Some Pitfalls for Experimenters with Random SAT

David G. Mitchell [1], Hector J. Levesque [2]

Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada

Abstract

We consider the use of random CNF formulas in evaluating the performance of SAT testing algorithms, and in particular the role that the phase transition phenomenon plays in this use. Examples from the literature illustrate the importance of understanding the properties of formula distributions prior to designing an experiment. We expect this to be of increasing importance in the field.

1 Introduction

Satisfiability testing lies at the core of many computational problems and because of its close relationship to various reasoning tasks, this is especially so in Artificial Intelligence. Randomly generated CNF formulas are a popular class of test problems for evaluating the performance of SAT testing programs. Not surprisingly, the choice of formula distribution is crucial to the validity of any investigation using random formulas. In [23], we argued that some families of distributions were more useful sources of test material than others, and suggested choosing formulas from the "hard region" associated with the satisfiable–to–unsatisfiable phase transition which occurs as the number of clauses is increased. Here we make this concrete, by presenting examples from the literature of experiments where sufficient consideration was not given to the properties of the formulas used. In most cases, the test formulas were implicitly assumed to be challenging in some way, or at least to have hardness dependent on a particular parameter, which we show in further experimentation not to be the case.

[1] Email: mitchell@cs.toronto.edu. Work carried out while at Simon Fraser University, Burnaby, Canada, supported by the Institute of Robotics and Intelligent Systems and an EBCO/EPIC Graduate Scholarship.

[2] Fellow of the Canadian Institute for Advanced Research. Supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

Our examples are based on currently popular "unstructured" random formulas. The value of these as test material – even when sampled from the "hard region" – can be questioned on the grounds that they may not be much like real problems [4,17,21], but some of these distributions appear challenging for a variety of methods, and we expect their use to continue. Moreover, most available alternatives are either puzzle-type problems (such as cross-word puzzles) or other distributions with no *a priori* greater validity. Nonetheless, some classes of "structured" random formulas are likely to become more popular, and in Section 8 we discuss the implications of our examples for such distributions.

In Section 2 we describe our SAT testing algorithm and the formula distributions under consideration, and in Section 3 we survey some properties of these formulas. Sections 4 to 7 examine individual experiments, in light of what we now know about the formulas used. Additional data is presented as needed. In Section 8 we summarize.

2 Materials and Methods

We assume the reader is familiar with the problem SAT as defined in [12]. The data for our analysis was produced by testing randomly generated SAT instances with a simple version of the Davis-Putnam Procedure [8]. Our procedure, which we refer to as "DP", is shown in Figure 1. In the figure, ω denotes a CNF formula and $\omega\backslash p$ denotes the formula that results when we simplify ω by setting the variable p true. We assume the variables are ordered, and let $vars(\omega)$ denote the set of variables mentioned in ω, and $min\{vars(\omega)\}$ the least variable mentioned in ω.

procedure DP(ω)
    if ω is empty **then return** *satisfiable*
    else if ω contains an empty clause **then return** *unsatisfiable*
    else if ω contains a unit clause $\{l\}$ **then return** DP($\omega\backslash l$).
    else
        let $p := min\{vars(\omega)\}$
        if DP($\omega\backslash p$)=*satisfiable* **then return** *satisfiable*
        else return DP($\omega\backslash\neg p$)
end DP

Fig. 1. The procedure DP.

DP is essentially the splitting variant of the Davis-Putnam Procedure as described in [7], but without the pure literal rule. It uses no heuristics other than unit propagation. This is perhaps the simplest complete SAT testing procedure that performs well enough to be useful, and so provides a kind of baseline for performance of many related methods. We count each recursive DP call (or equivalently, each simplification) as a step. We report median number of DP steps to find a satisfying assignment, if there is one, or report failure otherwise. (For most of the distributions used here, the median number of steps is highly correlated with mean, and also with mean and median run times.) We also report the proportion of formulas which are satisfiable. Each point in a graph represents a sample statistic based on 5000 formulas.

We examine three families of random CNF formulas. Each family has three parameters; the number of variables n, the number of clauses m and for each n, a distribution of clauses over n variables. We will see that the constructed parameter $c = m/n$, the ratio of clauses to variables, is often more useful than m. The families are as follows.

- The widely studied fixed clause length family, which we call "random k-SAT". Clauses are selected uniformly at random, with replacement, from the set of all $2^k\binom{n}{k}$ non-trivial [3] clauses of length k defined over n variables.
- The "constant-density" distributions. In the simplest version, a clause is constructed by including each of the $2n$ literals with some probability p (which may be a function of n). Clause lengths are binomially distributed, with expected length $2np$. It is often useful to study these formulas in terms of expected clause length k, in which case we set $p = k/2n$. Since empty clauses, unit clauses and trivial clauses are easy to simplify out [4], experimenters began using variants in which certain clause types were disallowed (in which case expected clause length is altered slightly by rejection of forbidden clause types). In Section 4 we consider the case where unit clauses are permitted, but trivial and empty clauses are forbidden. In Sections 3 and 5 we consider the case where empty, unit and trivial clauses are forbidden. We call this version random $\mathcal{E}k$-SAT (using $\mathcal{E}$ to denote that k is an expectation).
- Distributions with clause lengths distributed uniformly over some fixed range [k,l]. We will call such distributions "[k,l]-SAT".

[3] A clause is trivial if it contains both a variable and its negation.

[4] Moreover, Wu and Tang [27] extending work by Franco [9] and others presented an algorithm which solves instances of this family with probability not less than $1 - \epsilon$, for any constant $\epsilon > 0$, in polynomial expected time. See also [24,2].

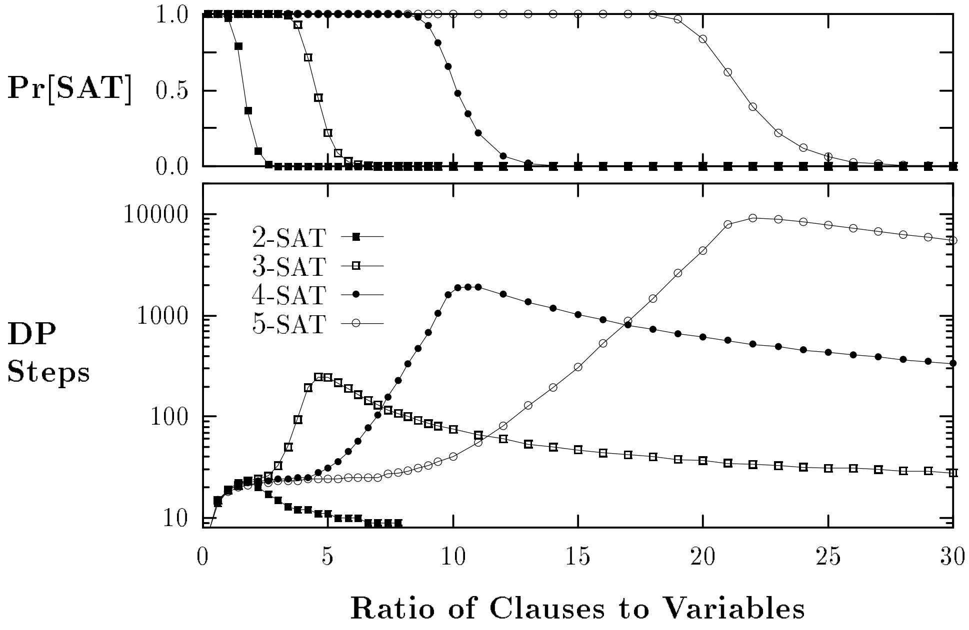

Fig. 2. Median DP steps for random k-SAT: varying c for selected values of k.

3 Patterns in Random SAT

In this section we will survey some basic (empirical) properties of random formulas for reference in following sections.

3.1 Random k-SAT

The upper graph of Figure 2 shows the proportion of random k-SAT formulas with $n = 25$ variables which are satisfiable. There is one curve for each clause length $k \in \{2, 3, 4, 5\}$, and for each curve the ratio of clauses to variables c is varied from $1/5$ to 30. As has been shown before, at small ratios almost all formulas are satisfiable and at high ratios almost all are unsatisfiable. For each k there is some range of c values over which the proportion of satisfiable formulas changes from almost 1 to almost 0. The location of this "transition region" varies with k, but for each k it occurs at approximately the same ratio for all values of n [5,14,20]. The asymptotic location of the transition region is analytically bounded above and below: For $k = 3$, the currently known bounds are $c = 4.758$ [18] and $c = 3.003$ [10] respectively.

The lower graph of Figure 2 shows the median number of DP steps required to test the same formulas. Note the logarithmic y-axis, necessitated by the dramatic increase of peak difficulty with increasing k. As expected, the peak at each k lines up with the corresponding transition region. We call the part of the curve near the peak the "hard region" (to distinguish it qualitatively from the regions far from the transition, which for small n are *relatively* easier. This is not to say there are no hard formulas in the "easy region", nor that the "hard region" is necessarily hard in any rigorous sense). Moving to the left from the hard region, there is a range of ratios, for each curve, over which difficulty changes little as the ratio is varied, and which we will refer to as the "plateau" region. In this region, the number of DP steps is just less than 25 (the number of variables) and almost no backtracking occurs. Roughly speaking, DP just assigns values to variables by guessing and using unit propagation. We refer to the region below $c = 1$, where difficulty drops off quickly with decreasing c, as the "shoulder" region. When c is increased beyond the transition region, difficulty gradually decreases. This general pattern is also found for larger k than is shown here (although we do not know if it holds for very large k).

Random 2-SAT

Observe in Figure 2 that as c is increased the difficulty of random 2-SAT increases only to the level of the plateau region and then begins to decrease again. Random 2-SAT does not exhibit a hard region in the sense that is seen when k is larger [5], which is perhaps not surprising since 2-SAT is solvable in time $O(n + m)$ [1]. Table 1 shows median DP steps, and the ratio of median DP steps to n, to test random 2-SAT with $c = 1$ and n up to 250. At least over this range the median number of steps is smaller than n, and grows no faster than n. DP requires time $\Theta(2^n)$ in the worst case for random 2-SAT, but this data suggests it may be linear on average.

n	25	50	75	100	125	150	175	200	225	250
DP steps	19	37	55	74	92	110	128	146	164	182
steps/n	0.76	0.74	0.73	0.74	0.74	0.73	0.73	0.73	0.73	0.73

Table 1
DP performance on random 2-SAT when $c = 1$

Location of the Peak

We are usually interested in performance for increasing n, rather than just at a particular n. Figure 3 shows the effect of increasing n on the curves for random 3-SAT. The transition region becomes narrower (that is, occurs over a smaller range of c) as n is increased. The peak hardness for DP occurs at approximately the point where 40% of the formulas are satisfiable, and shifts

[5] Cheeseman et al. [3] hypothesized that phase transitions might not occur for problems in **P**. Random 2-SAT is one of several counterexamples, but the present observation suggests a revised version of the hypothesis.

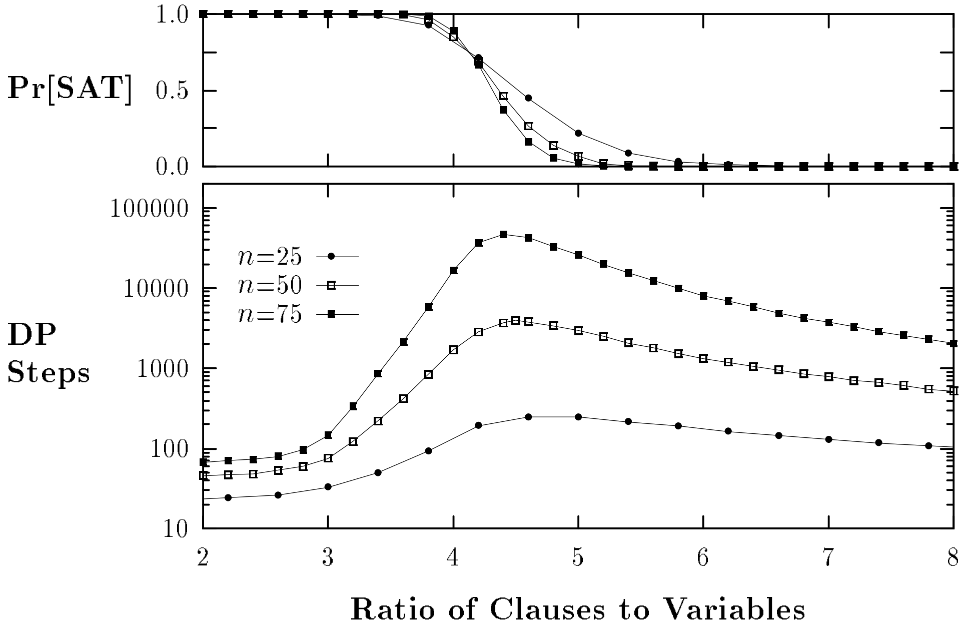

Fig. 3. DP steps for random 3-SAT: selected values of n.

slightly to the left as n is increased, in correspondence with the shift of that point.

Since empirical tests measure difficulty for a particular algorithm on a given set of problems, we expect the location of peak hardness to be somewhat procedure dependent. For example, adding a heuristic to DP which tends to do better on satisfiable instances but poorer on unsatisfiable instances would result in a peak further to the right, because the average performance is a weighted average of performance on satisfiable and unsatisfiable subsets, and this weighting changes with c. In [23] we suggested that peak hardness of random 3-SAT for the Davis-Putnam Procedure was near the ratio where about half the instances were satisfiable, but the algorithms in each of [23,22,5] appear to peak at slightly different ratios.

3.2 Random $\mathcal{E}k$-SAT

Figure 4 shows the proportion of satisfiable formulas and the median DP steps for testing random $\mathcal{E}k$-SAT with $n = 25$ variables, for $k \in \{2, 3, 4, 5\}$ and c varied from 1/5 to 12. (Because of length restrictions, random $\mathcal{E}2$-SAT and random 2-SAT are identical distributions.) As with random k-SAT, the four curves all converge to the same shoulder shape at small c, in this case when $c \leq 2$, and for each $k \geq 3$ there is an easy plateau region, a hard region, and a trailing off of difficulty on the right. As with the fixed clause length formulas, the peaks and transition points shift right with increased k, and the peak hardness also increases with k. The most significant difference between

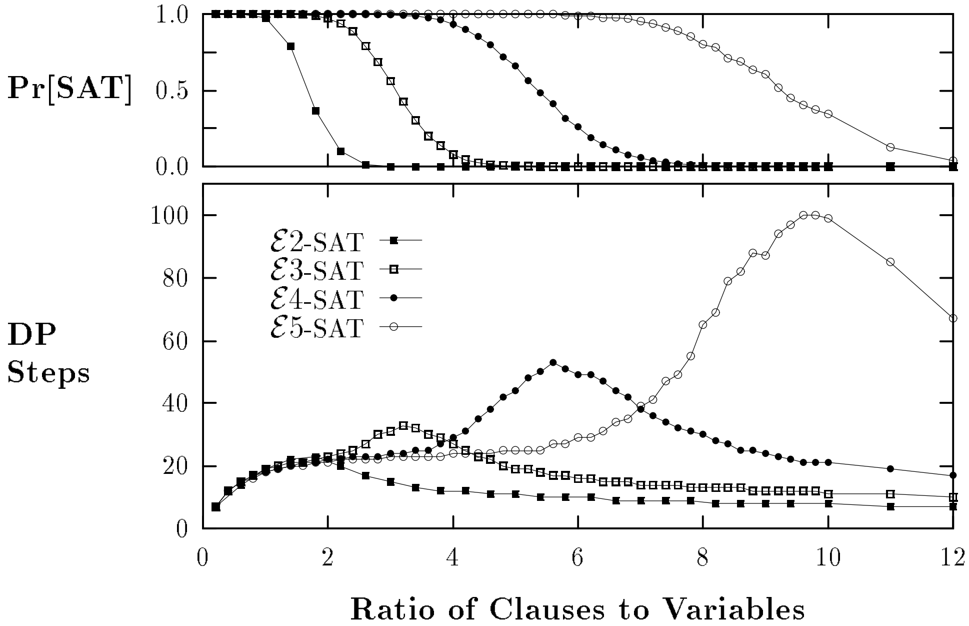

Fig. 4. Median DP steps for random $\mathcal{E}k$-SAT: varying c for selected values of k.

these formulas and random k-SAT, as has been noted before, is that these are very much easier. The difference is a factor of 100 for clause lengths of 5, and increases with increasing n. Another difference is that the transition regions are shifted to somewhat lower ratios for the same average clause lengths. Since these distributions have a few very long clauses, most clauses are shorter than the average clause length, which apparently lowers the transition region also.

3.3 [k,l]-SAT

The general patterns shown above also occur for [k,l]-SAT, albeit with the added complexity of an additional parameter (random k-SAT is the special case of [k,l]-SAT when $k = l$). We content ourselves in this case with presenting only the particular data needed, in Sections 4 and 6.

4 No Hard Region

In Section 3.1 we noted that random 2-SAT did not exhibit a hard region, in the usual sense, and that it can be solved very efficiently even in the worst case. In this section, we will consider experiments using two distributions over **NP**-complete sets of formulas which also do not appear to have hard regions, and which our data suggests are as easy on average as random 2-SAT. As a consequence, we conclude that the data from the experiments described below provides no evidence that the procedures are effective on non-trivial problems.

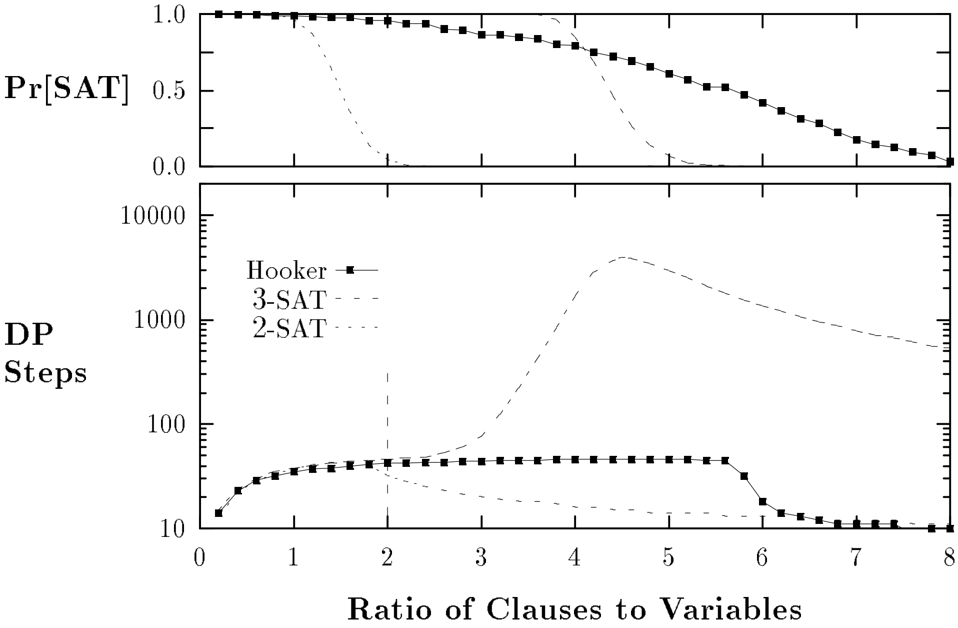

Fig. 5. Median DP steps for formulas used by Hooker.

Hooker [17] compared the performance of a resolution-based procedure with that of one based on cutting planes, using constant density formulas with unit clauses allowed, but empty and trivial clauses prohibited. In this case, p was set so that the expected clause length was 5, and formulas were generated with $c = 2$, and $n \in \{10, 20, 30, 50\}$, and also with $n = 20$ and $c \in \{1, 2, 3, 4, 6, 32\}$. We generated formulas from the same distribution and tested them with DP. Figure 5 shows the median DP steps for these formulas (labeled "Hooker" in the figure), with $n = 50$ and c being varied from 1/5 to 8. Corresponding curves for random 3-SAT and random 2-SAT, also with $n = 50$, are included for comparison.

There is no evidence of a hard region for Hooker's formulas, and the median number of DP steps to test these formulas was less than 50 at every ratio (we tested up to $c = 50$). We repeated the experiment with $n = 100$, and found exactly the same pattern. Hooker noted that the cutting plane algorithm required only a very small number of iterations to solve these problems, while his resolution procedure performed so poorly that he was unable to report completion times for most conditions. Yet DP – which can be expressed as a resolution strategy [15] – required almost no backtracking (less than 3 backtracks on average) to test them, and consistently solves them in a small fraction of a second.

Gallo and Urbani [11] compared the performance of several enhancements to the Davis-Putnam Procedure on [1,7]-SAT. We again generated a range of these formulas and tested them with DP. Figure 6 shows the median DP steps and proportion satisfiable for [1,7]-SAT formulas with $n = 50$ variables (and corresponding curves for [2,7]-SAT, random 2-SAT and random 3-SAT).

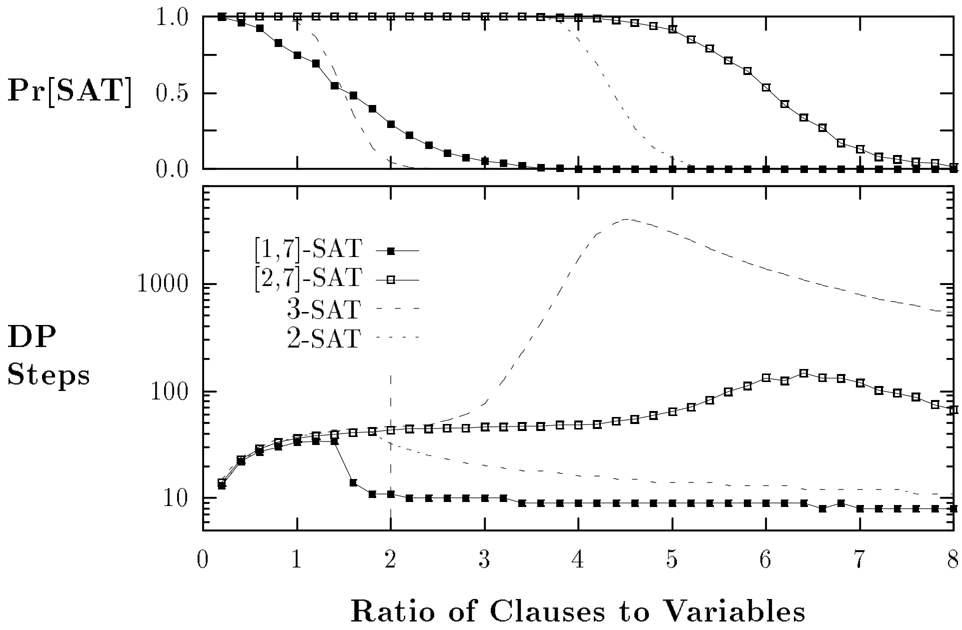

Fig. 6. Median DP steps for [1,7]-SAT and [2,7]-SAT.

The [1,7]-SAT formulas exhibit essentially the same behavior as random 2-SAT, with the usual "shoulder" pattern at the left, but no evidence of a hard region. We confirmed this behavior up to $n = 1000$, where the peak remains smaller than that of random 2-SAT. Virtually no backtracking is required to test these formulas.

We conclude that formula distributions such as [1,7]-SAT and those used by Hooker in [17] are no harder on average than random 2-SAT, and thus of little use in SAT algorithm evaluation.

5 Missing the Hard Region

We observed in Section 3 that random $\mathcal{E}k$-SAT does exhibit a hard region, although orders of magnitude easier than comparable random k-SAT formulas. In this section we will see that experimenters have often used formulas from this distribution, but from below the hard region, which are trivial in spite of their large size.

Gu [16] and Kamath et al. [19] evaluated their procedures (based on local search and interior point programming, respectively) by testing large random $\mathcal{E}3$-SAT formulas, Gu with $n \in \{50, 500, 1000\}$ and Kamath with $n = 1000$, in all cases with $c = 2$. We generated and tested random $\mathcal{E}3$-SAT with $n \in \{50, 500, 1000\}$, and show the results in the upper graph of Figure 7. Also shown, for comparison, is the curve for random 3-SAT with $n = 50$. The

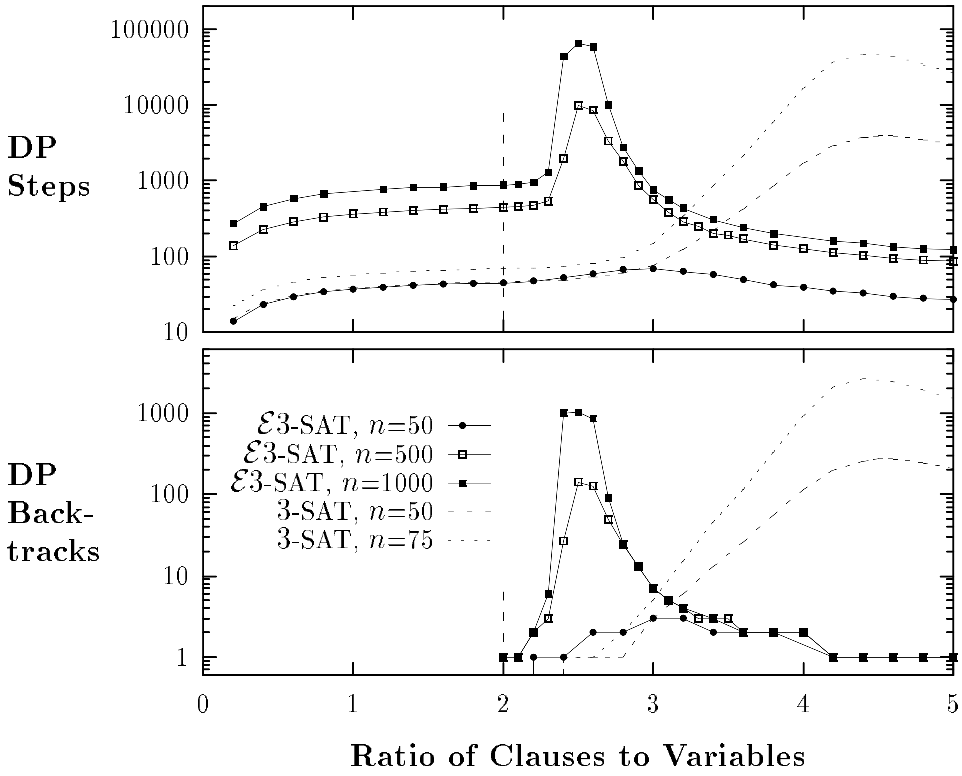

Fig. 7. Median DP steps and backtracks for random $\mathcal{E}$3-SAT.

experiments at $c = 2$ clearly fall to the left of the hard region. To confirm our intuition that testing these requires only guessing and unit propagation, we plotted the median number of backtracks in the lower graph of Figure 7. The median number of backtracks at $c = 2$ was 1 for $n = 500$ and 0 for $n = 1000$: even with fairly large n, these formulas are extremely easy. [6]

Both Gu and Kamath et al. also used formulas from random $\mathcal{E}$10-SAT. Gu tested formulas with various values of n up to 5000 and c up to 10, and Kamath et al. used formulas with $n = 1000$ and c up to 32. We expected that peak difficulty for these would be substantial, but also that the hard region should occur at quite a high value of c, probably in the hundreds. We generated and tested random $\mathcal{E}$10-SAT formulas with $n = 50$ and c varied from 1 to 50, with 1000 samples per point. We do not include a graph: the curve is essentially the same shape as the left one-fourth of a curve in the upper graph of Figure 7. The

[6] Truth in advertising: Although the vast majority of these formulas required 0,1 or 2 backtracks, the *mean* number backtracks is in the thousands due to a very few formulas which are harder (but still trivial compared with similarly sized hard random 3-SAT formulas). The harder formulas occur so rarely we would not expect to see them in studies with small sample sizes such as the examples in this section. Gent and Walsh [13] have studied apparently harder, but rarer still, formulas in this region. These are interesting in themselves, but we believe they occur even more rarely as n gets large, and do not make the distribution on the whole useful.

shoulder region appears up to about $c = 8$, and from $c = 8$ to $c = 50$ the curve is nearly flat. At $c = 50$ there is no evidence of approaching the hard region: all 1000 formulas tested were satisfiable, and essentially no backtracking was required to test these formulas at any ratio below 35, where the mean number of backtracks was about 1. We also tested random $\mathcal{E}$10-SAT formulas with $n = 1000$ variables and c ranging up to 10. At $c = 10$, the curve of median DP steps for these formulas has only just passed the shoulder region. About 90% of the formulas at $c = 10$ required zero backtracks to test, and only one of the 1,000 formulas we tested required as many as three. At lower ratios even fewer backtracks were needed. These formulas, despite being very large, are trivial to test on average.

6 Interaction of Parameters

In [6] d'Anjou and his colleagues report tests using a variety of random formulas to demonstrate the performance of a Boltzmann Machine method for SAT testing (which we will refer to as BM). Their data is presented as evidence that BM solves SAT, on average, in time "nearly linear" in the number of variables, regardless of clause length or number of clauses. We will see that, in spite of the authors' methodical manipulation of all distribution parameters, the data does not support this conclusion.

To show run time of BM is independent of clause size d'Anjou et al. tested random k-SAT formulas with fixed n and m, and observed the effect on run time of varying k. The range for k was $k \in \{2, 5, 6, 7\}$, and the experiment was repeated for each $n \in \{10, 15, 20, 25\}$. In all cases the formulas had $m = 25$ clauses. For each value of n, BM took essentially the same time for all choices of k. The clause-to-variable ratio for all of these test sets is in the range $1 \leq c \leq 2.5$. They all fall in the plateau region for DP (*cf* Figure 2), so if we performed this same experiment with DP instead of BM, we would also find no effect of k. But we know that in the hard region, k has a dramatic effect on hardness for DP. Just as importantly, if the same experiment design had been used, but with a larger value chosen for m so that some tests were in the transition region, the results would be just as suspect. For example, if we picked $n = 25$ and $m = 250$ (so $c = 10$), we would find that increasing k led to a dramatic *decrease* in DP steps!

To show run time of BM is independent of the number of clauses in a formula, d'Anjou et al. tested formulas from [2,7]-SAT with $n = 20$ variables and m varied from 25 to 55. The average time for BM increased only very slightly as m was varied from 25 to 55 (although the variance did increase substantially). The range of clause-to-variable ratios for this experiment is from 1.25 to 2.75. As Figure 6 shows, [2,7]-SAT does exhibit the easy-hard-easy pattern, but the number of steps for DP to test these formulas does not change significantly across the range of ratios tested by d'Anjou et al.. Again, if we performed this experiment with DP we would get the same results. We do know, however, that the performance of DP *is* affected by the number of clauses. (To a large extent, and in a more general form, this is what this volume is all about.) Manipulation of m will detect the easy-hard-easy pattern, but only if the range spans the transition region. Anomalous results will occur with other uninformed choices too. For example, varying m from 130 to 160 in this experiment would produce a decrease in difficulty with increasing m.

Believing BM to be unaffected by clause length and number of clauses, d'Anjou et al. concluded that they could demonstrate the scaling of run time by varying n, while keeping clause length distribution and m constant. Thus, to support their main claim they tested formulas from [2,7]-SAT with $n \in \{10, 15, 20, 25, 30, 35, 40, 100\}$, in every case with $m = 25$ clauses. The times for BM to solve these problems, as a function of n, fit very closely to a linear curve. Once again the range of ratios tested (from 2.5 down to 0.25 as the number of variables was increased from 10 to 100) falls within the easy region [7] (*cf* Figure 6). If m is fixed, then increasing n decreases c, so picking a larger value for m would not have helped this experiment. For example, if we repeated this experiment using DP, and with $m = 65$, then at $n = 10$ (so $c = 6.5$) we would be testing in the hard region, but at large n we would be in the easy region (eg., with $n = 40$ we have $c \approx 1.6$).

Our data suggests that none of the formula sets used in [6] are harder than random 2-SAT, and so all experiments reported there are examples of "missing the hard region", as discussed in Section 5. However, there are additional issues involved here. We don't know how BM would perform on hard formulas, but the results in [6] seem to be an artifact of the easy formulas, and unfortunate parameter choices, rather than indicative of positive performance characteristics of BM. Increasing n while m is fixed leads to small values of c, and thus easy instances. Increasing m alone leads to large values of c, and thus easy (relative to the peak) problems. Moreover, increasing k alone shifts the hard area to higher values of c, thus producing easy problems. To generate hard formulas of varying size requires that n and m be varied at the same time, since hard formulas result from an interaction of parameters. Similarly, if we want to change k but obtain formulas with the same relative hardness, or the same proportion satisfiable, we must also change m at the same time. More generally, we see that varying all parameters does not necessarily make an experiment informative about performance of the test procedure, if the effects of those manipulations on the formulas are not to some extent understood

[7] The 100-variable point was suspect *a priori*. There are 200 literals to choose from, but we only expect about 115 literals in a 25 clause formula, so almost half the literals won't even be mentioned.

beforehand.

7 Picking Ratios

When the chosen distribution has a hard region that corresponds to the transition region, it seems a simple heuristic like "test where about half the formulas are satisfiable" might keep us on safe ground. In most experiments we will vary at least one parameter (typically n), but with a fair understanding of parameter interactions, we can often experimentally track the transition region fairly easily. For random k-SAT it may be as simple as knowing the approximate transition ratio for each k (eg., setting $m = 4.25n$ when $k = 3$). In this section we consider whether this approach is adequate in the common situation in which the experimenter cannot precisely determine the location of peak hardness.

To begin, we consider an experiment by Gallo and Urbani [11], in which performance of several SAT procedures were compared on 5 sets of random 3-SAT formulas, for which partial results are shown in Table 2. The table gives the

test	1	2	3	4	5
variables	10	20	30	40	50
clauses	50	100	140	170	200
ratio	5.0	5.0	4.67	4.25	4.0
50% point	4.86	4.55	4.45	4.40	4.36
satisfiable	7	5	2	6	8
time (s)	0.3	1.2	5.1	13.5	32.7

Table 2
Locations of Tests in Gallo & Urbani (1989)

parameter values used, the number of test formulas which were satisfiable (out of 10), and the time in seconds for Gallo and Urbani's implementation of the Davis-Putnam Procedure. Also shown are our own experimental estimates (based on large samples and accurate to two decimal places) of the ratio at which 50% of the formulas will be satisfiable for the given values of n and m.

Gallo and Urbani chose m based on estimates of where about one half of the formulas would be satisfiable, but comparison with the known 50% ratios in Table 2 shows they were somewhat off, probably due to estimating from small sample sizes. We do not know exactly where the peak is for the procedure used in this experiment, but the procedure is almost identical to our DP, and our experience suggests the peaks for the two procedures will be very close. If we

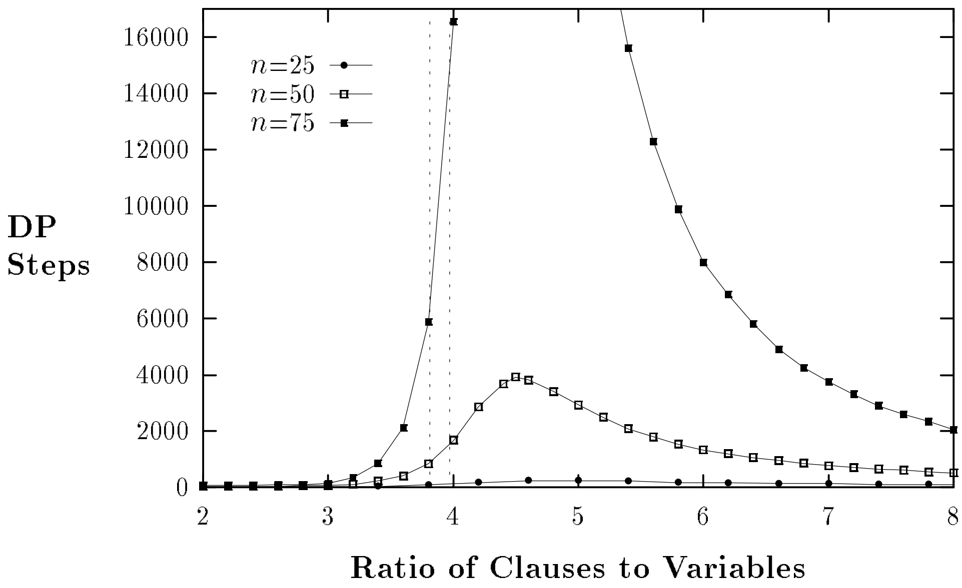

Fig. 8. Median DP steps for random 3-SAT: selected values of n.

assume this to be true, then the choices for c in Table 2 shift from somewhat above the peak, to somewhat below the peak, crossing it near the 3rd data point. Thus, looking at the first 3 points gives an over-estimate of the growth in time with n, and looking at the last 3 points we gives an under-estimate of growth.

Figure 8, shows median DP steps to test random 3-SAT with various n. Although we have not defined the hard region precisely, in some sense it seems to get wider with increasing n (even though the transition region gets narrower). We would claim that all the ratios used in [11] are well within this hard region. So can it matter that much if we test exactly at the peak? If we are reasonably close, and if we are only interested in dramatic performance improvements, perhaps not. But this broadening of the hard region may not occur with other procedures, or other distributions, so we cannot count on it. Further, the left slope of the hard region becomes rapidly much steeper as n is increased. With $n = 75$, a shift in c of only four per cent can result in a factor of two difference in the number of steps taken, as is shown by the two vertical lines in Figure 8, and this effect seems to become more pronounced with increasing n. Moreover, since the transition region becomes narrower as n is increased, finding an accurate experimental estimate of a particular point on the transition curve to within a few per cent can be a very expensive task for large n.

The steep slope on the left of the hard region does seem rather far from the transition region to be much of a problem, but again for procedures other than DP this may not be the case. Likely candidates here are some of the local-search based methods. These have been used with considerable success for finding satisfying truth assignments for satisfiable formulas [26,16,25]. Random 3-SAT formulas have been among the primary test sets used in this work, and success has been reported with formulas near the hard region and having thousands of variables [25]. Although performance of local search is very good where just over half of the formulas are satisfiable, as of this writing we know of no rigorous study of how these procedures perform as c is increased through the transition region, and it seems likely that if there are formulas on which they do poorly, they may be in the right-hand part of the transition region.

A major difficulty in evaluating these procedures is that there is no sound and complete SAT testing program available capable of testing such hard formulas. Since the local search procedures are sound but incomplete, there is no way to know if such a procedure has solved all the satisfiable formulas given, and thus if reported average times are correct. Since failure is more likely as n increases, measured run times can be expected to under-estimate the actual increase in solution time with n. The narrowing of the transition region with increasing n further exacerbates the problem, because for large n an error of even one or two per cent in the choice of c could make a substantial difference in both the proportion of formulas which are satisfiable, and in the difficulty of these formulas.

8 Discussion

Random formulas have been used by many researchers to empirically evaluate the performance of SAT testing programs. The value of such studies depends upon careful selection of formula distribution and parameter values, and we presented a number of examples from the literature to illustrate this. The lesson here may be summarized as follows. When using random formulas, an extensive enough study of the distribution's parameter space must be carried out before an experiment is designed, if the results are to be meaningful. In the case of a previously unused distribution family, this may require extensive testing, and even for known distributions considerable care may be needed. Simple heuristics for picking parameter settings may not always be adequate, and a range of parameter values should be tested at each n of interest, to establish where peak difficulty is. This is especially true for local search methods, since their incompleteness means we have less knowledge about behavior near the transition region.

While it might be argued that it is too easy to point the finger at previous work in light of our current state of knowledge, there is an important lesson for future work, especially regarding "more structured" random distributions, such as those used in [21], or those used by workers in scheduling or constraint satisfaction. Similar pitfalls should be expected with any non-trivial parameterized distribution of random problems, and in fact will likely become even more troublesome. Suggestions for "realistically structured" problem distributions generally have *more* parameters than the currently popular "unstructured" families, so that effects of parameter manipulation on problem difficulty may be much more complex. A full understanding requires examination of the effects of every parameter, and also of every *combination* of parameters. Understanding all interactions of even five or six parameters is not a task to be taken lightly.

Watching for implicit assumptions in experimental designs is essential. We would like to stress two examples of particular importance with random SAT. The first is that size alone (or number of variables) is not a useful measure of hardness (*cf* Section 5). This is important because in evaluating algorithm performance we are most interested in what happens as problem size increases. Formulas with 5000 variables, 50,000 clauses and 500,000 literals [16] sound impressively large, but when they are from a distribution like the one we called random $\mathcal{E}$10-SAT, they can be solved very easily by a simple procedure like DP, with less than one backtrack on average. The second is that manipulating a single parameter while holding others constant does not necessarily give us a good indication of how that parameter affects instance difficulty (*cf* Section 6). This is important because many distributions of interest have multiple parameters with interaction effects which are far from intuitively obvious.

References

[1] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, March 1979.

[2] M. Chao and J. Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k satisfiability problem. *Information Sciences*, 51:289–314, 1990.

[3] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *Proc. IJCAI-91*, 1991.

[4] James M. Crawford and Andrew B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proc. AAAI-94*, 1994.

[5] J.M. Crawford and L.D. Auton. Experimental results on the cross-over point in satisfiability problems. In *Proc. AAAI-93*, pages 21–29, 1993.

[6] A. d'Anjou, M. Graña, F.J. Torrealdea, and M.C. Hernandez. Solving satisfiability via Boltzmann machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5), May 1993.

[7] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.

[8] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[9] John Franco. On the probabilistic performance of algorithms for the satisfiability problem. *Information Processing Letters*, 23:103–106, August 1986.

[10] A. Frieze and S. Suen. Analysis of three simple heuristics on a random instance of k-SAT. Manuscript, 1992.

[11] Giorgio Gallo and Giampaolo Urbani. Algorithms for testing the satisfiability of propositional formulae. *Journal of Logic Programming*, 7:45–61, 1989.

[12] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, New York, 1979.

[13] I.P. Gent and T. Walsh. The hardest random SAT problems. In *Proceedings, KI-94*, 1994.

[14] I.P. Gent and T. Walsh. The SAT phase transition. In *Proceedings, ECAI-94*, pages 103–109, 1994.

[15] A. Goldberg. On the complexity of the satisfiability problem. Courant Computer Science Report No. 16., 1979. New York University.

[16] Jun Gu. Efficient local search for very large-scale satisfiability problems. *SIGART Bulletin*, 3(1):8–12, 1992.

[17] J.N. Hooker. Resolution vs. cutting plane solution of inference problems: some computational experience. *Operations Research Letters*, 7(1):1–7, 1988.

[18] Anil Kamath, Rajeev Motwani, Krishna Palem, and Paul Spirakis. Tail bounds for occupancy and the satisfiability threshold conjecture. In *Proc. FOCS-94*, 1994.

[19] A.P. Kamath, N.K. Karmarkar, K.G. Ramakrishnan, and M.G.C. Resende. Computational experience with an interior point algorithm on the satisfiability problem. In *Integer Programming and Combinatorial Optimization*, pages 333–349. Mathematical Programming Society, May 1990.

[20] Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264, may 27 1994.

[21] Kurt Konolige. Easy to be hard: Difficult problems for greedy algorithms. In *Proc. KR-94*, pages 374–378, 1994.

[22] T. Larrabee and Y. Tsuji. Evidence for a satisfiability threshold for random 3CNF formulas. Technical Report UCSC-CRL-92-42, CRL, University of California, Santa Cruz, November 1992.

[23] David Mitchell, Bart Selman, and Hector .J. Levesque. Generating hard satisfiability problems. This volume.

[24] P. Purdom. A survey of average time analyses of satisfiability algorithms. *Journal of Information Processing*, 13(4), 1990.

[25] Bart Selman, Henry Kautz, and Bram Cohen. Noise strategies for improving local search. In *Proceedings, AAAI-94*, pages 337–343, 1994.

[26] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings, AAAI-92*, San Jose, CA, 1992.

[27] L.C. Wu and C. Y. Tang. Solving the satisfiability problem by using randomized approach. *Information Processing Letters*, 41:187–190, 1992.