Easy Problems are Sometimes Hard*

Ian P. Gent	Toby Walsh
Department of AI	INRIA-Lorraine
University of Edinburgh	615, rue du Jardin Botanique
80 South Bridge, Edinburgh	54602 Villers-les-Nancy
United Kingdom	France
I.P.Gent@edinburgh.ac.uk	walsh@loria.fr

March 1994

Abstract

We present a detailed experimental investigation of the easy-hard-easy phase transition for randomly generated instances of satisfiability problems. Problems in the hard part of the phase transition have been extensively used for benchmarking satisfiability algorithms. This study demonstrates that problem classes and regions of the phase transition previously thought to be easy can sometimes be orders of magnitude more difficult than the worst problems in problem classes and regions of the phase transition considered hard. These difficult problems are either hard unsatisfiable problems or are satisfiable problems which give a hard unsatisfiable subproblem following a wrong split. Whilst these hard unsatisfiable problems may have short proofs, these appear to be difficult to find, and other proofs are long and hard.

This paper is a revised version of Research Paper 642, available from the department of Artificial Intelligence, Edinburgh. This version is to appear in the journal *Artificial Intelligence*.

*The first author was supported by a SERC Postdoctoral Fellowship and the second by a HCM Postdoctoral fellowship. We thank Alan Bundy, Bob Constable, Pierre Lescanne, Paul Purdom, the members of the Mathematical Reasoning Group at Edinburgh, the Eureca group at INRIA-Lorraine, and the Department of Computer Science at Cornell University for their constructive comments and for rather a lot of CPU cycles. We also wish to thank our anonymous referees who suggested several important improvements to this paper.

1 Introduction

Many techniques for generating problems randomly depend on a natural parameter; for example the average connectivity of a graph or the ratio of the number of clauses to variables in a propositional formula. As this parameter is varied, there is often a phase transition from problems which are almost always soluble to a region where they are almost always insoluble. Furthermore this phase transition is usually associated with problems which are typically hard to solve [2]. In this paper, we show that with SAT, the problem of determining the satisfiability of propositional formulae, there are also regions in which problems are usually easy *but* sometimes extraordinarily hard. The same effect appears to have been observed for graph colouring [8].

Random 3-SAT (this and other problem classes are described in §2) undergoes a phase transition for unsatisfiable to satisfiable when there about 4.25 clauses for every variable [12, 11, 3]. This is generally considered a good method of generating hard instances of SAT. By comparison, the constant probability model is considered to give easy problems since various theoretical results have shown that certain cases can be solved in polynomial average time [13]. In addition, limited experimental studies have reached similar conclusions [12].

One of the aims of this paper is to demonstrate that these conclusions are premature. We will show that extraordinary hard problems can be found in "easy" problem classes like the constant probability model, and in "easy" regions of the phase transition. For example, we found one 100 variable problem in the easy region of the constant probability model that required over 350 million branches to solve using a simplified Davis-Putnam procedure. This was considerably more than the total number of branches required by the same procedure on a *thousand* 100 variable problems from the hard region of random 3-SAT, a supposedly much harder problem class. Whilst problems generated by the constant probability model are indeed typically easy, there is enormous variability between problems generated in the same way. Furthermore, the greatest variability occurs not in a region where about 50% of problems are satisfiable, but in a region where nearly 100% are, and it occurs with both satisfiable and unsatisfiable problems.

2 Propositional Satisfiability, SAT

SAT, the problem of deciding if there is an assignment for the variables in a propositional formula that makes the formula true, is an interesting problem since it was the first computational task shown to be NP-hard [1]. It is also useful since many AI problems can be encoded quite naturally in SAT (*eg.* constraint satisfaction, diagnosis, planning). We consider SAT problems in conjunctive normal form (CNF); a formula, Σ is in CNF iff it is a conjunction of clauses, where a clause is a disjunction of literals. We consider two methods of randomly generating SAT problems, namely random 3-SAT and the constant probability model.

Problems in random 3-SAT with N variables and L clauses are generated as follows: 3 of the N variables are selected for each clause, and each variable is made positive or negative with probability $\frac{1}{2}$.

procedure DP(Σ)
  if Σ empty **then return** satisfiable
  if Σ contains an empty clause **then return** unsatisfiable
  (Tautology rule) **if** Σ contains a tautologous clause c **then**
    return DP($\Sigma - \{c\}$)
  (Unit Propagation) **if** Σ contains a unit clause c **then**
    return DP(Σ simplified by assigning truth value which satisfies c)
  (Pure-literal Deletion) **if** Σ contains a literal l but not the negation of l **then**
    return DP(Σ simplified by assigning truth value which satisfies l)
  (Split-rule) **if** DP(Σ simplified by assigning a variable arbitrarily) is satisfiable
    then return satisfiable
    else return DP(Σ simplified by assigning variable opposite value)

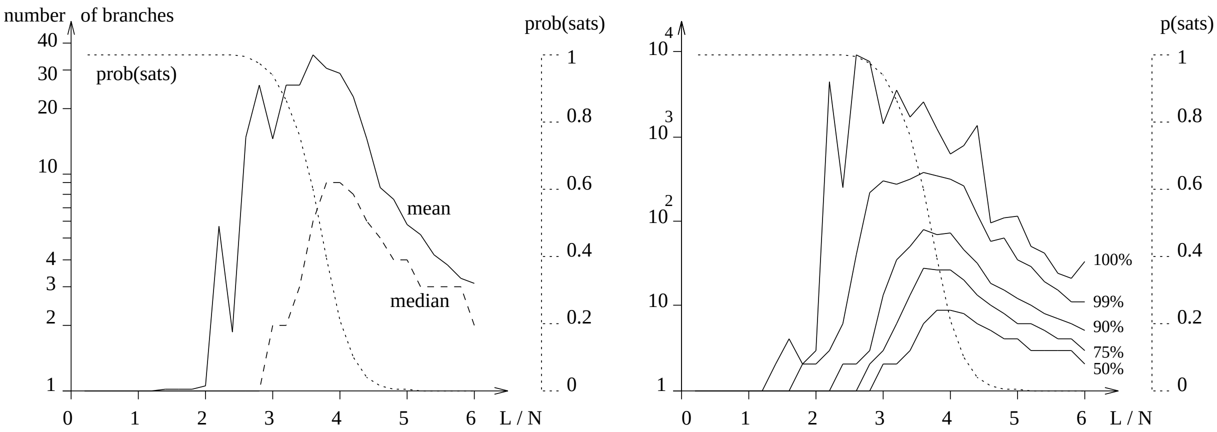
Figure 1: The Davis-Putnam Procedure

In the constant probability model, clause sets with N variables and L clauses are generated according to a parameter p, $0 < p \leq 1$. For each clause, each literal (that is, a variable or the negation of a variable) is included with probability p, independently of the inclusion of other literals. Note that tautologous clauses (containing both a literal and its negation) are allowed, and that clauses can vary in size. For several methods of varying L and p in terms of N, problems from the constant probability model can be solved in polynomial average time (see for example [13]). One source of easiness is that empty and unit clauses (clauses containing zero or one literal) typically makes problems easily unsatisfiable, while making p large enough to exclude them often makes clause sets very easily satisfied. Accordingly, we use a variant of the constant probability model used in [9] in which if an empty or unit clause is generated, it is discarded and another clause generated in its place. The results mentioned earlier do not apply to this model. We call this the "CP" model. We shall contrast our results with those for random 3-SAT.

3 The Davis Putnam Algorithm

A standard procedure for determining satisfiability is due to Davis and Putnam [4]. This is shown in Figure 1. To simplify a clause set with respect to a truth assignment, we delete each clause that is satisfied by the truth assignment, and in every other clause delete any literals that contradict the truth assignment. In [12], a study of the phase transition for random 3-SAT and CP was performed using a simplified (but nevertheless complete) version of this procedure with only the unit-propagation and split-rule. We will consider both this variant (which we will call "unit DP") as well as the full algorithm (which we will call "DP").

The Davis-Putnam procedure is non-deterministic as the literal used by the split-rule is unspecified.[1] In the results given in §4, the split-rule simply chooses the first literal in the first clause in the clause set. Heuristic procedures for determ-



(a) prob(sats), mean, median branches (b) prob(sats) and percentile branches

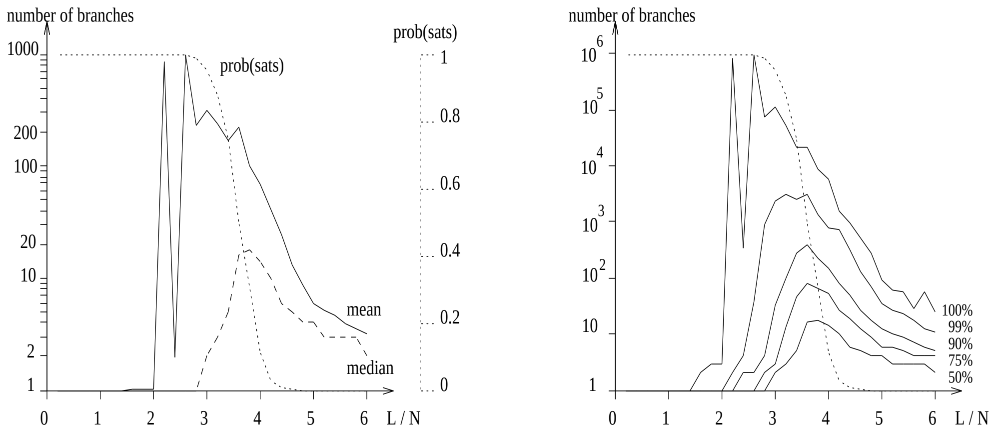
Figure 2: DP, N=100, random CP problems

ining a good literal to split on have been proposed. In §5 we use the Jeroslow-Wang heuristic [10] which estimates the contribution each literal is likely to make to satisfying the clause set. Each literal is scored as follows: for each clause c the literal appears in, $2^{-|c|}$ is added to the literal's score, where $|c|$ is the number of literals in c. The split-rule is then applied to the literal with the highest score.

4 Hard Problems in CP

We first report experiments using the CP model (see §2). In [6], we suggest that the position of the phase transition occurs at fixed L/N when $2\mathrm{N}p$ is kept constant. To aid comparison with results for random 3-SAT, p was varied so that $2\mathrm{N}p$=3. Although this gives an average clause length of about 3.6 after the exclusion of unit and empty clauses, the expected number of models for the same number of variables and clauses is very similar to that of random 3-SAT [6]. We have obtained similar results varying p in other ways, in particular scaling p as $\ln \mathrm{N}/\mathrm{N}$ [5]. This scaling was suggested to us by Paul Purdom as likely to give difficult problems for the CP model.

First, we consider the full Davis-Putnam algorithm, DP. Figure 2 (a) shows results for DP applied to 100 variable CP problems. Each data point represents the result of 1000 experiments, and we varied L/N from 0.2 to 6.0 in steps of 0.2. The dotted line in these graphs indicates the probability of satisfiability. The dashed line represents the median, and the solid line the mean, of the number of branches searched,[2] both plotted using the same logarithmic scale. The median graph shows the standard easy-hard-easy pattern as the number of clauses increases with the hard region being associated with the crossover from satisfiable to unsatisfiable problems. The peak median is only 9 branches at L/N=4.0. This at first supports previous research suggesting that the CP model generates very easy problems compared to random 3-SAT [12]. Note, however, the very different behaviour of



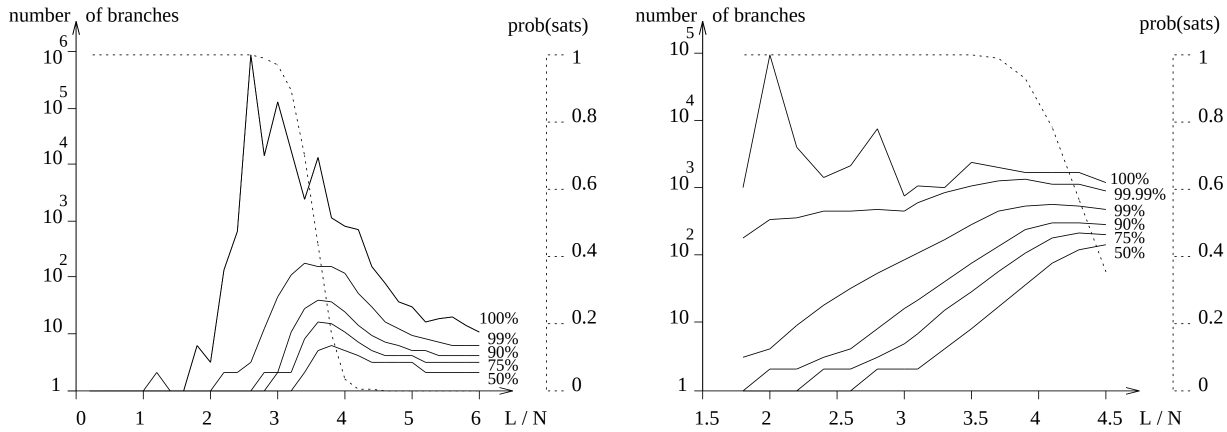
(a) prob(sats), mean, median branches (b) prob(sats) and percentile branches

Figure 3: DP, N=150, random CP problems

the mean in the region of L/N from 2.2 to 3.2, where the median is never above 2 branches, while the mean is an order of magnitude higher and also noisy.

The noise seen in Figure 2 (a) becomes clearer if we examine the worst case behaviour and that of varying percentiles. This is shown in Figure 2 (b). As we tested 1000 problems, the 99% contour, for example, gives the tenth most difficult problem found at each point. As we increase the percentile examined, the contours become more widely separated even on this logarithmic plot, and the peaks move towards lower values of L/N. The 100% contour, i.e. the observed worst case, shows a remarkable variability at small values of L/N, and can be more than an order of magnitude worse than even the 99% contour. The worst case was 9,471 branches at L/N=2.6, a region where 99.4% of the problems generated were satisfiable. At L/N=4.0, the point of worst median performance, the worst case was just 623 branches, an order of magnitude smaller. Comparison of Figure 2 (a) and (b) clearly shows that worst case behaviour is responsible for almost all the features of the mean in the mostly satisfiable region. Given the noise in the worst case contour, and the influence the worst cases have on the mean, very large sample sizes would be needed to reduce this noise and produce a smooth graph.

With larger problem sizes, this phenomenon becomes even clearer. Figure 3 (a) shows the results of DP applied to 150 variable CP problems, with 1000 experiments performed at each point. The maximum mean number of branches is now *not* seen in the region where approximately 50% of problems are satisfiable, but in a region where a very high percentage are satisfiable. In Figure 3 (b) we plot the breakdown in percentiles for the number of branches used by DP. The worst case was 981,018 branches at L/N=2.6, a region where 99.7% of the problems generated were satisfiable. At L/N=3.8 the point of worst median performance, the worst case was just 8,982 branches, two orders of magnitude smaller.



(a) CP, N=250, DP+Jeroslow-Wang (b) 3-SAT, N= 50, Unit DP

Figure 4: prob(sats) and percentile branches, two problem classes

We obtained very similar results with unit DP, although difficult problems were found with this simpler procedure at even smaller numbers of variables [5]. For example, the worst case satisfiable problem for DP described above with 100 variables, required 350,163,776 branches using unit DP.[3] This was considerably more than the total number of branches required by unit DP on a *thousand* 100 variable problems from the hard region of random 3-SAT.

To summarise, we have seen an incredible variability in the number of branches needed to solve problems generated randomly from the constant probability model without empty and unit clauses, using the Davis-Putnam procedure with and without pure literal deletion. This variability is sufficient to affect mean behaviour by many orders of magnitude. The behaviour is observed in regions where more than 90% of problems are satisfiable.

5 An Improved Algorithm

The versions of the Davis-Putnam procedure used in §4 take a very naive approach to application of the split-rule, simply splitting on the first literal in the first clause. However, even using such sophisticated heuristics as Jeroslow-Wang (see §3) for deciding upon which literal to split, we cannot eliminate variability. Figure 4 (a) shows the results of applying DP with the Jeroslow-Wang heuristic to 250 variable CP problems, with 5000 problems tested at each data point, broken down in percentiles of the number of branches searched. The worst case was 907,614 branches at L/N=2.6, a region where 99.66% of the problems generated were satisfiable. At L/N=3.8, where median performance peaked at only 6 branches, the worst case was 576 branches, three orders of magnitude smaller. The larger problem size and sample size required to see the behaviour shows that the Jeroslow-Wang heuristic does offer great improvements over the naive splitting heuristic, but it seems impossible to avoid highly variable and bad worst case behaviour.

6 Random 3-SAT

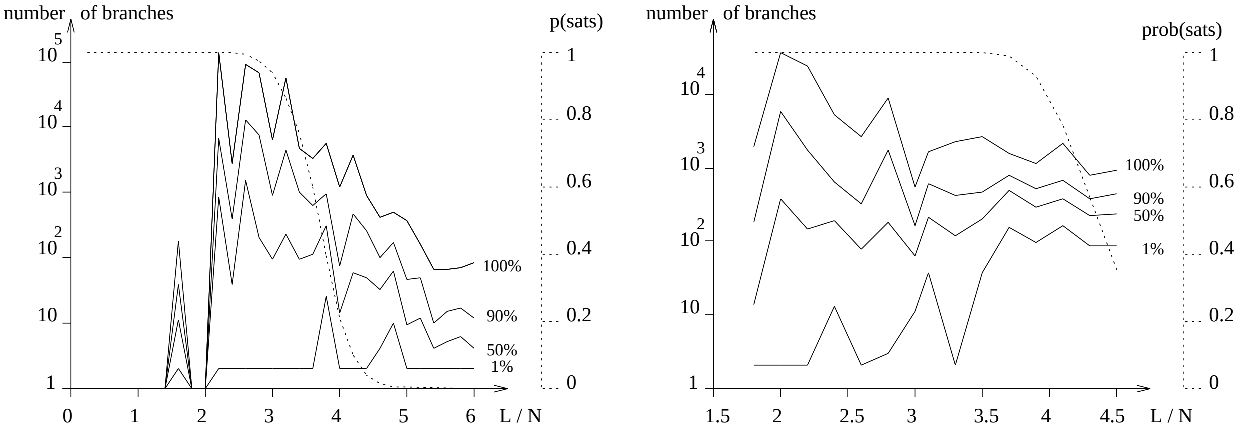
Most recent experimental work on SAT has studied random 3-SAT [12, 3, 11]. Our experiments suggest that exceptionally hard problems are considerably rarer in random 3-SAT than in the CP model, but that they do occur. While in §4, we performed experiments using 1000 problems per data point, we needed 100,000 problems per data point with random 3-SAT. Figure 4 (b) shows the breakdown in percentiles for the number of branches used by unit DP for 50 variable random 3-SAT problems, from L/N=1.8 to 3.0 and 3.1 to 4.5 in steps of 0.2. The worst case was 98,996 branches at L/N=2.0, where 100% of the problems generated were satisfiable. At L/N=4.5, the point of worst median performance, the worst case was just 1,238 branches. However, even the 99.99% contour does not show the dramatic variability visible in the worst case. Whilst hard problems are rarer than in CP, our results with CP suggest that they may nevertheless dominate the mean behaviour of random 3-SAT problems at large N. Unfortunately, current computational limits prevent us from testing this hypothesis.

7 Problem Presentation

To explore in more detail the cause of the the hardness of problems from the mostly satisfiable region, we examined the hard satisfiable problem described in §4 that took unit DP over 350 million branches. We were unable to find any obvious syntactical test, such as the distribution of clause lengths or variable occurrences, that distinguishes it from other problems in this region. We next tested different presentations of this problem.[4] Even using unit DP, a model was found on the first branch with more than half of the presentations of the problem, while worst case behaviour was still very bad.

An interesting possibility is that in the original problem, a single bad split is made early on, leading to a very difficult unsatisfiable problem.[5] Indeed, the first split of the original problem leads immediately to an unsatisfiable subproblem on which unit DP requires all but one of the millions of branches it searched. In addition, this unsatisfiable subproblem often remained hard when presented differently. Unit DP often took millions of branches and 500 presentations of the problem to DP had a median of 1,497 and a mean of 4,550 branches. However, the shortest proof found needed only 2 branches, 4 unit propagations, and no pure literal deletions, and 10% of presentations were solved in as few as 10 branches. This suggests that the variable behaviour observed in this paper depends on the difficulty of search in unsatisfiable problems, and that this search can go badly wrong even when there are short proofs available. These unsatisfiable problems can arise as randomly generated unsatisfiable problems, or as unsatisfiable subproblems of satisfiable problems after incorrect branching at an early stage.

To explore this issue further, we compared the difficulty of different presentations of the worst unsatisfiable problems at different L/N. If the worst problem at a given L/N was satisfiable, we used the first unsatisfiable subproblem found in DP's search. Figure 5 (a) shows the breakdown in percentiles of applying DP to 500 random presentations of the worst CP problems at each point from Figure 2,



(a) CP, N=100, DP (b) 3-SAT, N=50, Unit DP

Figure 5: Percentile behaviour on presentations of worst unsatisfiable problems

while Figure 5 (b) shows the results of applying unit DP to 500 random presentations of the worst 3-SAT problems at each point from Figure 4 (b). As a guide, we reproduce the probability of satisfiability from the earlier figures. The contours of 90% and worst case behaviour show a clear downward trend from low values of L/N to large values on this logarithmic scale, and the median case is always comparatively hard. On the other hand, best case behaviour at low values of L/N is very easy in all cases, and in most cases there is a proof needing only 2 branches. This strongly suggests that although short proofs are available, they can be much harder to find in regions of otherwise easy problems. Additionally, performance on 3-SAT problems in the crossover from satisfiability to unsatisfiability varies by only an order of magnitude, compared to four orders of magnitude at L/N=2.0. This suggests that problems in the crossover are more uniformly hard.

For the problems studied in Figure 5 (a), the ease of the 1% contour shows that hundredfold parallelisation would be sufficient to solve all these problems very quickly. However, it does not follow that this simple approach would continue to work as the problem size is increased. This is hinted at by an examination of the hard 250 variable CP problem reported in §5. As it was satisfiable, we tested different presentations of its first unsatisfiable subproblem. Although the shortest proof again required just 2 branches and 10 unit propagations, DP only found a proof in fewer than 1000 branches for 7% of presentations. This suggests that short proofs may get harder to find as problem sizes increase.

8 Related Work

In an empirical study of random 3-SAT using a variant of unit DP with very effective branching heuristics, Crawford and Auton [3] observed a secondary peak in problem difficulty in regions of high satisfiability for small problems. We believe this second peak is not due to variable behaviour, but more probably correlated with a peak in search depth we observed in the same region [7]. However, they also report anecdotally some problems below the crossover point as hard as those at the crossover point for problems beyond 1000 variables. This is consistent with our results (see §6), as they used a smaller sample size and we have also seen the use of improved heuristics delay the onset of variable behaviour (see §5). This suggests that very hard problems occur in otherwise easy regions of both random 3-SAT and the CP model, but are more easily observable in the CP model.

Hogg and Williams have observed a very similar result for graph colouring [8]. They found that the hardest graph colouring problems were in an otherwise easy region of graphs of low connectivity. The median search cost, by comparison, shows the usual easy-hard-easy pattern through the phase transition.

9 Conclusions

The "easy" region of the both random 3-SAT and the constant probability model can sometimes give problems much harder for the Davis-Putnam procedure than the hardest problems of the "hard" region of the random 3-SAT model. These very hard problems occur in a region where there is at least a 90% chance of problems being satisfiable (and not 50% as had been previously proposed [12]). It is not certain if this difficult region is associated with the satisfiability phase transition (as proposed in [2]), with a second phase transition (as proposed in [8]), or with some other effect. The use of better heuristics appears to delay but not postpone indefinitely the appearance of these difficult problems. Hogg and Williams have observed the same effect in graph colouring [8].

Hard problems from the mostly satisfiable region appear to be potentially useful for testing algorithms as, in the worst case, they can be much harder than problems from the middle of the phase transition. We suggest that these difficult problems are on the knife edge between satisfiablity and unsatisfiability. It can be difficult to determine whether they are satisfiable or unsatisfiable because they have relatively few constraints, even though they are mostly satisfiable. In this region, we have observed that hard unsatisfiable problems do in fact usually have short proofs, but that these proofs can be hard to find. Satisfiable problems can reduce to a very hard unsatisfiable subproblem if the wrong reduction is applied at an early stage. By comparison, in the middle of the phase transition, there is much more uniformity in the cost of proving satisfiability and unsatisfiability.

We conjecture that the phenomena discussed in this paper will occur for many other, and possibly all, complete algorithms for SAT. Furthermore, it is very likely that similar phenomena will be observed in other classes of randomly generated SAT problems and in many other NP-hard problems. The hardness of many NP-hard problem classes may lie with rare but exponentially hard problems that are found in otherwise easy regions.

References

[1] S.A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computation*, pages 151–158, 1971.

[2] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of the 12th IJCAI*, pages 163–169, 1991.

[3] J.M. Crawford and L.D. Auton. Experimental Results on the Crossover Point in Satisfiability Problems. In *Proceedings of the 11th National Conference on Artificial Intelligence.* American Association of Artificial Intelligence, 1993.

[4] M. Davis and H. Putnam. A computing procedure for quantification theory. *JACM*, 7:201–215, 1960.

[5] I.P. Gent and T. Walsh. Easy Problems are Sometimes Hard. Research Paper 642, Department of AI, Edinburgh University, June 1993.

[6] I.P. Gent and T. Walsh. The SAT phase transition. Research Paper 679, Department of AI, University of Edinburgh, 1994.

[7] I.P. Gent and T. Walsh. The Hardest Random SAT Problems, Research Paper 680, Department of AI, Edinburgh University, January 1994.

[8] T. Hogg and C. Williams. The Hardest Constraint Problems: A Double Phase Transition. PARC preprint, Dynamics of Computation Group, Xerox Palo Alto Research Center, June 1993.

[9] J. N. Hooker and C. Fedjki. Branch-and-cut solution of inference problems in propositional logic. *Annals of Mathematics and AI*, 1:123–139, 1990.

[10] R. E. Jeroslow and J. Wang. Solving propositional satisfiability problems. *Annals of Mathematics and AI*, 1:167–187, 1990.

[11] T. Larrabee and Y. Tsuji. Evidence for a Satisfiability Threshold for Random 3CNF Formulas. Technical Report UCSC-CRL-92-42, University of California, Santa Cruz, 1992.

[12] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *AAAI-92: Proceedings of the 10th National Conference on Artificial Intelligence.* AAAI Press/The MIT Press, 1992.

[13] P.W. Purdom Jr. and C. A. Brown. The pure literal rule and polynomial average time. *SIAM Journal of Computing*, 14(4):943–953, 1985.

Footnotes

1. The conditional part of the split-rule assigns a truth value to a particular variable v. If this variable is set to true, then we say that the split-rule uses the literal v. If this variable is set to false, then we say that it uses the literal $\neg v$.

2. By the number of branches we mean the number of leaf nodes in the search tree.

3. On a 30 Mips machine this one problem took our COMMON LISP implementation about a week.

4. These were constructed as follows: the clauses were permuted randomly; the literals in each clause were permuted randomly; the names of the literals were permuted randomly; and each literal, with probability 1/2, was negated throughout the clause set. The resulting problem is isomorphic to the original.

5. We thank Alan Bundy for suggesting this to us.