somewhat by the results of this letter which show that the following two trivial algorithms, run concurrently, solve "more" instances of SAT in polynomial time than any previously studied algorithm.

$A_1(I)$:
    Repeat
        Randomly choose a truth assignment t to variables in I
    Until t satisfies I
    Return (t)

$A_2(I)$:
    Search I for a null clause
    If a null clause is found Then Return ("not satisfiable")
    Else Return ("cannot determine whether I is satisfiable")

The results reported here represent the conclusion of work begun in [6]. They indicate that the favorable results previously obtained are probably due more to the probabilistic model chosen than to characteristics of the Davis-Putnam Procedure. Furthermore, it appears likely that similar statements can be made about probabilistic results obtained for some NP-complete problems on random graphs because of a certain property of constant-density models. This point will be expanded later.

2. Previous Probabilistic Results for SAT

The results of [7], [8] and [9] are based on the probabilistic model (or a closely related model) which we refer to as $M_1(n,r,p)$. According to $M_1(n,r,p)$ each of n clauses is constructed independently by including each of r boolean variables in the clause, independently, in the following way: with probability p the positive literal associated with the variable is placed in the clause and with probability p the negative literal associated with the variable is placed in the clause. Thus, both literals associated with the same variable are in the clause with probability p^2 and neither literal associated with the same literal is in the same clause with probability $(1-p)^2$. According to the results of [7],[8] and [9] a collection of variants of DPP and other algorithms solves SAT in polynomial average time under $M_1(n,r,p)$ if *a*) $n \leq t\ln(r)$, *b*) $p \leq t(\ln(r)/r)^{3/2}$, *c*) $n \geq \exp(\epsilon r)$ and *d*) $p \geq \epsilon$ where t is a positive constant equal to the exponent of the polynomial which bounds the average complexity and ϵ is any constant greater than zero. In this letter it is shown that algorithm A_1 finds a solution to a random instance of SAT under $M_1(n,r,p)$ with probability tending to 1 when $p \geq \ln(n)/r$ and A_2 verifies that no solution exists with probability tending to 1 when $p \leq \ln(n)/(2r)$ and n is not exponential in r. If n is exponential in r then exhaustive search will determine satisfiability in polynomial time.

3. Analysis of A_1 and A_2

The result for A_1 requires finding the probability that a random truth assignment is a solution to a random instance I of SAT. The probability that a truth assignment, T, to r variables does not satisfy a random clause, c, is the probability that none of the r literals made *true* by T are in c and this is $(1-p)^r$. Therefore the probability that T satisfies n independent clauses is $(1-(1-p)^r)^n$. If, for large r and n, $p = a \cdot \ln(n)/r$, where a is a constant and n is not exponential in r, then $(1-p)^r$ is approximately $exp(-a \cdot \ln(n)) = n^{-a}$. But $(1-n^{-a})^n$ approaches 1 as n gets large when $a > 1$ and approaches 0 when $a < 1$. Thus T is a solution to a random instance of SAT, with probability tending to 1, when $\lim_{n,r\to\infty} p > \ln(n)/r$ and A_1 finds a solution to a random instance of SAT in polynomial time with probability tending to 1 for the same limiting range of p.

Since any instance of SAT with a null clause is unsatisfiable (this is true under the interpretation given in [7],[8] and [9]), algorithm A_2 verifies that some instances of SAT are unsatisfiable in polynomial time. The analysis of A_2 requires finding the probability that a null clause exists in a random instance of SAT. The probability that none of $2r$ literals is in a random clause is $(1-p)^{2r}$. Therefore, the probability that at least one literal is in c is $1-(1-p)^{2r}$ and the probability that at least one literal is in every clause of I is $(1-(1-p)^{2r})^n$. Thus, the probability that

there is no zero-literal clause in I is $1-(1-(1-p)^{2r})^n$. If, for large n and r, $p = a \cdot \ln(n)/r$ and n is not exponential in r then $(1-p)^{2r}$ is approximately $\exp(-2a \cdot \ln(n)) = n^{-2a}$. But $1-(1-n^{-2a})^n$ approaches 1 as n gets large when $a < 1/2$. Thus, I contains a zero-literal clause, with probability tending to 1, when $\lim_{n,r\to\infty} p < \ln(n)/(2r)$. Since A_2 runs in time bounded by a polynomial in n and r, A_2 verifies that no solution to I exists in polynomial time with probability tending to 1 when $\lim_{n,r\to\infty} p < \ln(n)/r$.

4. Concluding Remarks

The results obtained assuming the constant-density model $M_1(n,r,p)$ are interesting in light of some probabilistic results obtained for the constant-component-size model $M_2(n,r,k)$: each of n clauses is selected independently and uniformly from the set of all possible k-literal clauses that can be constructed from r variables. Under $M_2(n,r,k)$ the algorithms of [7], [8] and A_1 and A_2 require exponential average time for any constant limiting ratio of n to r (see [6]) but the algorithm of [5] finds solutions to random instances of SAT when $\lim_{n,r\to\infty} n/r < 1$ for any $k \geq 3$. Furthermore, algorithms based on the unit-clause rule find solutions to random instances of SAT under $M_2(n,r,k)$ in polynomial time with probability approaching 1 when $\lim_{n,r\to\infty} n/r < O(2^k/k)$ (see [2] and [3]).

The apparent disparity between the two sets of results is due in large part to a property of the constant-density model for SAT: the constant-density model allows zero-literal clauses with about the same probability that it allows one-literal or two-literal or m-literal clauses, m a constant. Therefore, a random instance generated according to $M_1(n,r,p)$ has no solution with high probability unless the average number of literals per clause is increasing with n and r. But, if the average number of literals per clause is increasing with n and r then the probability that a clause is satisfied by a random truth assignment is increasing with n and r fast enough so that the probability that a random truth assignment satisfies a random instance of SAT under $M_2(n,r,p)$ is high in this case.

The same phenomenon can be observed when constant-density models are used to generate random graphs as instances for certain NP-complete problems on graphs. For example, consider the problem of finding a hamiltonian circuit in a given graph. This problem was studied in [1] under the assumption of the following constant-density model: n vertices are given and the probability that an edge (undirected) connects any pair of vertices is p independent of any other edges appearing in the graph. Under this model the probability that a vertex is isolated (no edges between this vertex and any other) is about the same as the probability that a vertex connects to one other vertex or two other vertices or m other vertices where m is fixed. But a graph containing an isolated vertex has no hamiltonian circuit. Therefore, under the constant-density model, a random graph does not have a hamiltonian circuit unless the average number of edges per vertex is increasing with n. But then the chance of a hamiltonian cicuit being in a random graph is great

and one may be found with high probability using the algorithm of [1]. It would be interesting to see how well the algorithm of [1] works probabilistically under the constant-component size model (known as the constant-degree model) for random graphs.

In view of the observations above and the observation that algorithms which perform well in probability on constant-component-size models also perform well in probability on constant-density models it seems reasonable that constant-component-size models be used in preferrence to the constant-density models when obtaining probabilistic performance benchmarks.

5. References

1. Angluin, D. and Valiant, L., "Fast probabilistic algorithms for hamiltonian circuits and matchings," *Proc. 9th annual ACM Symposium on Theory of Computation* (1977), 30-41.

2. Chao, M.T. and Franco, J., "Probabilistic analysis of the unit clause and maximum occurring literal selection heuristics for the 3-satisfiability problem," Tech. Report No. 164, Indiana University (1985).

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. Davis, M. and Putnam, H., "A computing procedure for quantification theory," *J.ACM* **7** (1960), 201-215.

5. Franco, J., "Probabilistic analysis of the pure literal heuristic for the satisfiability problem," *Annals of Operations Research* **1** (1984), 273-289.

6. Franco, J. and Paull, M., "Probabilistic analysis of the Davis- Putnam Procedure for solving the satisfiability problem," *Discrete Applied Mathematics* **5** (1983), 77-87.

7. Goldberg, A., Purdom, P.W. and Brown, C.A., "Average time analyses of simplified Davis-Putnam Procedures," *Information Processing Letters* **15** (1982), 72-75.

8. Purdom, P.W., "Search rearrangement bactracking and polynomial average time," *Artificial Intelligence* **21** (1983), 117-133.

9. Purdom, P.W. and Brown, C.A., "The pure literal rule and polynomial average time," to appear in *SIAM J. Comput.*.