The Satisfiability Constraint Gap

Ian P. Gent*
Department of AI
University of Edinburgh
80 South Bridge, Edinburgh
United Kingdom

Toby Walsh†
Mechanized Reasoning Group
IRST, Loc. Panté di Povo, Trento &
DIST, University of Genoa, Genoa
Italy

Abstract

We describe an experimental investigation of the satisfiability phase transition for several different classes of randomly generated problems. We show that the "conventional" picture of easy-hard-easy problem difficulty is inadequate. In particular, there is a region of very variable problem difficulty where problems are typically underconstrained and satisfiable. Within this region, problems can be orders of magnitude harder than problems in the middle of the satisfiability phase transition. These extraordinarily hard problems appear to be associated with a "constraint gap". That is, a region where search is a maximum as the amount of constraint propagation is a minimum. We show that the position and shape of this constraint gap change little with problem size. Unlike hard problems in the middle of the satisfiability phase transition, hard problems in the variable region are not critically constrained between satisfiability and unsatisfiability. Indeed, hard problems in the variable region often contain a small and unique minimal unsatisfiable subset or reduce at an early stage in search to a hard unsatisfiable subproblem with a small and unique minimal unsatisfiable subset. The difficulty in solving such problems is thus in identifying the minimal unsatisfiable subset from the many irrelevant clauses. The existence of a constraint gap greatly hinders our ability to find such minimal unsatisfiable subsets. However, it remains open whether these problems remain hard for more intelligent backtracking procedures. We conjecture that these results will generalise both to other SAT problem classes, and to the phase transitions of other NP-hard problems.

*Current address: Department of Computer Science, University of Strathclyde, Glasgow G1 1XH, United Kingdom. ipg@cs.strath.ac.uk

†toby@irst.it

1 Introduction

Many randomly generated NP-hard problems display a phase transition as some parameter is varied, and as the problems go from being almost always soluble to being almost always insoluble [3]. This phase transition is often associated with problems which are *typically* hard to solve. In this paper, we show that with several different classes of satisfiability problems including random 3-SAT, the phase transition is indeed associated with problems which are typically hard *but* there are also regions of very variable problem difficulty in which problems are usually easy but sometimes extraordinarily hard. We identify the cause of this behaviour and show that it does not disappear with better algorithms or heuristics. We predict that similar regions of very variable problem difficulty will be found with many other NP-hard problems besides satisfiability. The extraordinarily hard problems found in these regions may be of use in analysing and comparing the performance of algorithms for NP-hard problems.

2 Satisfiability

Propositional satisfiability (or SAT) is the problem of deciding if there is an assignment of truth values for the variables in a propositional formula that makes the formula true using the standard interpretation for logical connectives. We will consider SAT problems in conjunctive normal form (CNF); a formula, Σ in CNF is a conjunction of clauses, where a clause is a disjunction of literals, and a literal is a negated or un-negated variable. A standard procedure for determining satisfiability is due to Davis, Putnam, Logemann, and Loveland [6, 5]. We call this the "Davis-Putnam procedure".[1] See Figure 1.

An empty clause contains no literals, a unit clause contains just a single literal, and a tautologous clause contains both a literal and its negation. To simplify a set of clauses by the assignment of the literal l to $True$, we delete every clause that contains l and delete the negation of l whenever it occurs in the remaining clauses. Note that the Davis-Putnam procedure is non-deterministic since the literal used by the split rule is unspecified. As in previous studies (*eg.* [8, 14]), we will split upon the first literal in the first clause. We call this variant of the Davis-Putnam procedure "DP". With good heuristics for choosing the literal to split on, an efficient implementation of the Davis-Putnam procedure is still the best complete procedure for satisfiability [7].

[1] We follow recent nomenclature in this, for example [14, 8]. Davis and Putnam [6] introduced the unit and pure rules, while it was Davis, Logemann, and Loveland [5] who introduced the split rule and the use of backtracking. The latter authors modestly presented the difference as merely one of implementation.

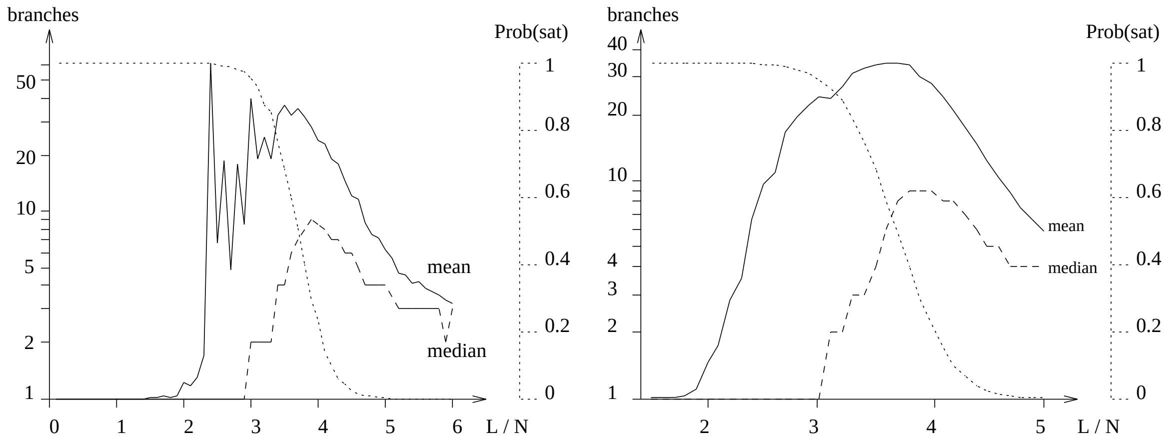
procedure DP(Σ)
    if Σ is empty **then return** satisfiable
    if Σ contains an empty clause **then return** unsatisfiable
    (*Tautology*) **if** Σ contains a tautologous clause c **then return** DP($\Sigma - c$)
    (*Unit propagation*) **if** Σ contains a unit clause $\{l\}$ **then**
        return DP(Σ simplified by assigning l to $True$)
    (*Pure literal deletion*) **if** Σ contains a literal l but not the negation of l **then**
        return DP(Σ simplified by assigning l to $True$)
    (*Split*) **if** DP(Σ simplified by assigning a literal l to $True$) is satisfiable
        then return satisfiable
        else return DP(Σ simplified by assigning the negation of l to $True$)

Figure 1: The Davis-Putnam Procedure

3 Constant Probability Model

The constant probability model of randomly generated problems has been the subject of considerable theoretical and experimental attention. In this model, given N variables and L clauses, each clause is generated so that it contains each of the 2N different literals with probability p. Our experiments use a variant of the constant probability model proposed in [11] and since used in other experimental studies [8, 9, 14]. In this problem class, if an empty or unit clause is generated, it is discarded and another clause generated in its place. This is because the inclusion of empty or unit clauses typically makes problems easier. We shall call this the "CP" model. In all our experiments, as in [8, 9], we choose p so that $2\mathrm{N}p = 3$ and the mean clause length remains approximately constant as N varies. In [14], it is shown that there is a phase transition between satisfiability and unsatisfiability for CP as the ratio of clauses to variables, L/N is varied. If $2\mathrm{N}p$ is kept constant, then this phase transition occurs at $\mathrm{L/N} \approx 2.80$ as $\mathrm{N} \rightsquigarrow \infty$ [9].

The satisfiability phase transition is of computational importance since there is an easy-hard-easy pattern in problem difficulty as we cross the phase transition with the hardest instances occurring in the phase transition [14]. When the ratio of clauses to variables is large, problems are usually over-constrained, and thus easily shown to be unsatisfiable. When the ratio is small, problems are usually under-constrained, and a satisfying assignment can be "guessed" quickly. The hard instances tend to occur in the phase transition where the problems are neither over-constrained nor under-constrained. In [8], we showed that whilst median problem difficulty has a simple easy-hard-easy pattern, there is also a region of very variable and sometimes exceptionally hard problem difficulty at a high percentage satisfiability. The worst case problems in this region can be orders of magnitude harder than those in the middle of the satisfiability phase transition. These extraordinary problems can easily dominate the mean problem difficulty.



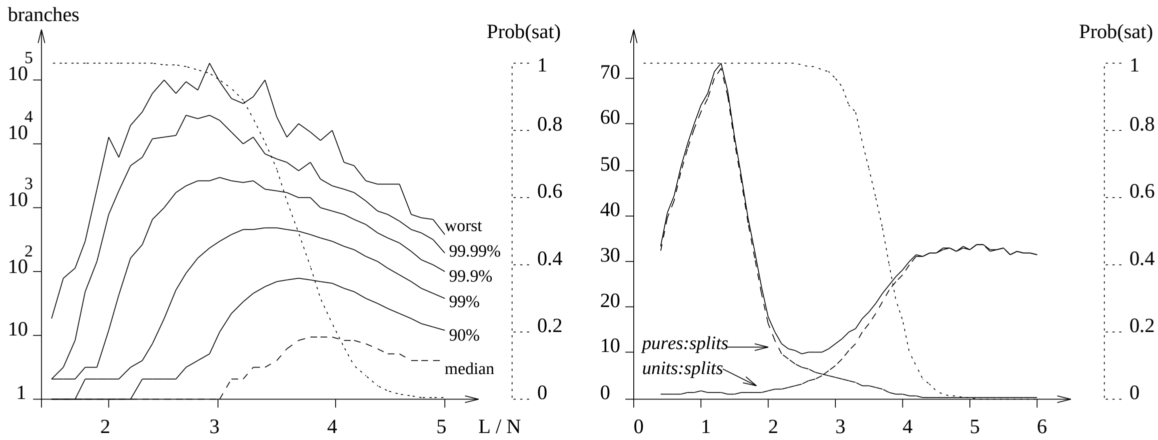
(a) 1000 problems (b) 100,000 problems

Figure 2: CP problems tested using DP, N= 100. Mean and median branches

Similar behaviour has been observed by Hogg and Williams for randomly generated 3-colourability problems [10].

Figure 2 (a) gives the mean and median number of branches used by DP for 1000 problems from the CP model at N = 100 with L/N from 0.1 to 6.0 in intervals of 0.1. The number of branches is the number of leaf nodes in the search tree. It provides a good indication of problem difficulty and run time. The dotted line indicates the observed probability that problems were satisfiable. There is a very considerable difference between mean and median performance. The worst case mean of 62.5 branches occurs at L/N = 2.4 in a mostly satisfiable region, whilst the worst case median of just 9 branches occurs at L/N = 3.9 in the middle of the satisfiability phase transition. Despite testing 1000 problems at each point, there is a large amount of noise in the mean, especially in the region of L/N from about 2 to 4, and despite the use of a logarithmic scale. In Figure 2 (b), we therefore tested 100,000 problems per point. Due to the large cost of testing this number of problems, we restricted our attention to the region L/N = 1.5 to 5.0. Although the noise is reduced considerably by taking 100,000 problems, there is still a large difference between mean and median performance. The greatest difference is in the region that was previously noisy and where median performance is only 1 or 2 branches. There is a secondary peak in mean problem difficulty of 24.3 branches at L/N = 3.0. The worst case mean of 35.0 branches occurs at L/N = 3.6, close to the worst case median of 9 branches in the middle of the satisfiability phase transition. The variable behaviour in the satisfiable region increases rapidly with N and eventually dominates the mean [8]. The worst mean performance at large N therefore occurs in the satisfiable region and not in the middle of the phase transition.

To explore this phenomenon further, in Figure 3 (a) we give a breakdown



(a) percentile branches (b) constraint propagations/splits

Figure 3: CP problems tested using DP, N= 100.

in percentiles for the number of branches used from 50% (median) up to 100% (worst case) in the experiment described above in which 100,000 problems were tested at each point. The 99.9% contour, for example, gives the difficulty of the problem which took more branches than all but 0.1% of problems: in this case, this would be the hundredth hardest problem at each point. Interestingly, the worst case contour is very noisy despite the very large number of problems tested and the logarithmic scale it is plotted on. Furthermore, the different contours are widely separated, especially at smaller values of L/N, showing how, for example, the tenth worst problem can be almost an order of magnitude harder than the hundredth worst problem. As we examine contours closer to the worst case, they peak at smaller values of L/N. In particular, the worst case was 185,902 branches at L/N = 2.9, while at L/N = 3.9, the point of worst median performance, the worst case was just 10,959 branches, more than an order of magnitude smaller.

In [8], we show that similar behaviour for CP is observed with better splitting heuristics. However, variable and difficult behaviour in the high percentage satisfiable region is not apparent till larger N.

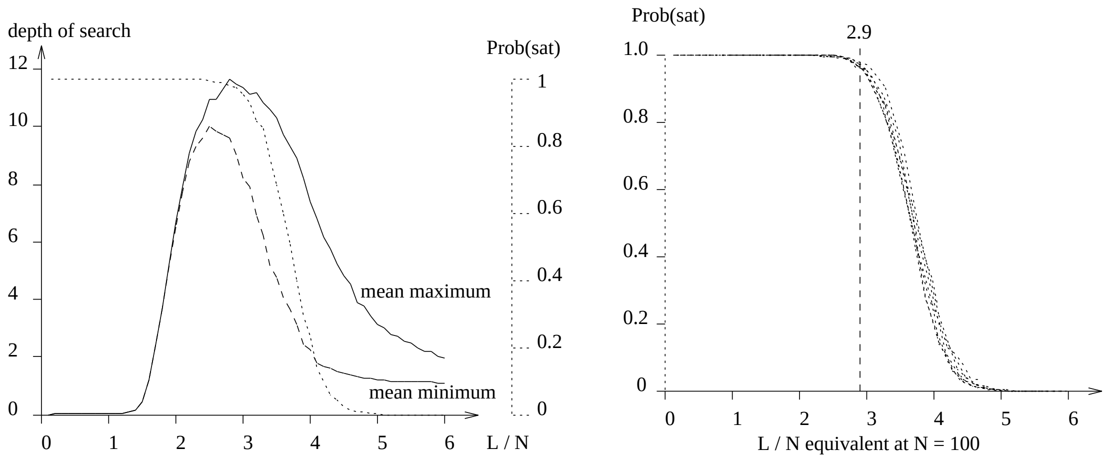
4 Constraint Gap

In the Davis-Putnam procedure, the split rule is the only rule which gives rise to exponential behaviour. The other rules simplify the problem and do not branch the search. In particular, the unit and pure rules take advantage of constraints to commit in polynomial time to particular truth assignments. Since the extremely bad worst case performance in the mostly satisfiable region is presumably due to exponential behaviour, we therefore conjecture that both the unit and pure rules will be of less importance than the split rule in this region.

In Figure 3 (b) we plot the mean ratio of pure literal deletions to splits, of unit propagations to splits and of the sum of pure literal deletions and unit propagations to splits for CP at N=100. The uppermost (solid) line shows the ratio of all constraint propagations to splits. Underneath this, plotted to the same scale, are the ratio of number of applications of the pure rule to splits (peaking to the left), and the ratio of number of applications of the unit rule to splits (peaking to the right). Since the split rule is merely formalised guessing, the ratio of all propagations to splits indicates the number of variable assignments that can be deduced for each guess during search. To avoid division by zero, we exclude the trivial problems which tend to occur at small L/N that are solved with no splits. Such problems can be solved in polynomial time using a simple preprocessing step which exhaustively applies the unit and pure rules. As a guide, the dotted line repeats the probability of satisfiability from Figure 2 (a). We show the effect of both unit and pure propagations individually, and of them both together. The number of pures dominates behaviour at small values of L/N, while unit propagations start to dominate at large values. The ratio of all propagations to splits shows a large peak of 73.5 at L/N = 1.3. In this region, almost all problems are trivial, being solved almost exclusively by pure literal deletion. However, with increasing L/N, the number of pure literal deletions drops very rapidly, and applications of unit propagation take over. A local maximum of 33.8 propagations per split is reached at L/N = 5.2, though in this region performance is comparatively noisy.

The most interesting region in Figure 3 (b) is the region where neither pure literal deletions nor unit propagations dominate behaviour, since in this region the total number of propagations shows a pronounced minimum. The minimum in the mean ratio of the sum of units and pures to splits is 9.8 and occurs at L/N = 2.5, close to the position of the hardest worst case.

These graphs confirm that the unit and pure rules are not effective in the mostly satisfiable region. There appears to be a "constraint gap"; that is, there seems to be a region where the unit and pure rules are often unable to identify constraints on the truth assignments and we have to use the split rule extensively. This would suggest that the depth of search (*i.e.* the depth of nesting of split rule applications) would also peak in this region. In Figure 4 (a), we plot the mean minimum, and mean maximum depth of search. The peak of the minimum depth is 10.0 at L/N = 2.5 while the peak of maximum depth is 11.6 at L/N = 2.8. This coincides closely with the minimum in the ratio of the sum of units and pures to splits, and with the position of the hardest worst case. For unsatisfiable problems, a peak in minimum search depth corresponds to an exponentially larger peak in problem difficulty, as all branches must be searched to at least the minimum depth of the tree. We confirmed this by plotting the logarithm of problem difficulty for unsatisfiable problems alone. This was approximately proportional to mean minimum search depth.



(a) mean min and max depth, N=100 (b) scaling of Prob(sat), N= 25 to 250

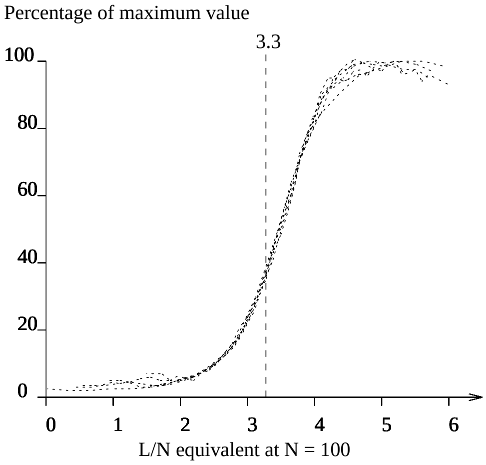
Figure 4: CP problems tested using DP.

5 Scaling of the Constraint Gap

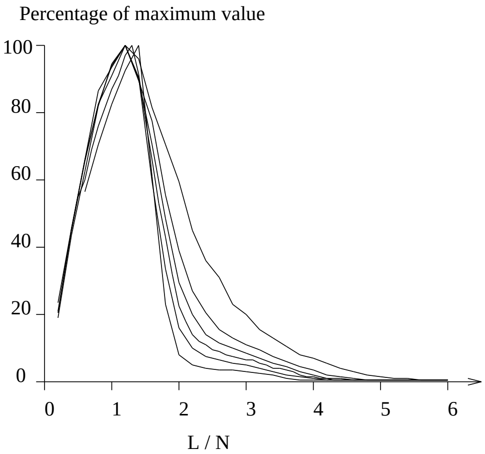
The importance of the constraint gap depends in part upon the relationship of the constraint gap to the phase transition from satisfiability to unsatisfiability. For instance, if the constraint gap occurs at or near to the satisfiability phase transition, then it is likely to prove much more costly than if the constraint gap occurs well away in a region of otherwise very easy problems. To help determine the relationship between the constraint gap and the satisfiability phase transition, we draw upon an analogy with phase transitions in physical systems.

One of the most unusual and theoretically interesting phase transitions occurs in spin glasses. Each of the N atoms in a spin glass has a magnetic spin which can have only one of two values, "up" or "down" (1 or -1). The system therefore has 2^N possible configurations. Macroscopic properties of a configuration (*eg.* the energy, entropy) depend only on interactions between the spins of nearest neighbours. Due to the differences in separation of the atoms, some of these interactions are ferromagnetic (promoting alignment of spins) whilst others are anti-ferromagnetic (promoting opposite spins). The net effect is a random force. An analogy can be made between such spin glasses and randomly generated SAT problems. Each of the N variables in a truth assignment has one of two values, "True" or "False". The system therefore has 2^N possible configurations. Macroscopic property like satisfiability depend only on the interaction between variables neighbouring each other in a clause. Due to the random polarities of these variables, the net effect on a variable is a random "preference" towards True or False.

Kirkpatrick *et al.* [13] have used this analogy to suggest a fascinating scaling result for random k-SAT (this randomly generated problem class contains clauses of fixed length k; it is described in §6). They propose that there is a fundamental



(a) scaling of units/splits



(b) scaling of pures/splits

Figure 5: CP problems tested using DP, N= 25 to 300

function f, and values α and v, such that

$$Prob(sat) = f((\mathrm{L/N} - \alpha)\mathrm{N}^{1/v}) \tag{1}$$

Here we show that this relation also holds for problem classes like CP which contain clauses of mixed lengths. In Figure 4 (b) we plot the probability of satisfiability for CP along the y axis and $(\mathrm{L/N}-\alpha)\mathrm{N}^{1/v}$ along the x axis. For convenience, we have also multiplied the x-ordinate by $100^{-1/v}$ and added α so that the values on the x-axis give the equivalent value of L/N at N= 100. The dashed line gives the point $\mathrm{L/N} = \alpha$. We set α to 2.9 as this was the experimentally observed position of the satisfiability phase transition. A value for v was found by trial and error. Using $v = 2.5$, we found a very good fit with the curves for $\mathrm{N} = 25, 50, 75, 100, 150, 200$, and 250. These plots are never more than 0.3 apart in terms of L/N as measured at $\mathrm{N} = 100$. This is a slightly less good fit than has been observed for random 3-SAT [13] or for random mixed SAT [9] (this problem class contains a fixed distribution of clause lengths; it is defined in §7). This may be because, in the CP model, the distribution of clause lengths changes slightly with problem size, or because the expected number of tautologies, which are allowed in the CP model but not in random 3-SAT or random mixed SAT, changes with problem size.

The normalized ratios of units to splits and pures to splits are, like the probability of satisfiability, macroscopic properties of the satisfiability system which vary with N and L between 0 and 100%. We therefore investigated how these features of search scale as the problem size changes. Very surprisingly, the normalized ratio of units to splits seems to scale in a very simple fashion similar to that of probability of satisfiability. That is, there exists a function g, and constants α_u,

v_u such that,

$$\frac{units/splits}{\max(units/splits)} \quad = \quad g((\mathrm{L/N} - \alpha_u)\mathrm{N}^{1/v_u}) \qquad (2)$$

The normalizing maximal values are for a fixed N and varying L. In Figure 5 (a) we plot the normalized ratios of units to splits for CP problems for N = 25, 50, 75, 100, 150, and 300. The x axes are scaled in the same way as Figure 4 (b). Using $\alpha_u = 3.3$, $v_u = 2.5$, we found a very good fit to Equation (2), as can be seen clearly from the graph, particularly in the region of L/N from approximately 2 to 4. Behaviour at very small or large values of L/N does not seem to fit the model as closely. It is particularly important to note that the value of α_u used here, namely 3.3, is larger than values of L/N where we earlier observed very bad worst case behaviour, and the constraint gap. The maximum ratio of units to splits seems to scale as approximately $N^{0.6}$. We would expect less than linear scaling of this ratio, as otherwise the number of splits required would not increase with N. Sublinear scaling of this ratio suggests exponential growth in the number of branches required to be searched.

The scaling of the ratio of pures to splits is less simple than that of units. In Figure 5 (b) we show the ratios of the number of pure literal deletions to splits for the same values of N as before, normalised by the relevant maximum value on the y-axis but unscaled on the x-axis. It can be seen that there is a very large peak at very low L/N, in a region where most problems are proved to be satisfiable simply by a large number of applications of the pure rule. This peak moves slightly to the right with increasing N. By contrast, the decline from this peak value is very fast, and takes place with dramatically increasing speed as N increases. Unlike unit propagations, the peak of ratio of pures to splits seems to approach the number of variables as N increases. However, this peak occurs at extremely small values of L/N, suggesting that such problems are essentially trivial. It seems likely that in the constraint gap, and beyond, scaling of the number of pures to splits is indeed sublinear. For example, at L/N = 2, the ratio of pures to splits increases only from 16 at N = 100 to 18 at N = 300, hardly increasing at all while the problem size triples. It is harder to come to any definite conclusion as it was for unit propagations. However, it should be noted that the peak in pure applications seems to occur at significantly lower values of L/N than where we observed bad behaviour and the constraint gap. As the behaviour at low L/N is dominated by easy or trivial problems, it is possible that if these could be eliminated in a suitable way, we would observe a similar scaling to that seen with the probability of satisfiability and of unit propagations to splits.

These results suggest that the constraint gap will get more pronounced with increasing N. In particular, the decline on the number of unit propagations as N increases and L/N decreases below α_u gets sharper. The behaviour of the pure rule is less clear-cut, but seems to show broadly similar behaviour. In [8], we showed worst case behaviour is due to unsatisfiable problems and to unsatisfiable subproblems of satisfiable problems [8]. As pure literal deletions only directly help to solve satisfiable problems (though at least simplifying unsatisfiable problems), we would not expect pure literal deletion to help suppress bad worst case behaviour in the constraint gap, even if the utility of the pure literal rule did not decay.

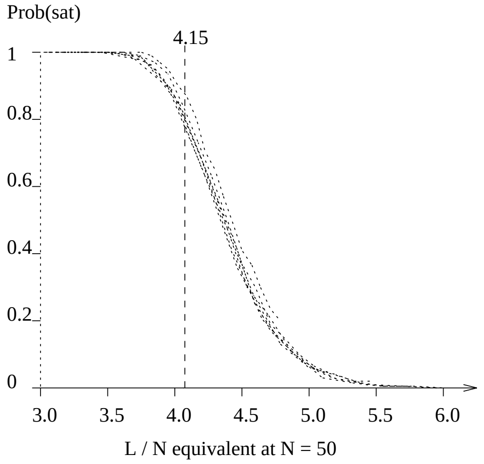
These results show that the constraint gap for CP, like the phase transition between satisfiability and unsatisfiability, occurs at a fixed value of L/N; that is, at a fixed ratio of constraints to variables. The constraint gap seems to be open between approximately L/N = 2 and 3.3. The position of the constraint gap is thus fixed relative to the position of the satisfiability phase transition. From the shape of g , the scaling of the maximal values, and the value v_u it also appears that the constraint gap will become more pronounced as N increases.

6 Random k-SAT

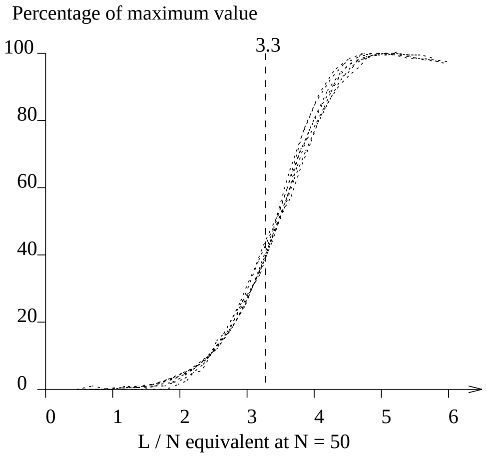
Following [14], many studies of the satisfiability phase transition have concentrated on the random k-SAT problem class. A problem in random k-SAT consists of L clauses, each of which has k literals chosen uniformly from the 2N possible variables, each literal being positive or negative with probability $\frac{1}{2}$. Unlike CP, all clauses in the random k-SAT model are of the same length. In [14], it was shown that there is a satisfiability phase transition for random 3-SAT at L/N ≈ 4.3, and that median behaviour displays a simple easy-hard-easy pattern through this phase transition. In [8] we showed that hard random 3-SAT problems can also occur in the mostly satisfiable region, and Crawford and Auton also anecdotally report this for very large problems [4]. Indeed, problems in the mostly satisfiable region can be several orders of magnitude harder than the hardest problems from the middle of the satisfiability phase transition (the point of worst median performance). These extraordinarily hard problem appear to be rarer in random 3-SAT than in CP. Although we found such hard problems in a sample of 1000 for CP, it required 100,000 for random 3-SAT. As with CP, these hard random 3-SAT problems occur in the region of a constraint gap, a minimum in the ratio of constraint propagations to splits. Again, as with CP, the constraint gap and the probability of satisfiability scale as in Equations (1), (2).

In Figure 6 (a) and (b) we plot the probability of satisfiability and the normalized ratios of units to splits for random 3-SAT problems for N = 10 to 70 in steps of 10 Using $\alpha = 4.15$, $v = 1.5$ (following [13]), $\alpha_u = 3.3$ and $v_u = 2.85$, we found a very good fit to Equations (1) and (2). The peak ratio of units to splits seems to vary approximately as $N^{0.65}$. Again, mean minimum depth of search peaks at a small value of L/N. For example, at N = 50, the maximum value was 11.1 at L/N = 2.6 As with CP, we did not obtain a similar scaling result for the ratio of pures to splits. However, the regions of many pure literal deletions again decays at a smaller value of L/N than the region of many unit propagations.

To conclude, hard problems can also occur with random 3-SAT in the mostly satisfiable region. These hard problems again appear to be associated with a constraint gap. This constraint gap occurs at a fixed value of L/N, and becomes



(a) scaling of Prob(sat)



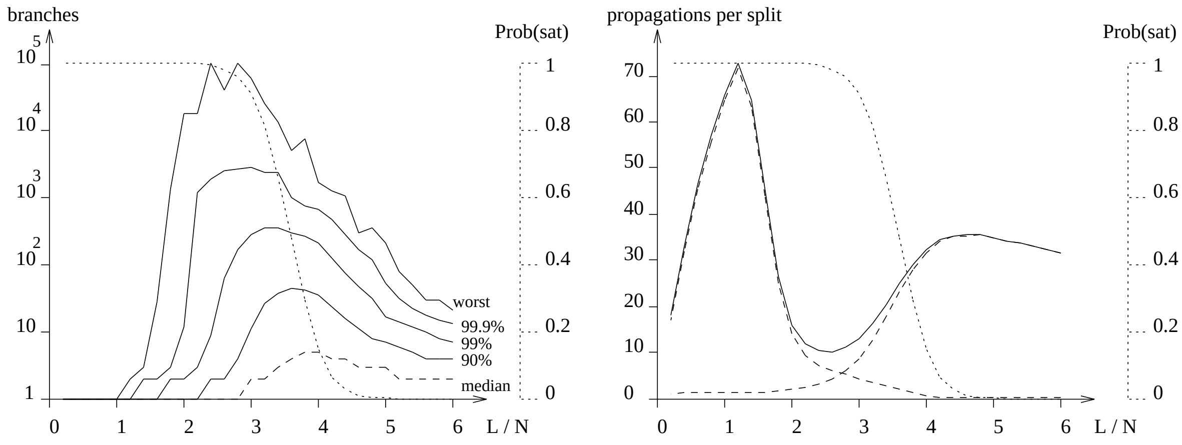
(b) scaling of units/splits

Figure 6: Random 3-SAT problems tested using DP, N= 10 to 70

more pronounced as N increase. The position of the constraint gap is fixed relative to the position of the satisfiability phase transition. While the satisfiability phase transition for CP seemed to be within the constraint gap (see §5), this does not seem to be the case for 3-SAT. The probability phase transition seems to occur at approximately 4.2, while the constraint gap seems to end at approximately 3.3. This difference may account for some aspects of the differences in behaviour between CP and 3-SAT.

7 Random Mixed SAT

In [9], we introduced a generalisation of the random k-SAT model, called "random mixed SAT". In the satisfiability phase transition, problems from random mixed SAT can be much harder than comparably sized random k-SAT problems. In the random mixed SAT model, a set of clauses is generated with respect to a probability distribution ϕ on the integers. Each clause is generated as in random k-SAT except that k, the length of the clause, is chosen randomly according to ϕ. For example, if $\phi(2) = \phi(3) = \frac{1}{2}$, then clauses of length 2 and 3 appear with probability $\frac{1}{2}$, whilst if $\phi(2) = \frac{1}{3}$ and $\phi(4) = \frac{2}{3}$, clauses of length 2 appear with probability $\frac{1}{3}$ and of length 4 with probability $\frac{2}{3}$. In this paper, we will call these problem classes "2,3-SAT" and "2,4,4-SAT" respectively. The frequency of occurrence of an integer in the name reflects the frequency of occurrence of clauses of this length in the problem. Random k-SAT is a special case of random mixed SAT in which $\phi(k) = 1$, and $\phi(j) = 0$ for $j \neq k$. For a given ϕ, we define



(a) percentile branches (b) constraint propagations/splits

Figure 7: Random 2,4,4-SAT problems tested using DP, N= 100.

the *density* of ϕ, d_ϕ as:

$$d_\phi =_{def} \sum_{k=1}^{\infty} \phi(k)(1 - (\frac{1}{2})^k)$$

The density gives the mean fraction of all truth assignments that are consistent with any given clause generated by ϕ. The expected number of models of a random mixed SAT problem with N variables and L clauses is then $2^{\mathrm{N}}(d_\phi)^{\mathrm{L}}$.

The random mixed SAT model may generate problems more similar to real-world problems than random k-SAT. For example, many structured problems encode into SAT problems with mixed clause lengths (*eg.* scheduling problems can be encoded into SAT using large numbers of binary clauses). The random mixed SAT model also produces problems which can, as we have said, be much harder than random k-SAT. In [9], we showed that the satisfiability phase transition for random mixed SAT for a given ϕ seems to occur, as with random k-SAT, at a fixed ratio of L/N. For random 2,4,4-SAT, the phase transition occurs at L/N ≈ 2.74, and we observed a similar kind of scaling as seen in Figure 4 (b), with constants $\alpha = 2.74$, $v = 3.5$.

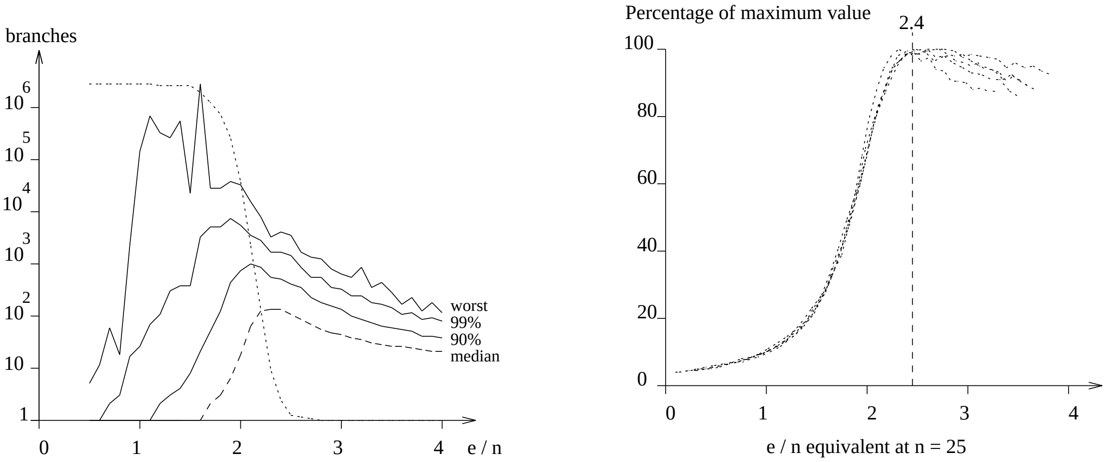
Although random 2,4,4-SAT has the same density as random 3-SAT, problem difficulty through the satisfiability phase transition of random 2,4,4-SAT is more similar to that of CP than that of random 3-SAT. Figure 7 (a) shows the percentile branches used by DP for random 2,4,4-SAT problems at N=100, with 10,000 problems tested at each value of L/N from 0.2 to 6.0 in steps of 0.2. Median problem difficulty shows a simple easy-hard-easy pattern, whilst the hardest problems are again found in a region of high percentage satisfiability. The worst case was 104,885 branches at L/N = 2.8, while at L/N = 3.8 the point of worst median performance, the worst case was just 7,534 branches, two orders of magnitude smaller. As with CP, these extraordinarily hard problems appear to be associated with a constraint gap. In Figure 7 (b) we plot the mean ratio of constraint propagations to splits for random 2,4,4-SAT at N=100. The minimum in the mean ratio of the sum of units and pures to splits is 10.1 and occurs at L/N = 2.6, close to the position of the hardest worst case. As with CP, the depth of search also peaks in this region. The peak of the minimum depth is 9.8 at L/N = 2.4 while the peak of maximum depth is 11.3 at L/N = 2.8. It seems likely that we will observe a similar scaling of the constraint gap for 2,4,4-SAT as we observed above for CP.

8 k-Colourability

Another way of randomly generating SAT problems is to map random problems from some other NP-hard problem into SAT. For example, the k-colourability (kCOL) of random graphs can be easily mapped into SAT. Given a graph, G the k-colourability problem is to assign one of k labels to each vertex of G so that adjacent vertices carry different labels. For a graph with n vertices and e edges, our encoding of kCOL into SAT uses $n.k$ variables. We generate random graphs to encode into SAT by choosing e edges from the $n.(n-1)/2$ possible edges uniformly at random. We use $\chi(n, e)$ to denote graphs drawn from this class.

In Figure 8 (a) we plot the breakdown in percentiles for the number of branches used by DP for encodings of 3-colourability for 1000 problems taken from $\chi(n, e)$ with $n = 40$ and $e/n = 0.5$ to 4 in steps of 0.1. The worst case was 2,905,011 branches at $e/n = 1.6$, while at $e/n = 2.4$, the point of worst median performance, the worst case was just 4,139 branches, 3 orders of magnitude smaller. As with the other random problem classes, median problem difficulty shows a simple easy-hard-easy pattern through the k-colourability phase transition. Very similar behaviour for k-colourability was observed by Hogg and Williams using two special purpose colouring algorithms [10].

The constraint gap seems to be an important feature in 3COL as well as in other problem classes of satisfiability problems. However, pure literal deletions seem less important than previously. Even at $e/n = 0.4$ the number of pure propagations per split is only 3.8 at $n = 40$, compared to a peak of 34.9 unit propagations per split at $e/n = 2.65$. The number of unit propagations per split seems to scale similarly with n to the earlier cases of CP and 3-SAT, suggesting that once again the constraint gap is an important way of understanding search. In Figure 8 (b) we plot normalized values of the ratio of unit propagations to splits for $n = 10$, 20, 25, 30, 40, scaled by Equation 1 with the values $\alpha_u = 2.4$, $v = 3$. The depth of search also peaks in the region of extraordinarily hard problems.



(a) percentile branches, n = 40 (b) constraint propagations/splits, n = 10 to 40

Figure 8: 3COL problems from $\chi(n, e)$ tested with DP

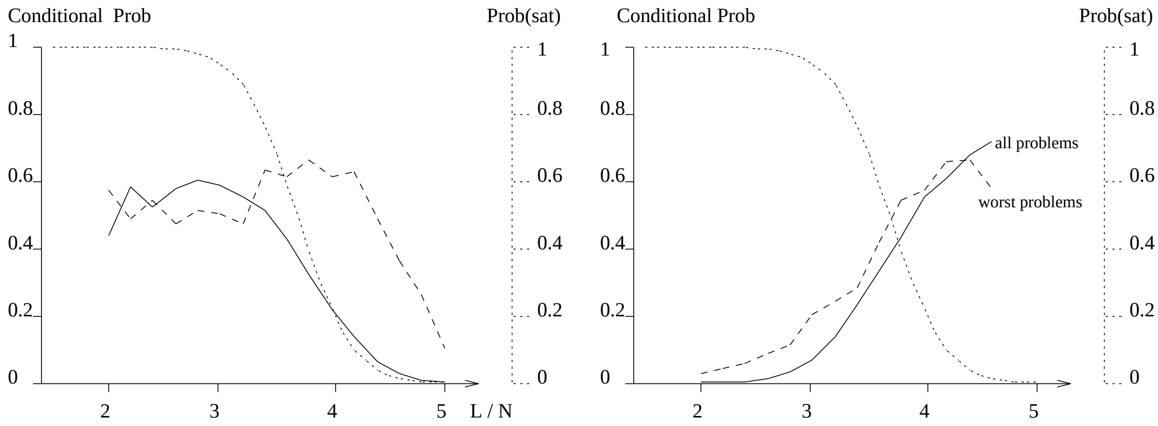
9 Critically Constrained Problems

Crawford and Auton [4] have suggested that hard problems in the satisfiability phase transition are *critically constrained.* That is, they are neither so under-constrained that we can guess one of the many models easily, nor so over-constrained that we can determine their unsatisfiability with little search. Such problems are on the knife edge between satisfiability and unsatisfiability. To test this hypothesis, we ran a series of experiments in which we added and deleted constraints from the hardest CP problems in the experiments described in §3.

As the most important aspects of behaviour seem to arise on unsatisfiable problems [8], we first examined behaviour on unsatisfiable problems only. We took the 100 hardest unsatisfiable problems from 100,000 CP problems at each point and measured the probability that they become satisfiable if we delete 20 clauses at random. This is shown in Figure 9 (a) by the dashed line. Noise in this graph is accounted for by the selection of a small number of problems at each point. For comparison, the solid line gives the probability that any unsatisfiable problem at each point becomes satisfiable if we delete 20 clauses at random, and the dotted line gives the overall probability of satisfiability. This graph has two interesting features. First, it confirms that in the satisfiability phase transition, the hardest unsatisfiable problems are significantly more critically constrained than all problems. This is consistent with the suggestion of [4] that hard problems in the phase transition are critically constrained. Equally importantly, however, in the region of mostly satisfiable problems and of the constraint gap, the hardest unsatisfiable problems are significantly *less* constrained than typical unsatisfiable problems. Yet these problems are precisely the ones that require most search to solve of all problems at all values of L/N. We defer detailed discussion of these problems to the next section, where we show that the large amount of search arises precisely because these unsatisfiable problems are not critically constrained, and hence contain little information to help towards proving unsatisfiability.

Satisfiable problems are hard for rather different reasons to unsatisfiable ones. In particular, much search can only be needed if reductions early in search lead to unsatisfiable subproblems. This suggests that the hardest satisfiable problems should be more critically constrained at all values of L/N. To test this, we took the 100 hardest satisfiable problems from 100,000 at each point, and measured the probability that they become unsatisfiable if we add 20 randomly generated CP clauses. This is shown in Figure 9 (b) by the dashed line. For comparison, the solid line gives the probability that any satisfiable problem at this point (not just one of the hardest 100) becomes unsatisfiable if we add 20 randomly generated CP clauses. For reference, the dotted line gives the overall probability of satisfiability. It can be seen that except at large L/N (where there are very few satisfiable problems and so unusual effects may be due to noise), the worst satisfiable problems are easier to make unsatisfiable than typical problems. This suggests that the hardest satisfiable problems are more critically constrained than typical satisfiable problems. The most significant feature of the data may be that the largest discrepancy between the hardest satisfiable and all satisfiable problems is between L/N = 2 and 3. This is consistent with the observation in [8] that very hard satisfiable problems in this region arise because an incorrect choice at a branching point in search leads to a hard unsatisfiable subproblem. While unsatisfiable problems in this region are hard because they are not critically constrained, satisfiable problems can only give rise to hard unsatisfiable subproblems at an early stage in search if they are critically constrained. Even so, these problems are significantly less critically constrained than satisfiable problems in the middle of the phase transition. This helps to explain why the hardest satisfiable problems can often be solved very quickly if randomly different choices are made during search, as we showed in [8].

These graphs show that the most critically constrained problems occur, as suggested by Crawford and Auton, in the middle of the satisfiability phase transition. We can neither add nor delete many constraints from problems in this region without changing their satisfiability. By comparison, hard unsatisfiable problems in the mostly satisfiable region are not critically constrained since we can delete many constraints without making them satisfiable. The hardest satisfiable problems in this region are more critically constrained than typical problems, and thus they can reduce at early point in search to a hard unsatisfiable problem. In the next section we will argue that, whilst problems in the middle of the satisfiability phase transition are difficult because they are so *critically* constrained, problems in the mostly satisfiable region are difficult because they are so *uncritically* constrained.



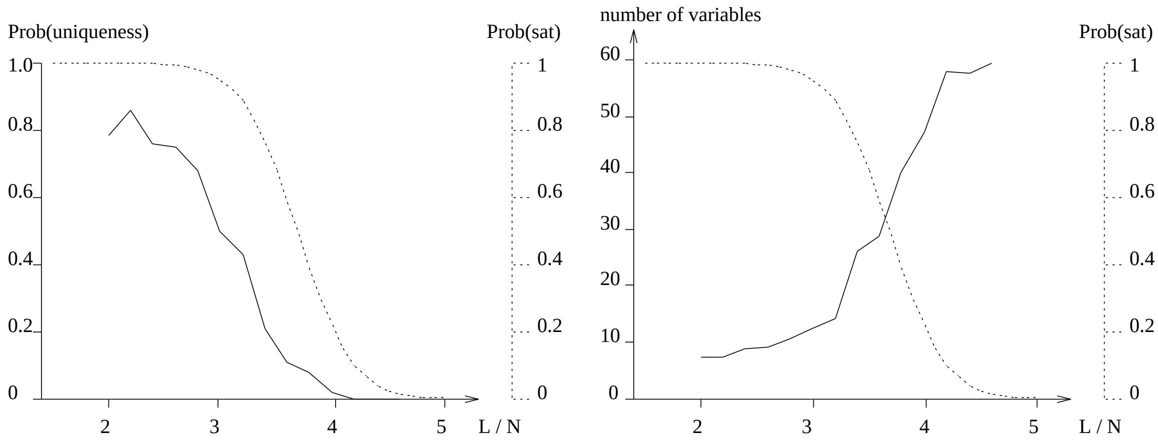
(a) deleting 20 clauses (b) adding 20 clauses

Figure 9: CP, 100 hardest sat/unsat problems for DP from 100,000, N=100

10 Minimal Unsatisfiable Subsets

In [8], we showed that hard problems in the mostly satisfiable region are either hard unsatisfiable problems or are satisfiable problems that give rise to hard unsatisfiable subproblems following an incorrect split at the start of search. To explain the difficulty of problems in the mostly satisfiable region, we will therefore focus on unsatisfiable problems. Since hard unsatisfiable problems in the mostly satisfiable region are not critically constrained, we can delete many clauses from them without making them satisfiable. This suggests that hard unsatisfiable problems from the mostly satisfiable region contain a large number of irrelevant clauses. To test this hypothesis, we computed minimal unsatisfiable subsets of the hardest unsatisfiable problems. S is a *minimal unsatisfiable subset* of T iff $S \subseteq T$, S is unsatisfiable and there does not exist R with $R \subset S$ and R unsatisfiable. To compute a minimal unsatisfiable subset of T, we deleted each clause of T in turn, adding it back only if deleting it makes the set of clauses satisfiable. Unfortunately, it is too computationally expensive to compute all minimal unsatisfiable subsets. We did, however, determine if the computed minimal unsatisfiable subset is unique. S is a *unique* minimal unsatisfiable subset of T iff S is a minimal unsatisfiable subset of T and for all $\varphi \in S$, $T - \{\varphi\}$ is satisfiable.

In Figure 10 (a), at each value of L/N, we took the 100 hardest unsatisfiable CP problems for DP from 100,000 and measured the probability that they have a unique minimal unsatisfiable subset. For reference, the dotted line gives the probability of satisfiability for all problems at this point. In Figure 10 (b) we plot the average number of variables in the computed minimal unsatisfiable subset. When there is not a unique minimal unsatisfiable subset, our method of computation will tend to under-estimate the average number of variables of *all* the minimal unsatisfiable subsets since there are more ways of computing a small minimal un-



(a) probability of uniqueness (b) no. of variables

Figure 10: Minimal unsatisfiable subsets for 100 hardest unsat CP problems. N = 100

satisfiable subset than a large one. Again, for reference, the dotted line gives the probability of satisfiability for all problems at this point. A graph of the average number of clauses in the minimal unsatisfiable subsets looks very similar in shape to that of the number of variables in the minimal unsatisfiable subsets. The ratio of clauses to variables in the minimal unsatisfiable subsets increases slowly from 1.15 at L/N = 2.2 to 1.29 at L/N = 4.6. The observed minimal unsatisfiable subsets are typically dominated by binary clauses, especially the very small minimal unsatisfiable subsets observed at small values of L/N. As the size of minimal unsatisfiable subsets increase, the number of non-binary clauses increases accordingly, but even so the average size of clauses we observed increased only from 2.05 at L/N = 2.0 to 2.61 at L/N = 4.6.

These graphs show that hard unsatisfiable problems in the mostly satisfiable region tend to have very small and unique minimal unsatisfiable subsets which mention few variables. The difficulty in solving such problems is thus one of *irrelevancy*. DP will waste most of its time splitting on irrelevant variables. Because of the constraint gap, few constraint propagations follow each split. We are thus unlikely to simplify on one of the variables in a minimal unsatisfiable subset. Only after DP has backtracked through a large number of irrelevant variable assignments, will we discover a minimal unsatisfiable subset. As the minimal unsatisfiable subsets of hard problems in the mostly satisfiable region are small and mention few variables, they have very short proofs. This suggests that most of the computation performed by DP was (logically speaking) unnecessary. It is, however, very hard to find a short proof since our heuristics and constraint propagation must identify the minimal unsatisfiable subset from the large number of irrelevant (and satisfiable) clauses. We are much more likely to find one of the many long proofs containing large amounts of unnecessary computation than one of the few short proofs containing little unnecessary computation. This explains why in [8] we found that hard unsatisfiable problems in the mostly satisfiable region can sometimes have short proofs but that such proofs are hard to find.

Note that a search strategy optimized for short proofs (for example, breadth first or iterative deepening search) will not ultimately help. If the length of the shortest proof of the minimal unsatisfiable subset is l splits, then the search space for a complete procedure like breadth first search is $O(\mathrm{N}^l)$. This is polynomial if and only if l is bounded. Unfortunately, even though the minimal unsatisfiable subset is small in comparison to L and N, it could increase in size with N and still be hard to identify. The length of the shortest proof could therefore easily be unbounded. For example, the size of the minimal unsatisfiable subset, and l might increase as $O(\sqrt{\mathrm{N}})$ or $O(\log(\mathrm{N}))$.

These results help to explain why hard problems are much rarer in the mostly satisfiable region of random 3-SAT than the mostly satisfiable regions of CP and random 2,4,4-SAT. CP and random 2,4,4-SAT problems contain large numbers of binary clauses. It is thus not too difficult to hide a small and unique minimal unsatisfiable subset within a large satisfiable CP or random 2,4,4-SAT problem. By comparison, all clauses in a random 3-SAT problem must contain 3 literals. The minimal unsatisfiable subsets in random 3-SAT are therefore typically larger and mention more variables. It is thus more difficult to hide a minimal unsatisfiable subset within a satisfiable random 3-SAT problem.

In the mostly satisfiable region, the problem of solving a rare unsatisfiable problem is that of identifying a single minimal unsatisfiable subset. Once found it is usually very easy to solve. However, there may be few clues available to its identification. On the other hand, in the middle of the phase transition, the identification of a minimal unsatisfiable subset is not so important, because the minimal unsatisfiable subsets are comparatively large, and indeed may not in themselves be significantly easier than the problem as a whole. Furthermore, choice points in search are less likely to be wasted, because most splits contribute towards a proof of unsatisfiability, as is seen by the large number of variables in the minimal unsatisfiable subsets. In the mostly satisfiable region, bad choices can double search time, as they may make no contribution to deriving the unsatisfiability of the unique minimal unsatisfiable subset.

To conclude, hard unsatisfiable problems in the mostly satisfiable region often have very small and unique minimal unsatisfiable subsets. These minimal unsatisfiable subsets are hidden within much larger random satisfiable problems. It is thus very difficult to find such minimal unsatisfiable subsets. An analogy can be made with cryptography where it is very difficult to identify a short message if it is hidden in a long stream of white noise. By comparison, hard unsatisfiable problems in the middle of the satisfiability phase transition typically have much larger minimal unsatisfiable subsets which are not unique.

11 Binary Constraints

The minimal unsatisfiable subsets often contain many binary clauses, or reduce to binary clauses after just one variable assignment. Since there exists a linear time algorithm for the satisfiability of binary clauses [1], problems containing such minimal unsatisfiable subsets can be solved in polynomial time. We have therefore augmented DP with the following rule:

(Binary) if the binary clauses of (Σ simplified with the literal l assigned to $True$) are unsatisfiable then assign the negation of l to $True$.

This rule has a non-deterministic choice of literal; this may affect the number of pure, unit or binary rules applied but *not* the number of splits or branches. We tested DP augmented with the binary rule on 100 variable CP problems. The worst mean performance was only 2.5 branches at L/N = 3.4, more than an order of magnitude less than the performance of DP alone. (Note that it cannot be deduced from this that *run-time* is reduced, as there is a large overhead in applying the rule.)

The binary rule is able to solve many of the (previously hard) unsatisfiable problem in the variable region, sometimes without search. For example, we tested the 100 worst unsatisfiable 100 variable problems of 100,000 CP problems tested at L/N = 2.6. This point was chosen as it has very bad performance of the worst cases, and is in the middle of the constraint gap. DP augmented with the *(Binary)* rule was able to solve 92 of these 100 problems without needing to search at all. That is, the binary and near-binary clauses contained enough information alone to prove unsatisfiability. By contrast, at the point of worst median performance, L/N = 3.9, only 43 of the 100 worst problems were solved without search.

It can thus be seen that in terms of reducing the amount of search, augmenting DP with the *(Binary)* rule is highly effective. Nevertheless, it is *not* able to eliminate the extremely bad worst case performance in the mostly satisfiable region. Of the 8 of the worst 100 unsatisfiable problems not solved trivially at L/N = 2.6, one required as many as 6,520 branches using the binary rule and another 2,454. By contrast, at L/N = 3.9, the worst of the 57 problems not solved trivially needed only 574 branches using the binary rule. Furthermore, the binary rule is not able to eliminate the constraint gap. The ratio of applications of the binary rule to splits is broadly similar in behaviour to the ratio of units to splits seen in Figure 3 (b), peaking at comparatively large values of L/N.

The fact that the variable behaviour in an otherwise easy region of problems cannot be eliminated by the *(Binary)* rule is particularly significant. This rule was an especially good candidate to eliminate variable behaviour, since we showed above that the worst case problems were associated with minimal unsatisfiable subsets largely containing binary clauses. Furthermore, we have shown in the past [8] that improved branching heuristics seem unable to eliminate the variable behaviour. Thus, neither better heuristics, nor better constraint propagation are able to eliminate variable behaviour.

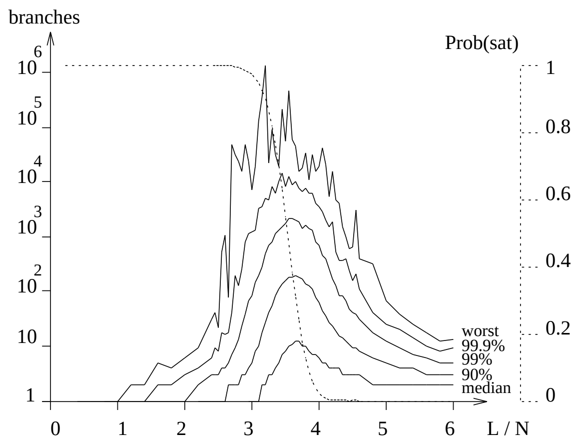


Figure 11: CP problems tested using DP with Intelligent Backtracking, N= 300.

We also implemented a restricted version of the binary rule which just determines the satisfiability of the binary clauses and does not simplify on any of the literals. Although this restricted rule is less expensive, it appears to be of little use in reducing search; for CP at N=100, $2\mathrm{N}p = 3$, it closed at most 20% of branches at large L/N but less than 3% of branches in the region of the constraint gap. It thus had little affect on mean behaviour.

12 Intelligent backtracking

There are three ways in which we can try to improve the behaviour of backtracking algorithms. We can improve the quality of our branching choices, we can increase the work we do between each branch to try to decrease the amount of branching, and we can use information gained during search to improve the quality of backtracking. Improved branching heuristics appear to be unable to eliminate exceptionally hard problems [8]. Also, as we saw in §11, improved constraint propagation between branching points in search appears unable to eliminate the constraint gap or exceptionally hard problems. This leaves open only the question of whether intelligent backtracking can rid us of these difficult problems.

To test this, we used an implementation of the Davis-Putnam procedure written by Mark Stickel which uses a form of intelligent backtracking.[2] When a branch closes, the variable assignments which contributed to the generation of inconsistency are recorded. Backtracking then jumps over all other variable assignments since they are irrelevant to the closing of the branch. Such intelligent backtracking may identify branching points which were made on irrelevant variables outside the minimal unsatisfiable subset. When it succeeds in doing so, a dramatic reduction

[2]This implementation lacks the pure literal deletion rule. This is unlikely to affect results qualitatively since the pure literal rule is not effective in the regions where hard problems occur.

in search is achieved.

Intelligent backtracking greatly reduced the number of exceptionally hard problems for small problem sizes. However, with increasing problem size they seem to reappear. At N = 300, we tested 10,000 problems from the constant probability model with $2\text{N}p = 3$, at each value of L/N from 0.2 to 6.0 in steps of 0.2, and additionally each value from 2.2 to 4.6 in steps of 0.05 to investigate the phase transition in more depth. Figure 11 shows the percentile contours of results, and also the probability of satisfiability. The worst case observed was a *satisfiable* problem that needed 1,386,500 branches at L/N = 3.2, where 90.14% of problems were soluble, and where median behaviour was only 2 branches. This is five orders of magnitude worse than the worst median behaviour of only 12 branches at L/N = 3.7, at which point the worst case we saw was 16,003 branches. Further hard problems were seen at smaller values of L/N, for example one unsatisfiable problem needed 48,269 branches at L/N = 2.7, where 99.52% of problems were soluble.

It is clear from these results that rare but exceptionally hard problems still occur in the mostly soluble and easy region. However, the worst cases may occur at slightly larger values of L/N than before, and not as clearly in the constraint gap. The existence of other hard problems at smaller values of L/N suggests that the constraint gap is still important. To investigate this further, extensive computational experiments will be required, both with larger problem and sample sizes.

Other forms of intelligent backtracking are possible, such as dependency directed backtracking [15]. Baker has shown that this can eliminate exceptionally hard problems from the underconstrained region in 3-COL for 100-node random graphs [2]. However, it remains an open question whether this or any other form of intelligent backtracking can eliminate such hard problems as problem sizes increase.

13 Related Work

Phase transitions are becoming increasingly important in the study of AI systems. Huberman and Hogg [12] predict that many large scale systems will undergo sudden phase transitions that affect computational performance. They show, for example, that a simple model of heuristic search changes from linear to exponential behaviour at a phase boundary. Cheeseman *et al.* [3] observed that many NP-hard problems (*eg.* graph colouring and Hamiltonian circuits) have a control parameter, that a phase transition between underconstrained (and typically soluble) problems and overconstrained (and typically insoluble) problems occurs at a critical values of this parameter, and that hard problems occur at this critical value. Williams and Hogg [16] give theoretical reasons why the solubility phase transition should coincide with peaks in problem difficulty.

For random 3-SAT and CP, Mitchell *et al.* [14] demonstrated that the parameter is L/N, the ratio of clauses to variables, and that median performance of DP has an easy-hard-easy pattern through the phase transition with the hardest median instance occurring in the phase transition. Although they noted that the mean is influenced by a very small number of very large values, they concentrated solely on the median as they felt that "it appears to be a more informative statistic". Our results suggest that the *distribution* of values is, in fact, of considerable importance in understanding problem difficulty, and that the median alone provides a somewhat incomplete picture. Crawford and Auton have also observed a secondary peak in mean problem difficulty for a tableau based procedure in a mostly satisfiable region of random 3-SAT [4]. However, they failed to observe this peak with DP and therefore speculated that it was probably an artifact of the branching heuristics used by their procedure. In fact it seems more likely that it is an artifact of the statistics they compiled, namely the number of nodes in the search tree. Obviously this is related to depth of search, and we observed a distinctive peak in this at low L/N in CP (see §4) and a similar one in 3-SAT (see §6). This would be enough to account for the secondary peak they report of approximately 13 nodes at N = 50. It was only on much larger problems, with hundreds of variables, that they report occasional extremely hard behaviour. In [8] we were able to show, however, that unusually hard problems do occur at low L/N at N = 50 in 3-SAT using a simplified variant of DP, but that it requires very large sample sizes to be seen. For this effect in 3-SAT to be studied in more detail, larger problems and larger sample sizes will be needed.

Hogg and Williams have observed extremely variable problem difficulty for graph colouring using both a backtracking algorithm based on the Berlaz heuristic and a heuristic repair algorithm [10]. They found that the hardest 3 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 3 colourability phase transition. They propose that these very hard problems are associated with a transition between polynomial and exponential average search cost.

14 Conclusions

We have performed a detailed experimental investigation of the satisfiability phase transition for several different classes of randomly generated problems including the constant probability model, random k-SAT, random mixed SAT, and an encoding of k-COL into SAT. With each problem class, the median problem difficulty for the Davis-Putnam procedure displays an easy-hard-easy pattern with the hardest problems being associated with the satisfiability phase transition. We have shown, however, that the "conventional" picture of easy-hard-easy behaviour is inadequate since the distribution of problem difficulties has several other important features. In particular, all the problem classes have a region of very variable problem difficulty where problems are typically underconstrained and satisfiable.

Within these regions, we have found problems orders of magnitude harder than problems in the middle of the phase transition. These extraordinarily hard problems appear to be associated with a constraint gap, a minimum in the ratio of the constraint propagations to splits. The position and shape of both the constraint gap and the satisfiability phase transition are remarkably consistent with problem size. For example, both occur at fixed ratios of constraints to variables.

We have shown that the hardest problems in the middle of the satisfiability phase transition are more critically constrained between being satisfiable and unsatisfiable than typical problems. By comparison, hard problems in the variable region arise for different reasons. The hardest unsatisfiable problems in the variable region are actually *less* critically constrained than typical unsatisfiable problems. Indeed, the hardest unsatisfiable problems in the variable region often contain a very small and unique minimal unsatisfiable subset. The difficulty in solving such problems is thus in identifying the minimal unsatisfiable subset from the many irrelevant clauses. All branching points or constraint propagations during search which do not affect a variable in the minimal unsatisfiable subset represent entirely wasted work. As each branching point doubles the search space, search time can be exponential in the number of irrelevant choices. On the other hand, hard satisfiable problems in the variable region are much more critically constrained than typical satisfiable problems. As a result, an incorrect assignment at an early stage in search leads to an unsatisfiable subproblem, which then behaves like hard unsatisfiable problems in the same region, and contains a small and unique minimal unsatisfiable subset.

The difficulty in solving the hardest unsatisfiable problems away from the probability phase transition is in identifying a small minimal unsatisfiable subset. The existence of a constraint gap greatly hinders our ability to find such minimal unsatisfiable subsets. Despite the fact that the minimal unsatisfiable subsets contain many binary clauses, or reduce to a large number of binary clauses after just one variable assignment, the addition of a rule to the Davis-Putnam procedure to propagate on binary or near-binary constraints reduces but does not eliminate very variable behaviour. We have shown elsewhere that improved branching heuristics also do not eliminate this behaviour. The fact that neither heuristically nor non-heuristically based improvements are able to eliminate this behaviour suggests that it is likely to be fundamental to the problem class. It remains possible that some form of intelligent backtracking might eliminate exceptionally hard problems or the constraint gap. Our results on one form of intelligent backtracking tentatively suggest otherwise.

Given the range of problem classes used in our experiments, we believe that these results may generalise to other SAT problem classes, and perhaps even to the phase transitions of other NP-hard problems. The identification of factors like the constraint gap which help to make instances of an NP-complete problem like SAT hard should be useful both empirically for testing and improving algorithms and theoretically for understanding the fine detail of NP-hardness.

Acknowledgements

This first author was supported by a SERC Postdoctoral Fellowship and the second by a HCM Postdoctoral fellowship. Part of this research was conducted whilst the second author was at INRIA-Lorraine, Nancy, France. We thank Alan Bundy, Pierre Lescanne, the members of the Mathematical Reasoning Group at Edinburgh (supported by SERC grant GR/H 23610), and the Eureca group at INRIA-Lorraine for their constructive comments and for quite a few CPU cycles. We also thank Mark Stickel for the implementation he kindly gave us. Finally we thank Alan Bundy, Paul Purdom, and a referee for this journal for comments which improved this paper.

References

[1] B. Aspvall, M.F. Plass, and R.E. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8:121–123, 1979.

[2] Andrew Baker. Personal communication. 1995.

[3] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of the 12th IJCAI*, pages 331–337. International Joint Conference on Artificial Intelligence, 1991.

[4] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 21–27. AAAI Press/The MIT Press, 1993.

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

[6] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. Association for Computing Machinery*, 7:201–215, 1960.

[7] O. Dubois, P. Andre, Y. Boufkhad, and J. Carlier. SAT versus UNSAT. Presented at the Second DIMACS Challenge Workshop, October 1993.

[8] Ian P. Gent and Toby Walsh. Easy problems are sometimes hard. *Artificial Intelligence*, 70:335–345, 1994.

[9] Ian P. Gent and Toby Walsh. The SAT phase transition. In *Proceedings of ECAI-94*, pages 105-109, ed. A.G. Cohn, John Wiley & Sons, 1994.

[10] T. Hogg and C. Williams. The hardest constraint problems: A double phase transition. *Artificial Intelligence*, 69:359–377, 1994.

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

[12] B.A. Huberman and T. Hogg. Phase Transitions in Artificial Intelligence Systems. *Artificial Intelligence*, 33:155–171, 1987.

[13] S. Kirkpatrick, G. Györgyi, N. Tishby, and L. Troyansky. The statistical mechanics of k-satisfaction. *Advances in Neural Information Processing Systems 6*, pages 439–446, ed. J.D. Cowan, G. Tesauro and J. Alspector, Morgan Kaufmann, San Francisco, 1994.

[14] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *Proceedings, 10th National Conference on Artificial Intelligence*, pages 459-465. AAAI Press/The MIT Press. 1992.

[15] R.M. Stallman and G.J. Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9:135–196, 1977.

[16] C.P. Williams and T. Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.