Journal of Artificial Intelligence Research 28 (2007) 107-118 Submitted 9/05; published 2/07

Generating Hard Satisfiable Formulas by Hiding Solutions Deceptively

Haixia Jia HJIA@CS.UNM.EDU
Computer Science Department
University of New Mexico
Cristopher Moore MOORE@SANTAFE.EDU
Computer Science Department
University of New Mexico
Doug Strain DOUG.STRAIN@GMAIL.COM
Computer Science Department
University of New Mexico

Abstract

To test incomplete search algorithms for constraint satisfaction problems such as 3-SAT, we need a source of hard, but satisfiable, benchmark instances. A simple way to do this is to choose a random truth assignment A, and then choose clauses randomly from among those satisfied by A. However, this method tends to produce easy problems, since the majority of literals point toward the "hidden" assignment A. Last year, Achlioptas, Jia and Moore proposed a problem generator that cancels this effect by hiding both A and its complement $\overline{A}$ (Achlioptas, Jia, & Moore, 2004). While the resulting formulas appear to be just as hard for DPLL algorithms as random 3-SAT formulas with no hidden assignment, they can be solved by `WalkSAT` in only polynomial time.

Here we propose a new method to cancel the attraction to A, by choosing a clause with $t > 0$ literals satisfied by A with probability proportional to q^t for some $q < 1$. By varying q, we can generate formulas whose variables have no bias, i.e., which are equally likely to be true or false; we can even cause the formula to "deceptively" point away from A. We present theoretical and experimental results suggesting that these formulas are exponentially hard both for DPLL algorithms and for incomplete algorithms such as `WalkSAT`.

1. Introduction

To evaluate search algorithms for constraint satisfaction problems, we need good sources of benchmark instances. Real-world problems are the best benchmarks by definition, but each such problem has structures specific to its application domain; in addition, if we wish to study how the running times of search algorithms scale, we need entire families of benchmarks with varying size and density.

One way to fill this need is to generate *random* instances. For instance, for 3-SAT we can generate instances with n variables and m clauses by choosing each clause uniformly from among the $8\binom{n}{3}$ possibilities. We can then vary these formulas according to their size and their density $r = m/n$. While such formulas lack much of the structure of real-world instances, they have been instrumental in the development and study of new search methods such as simulated annealing (Johnson, Aragon, McGeoch, & Shevon, 1989), the breakout procedure (Morris, 1993), WalkSAT (Selman, Kautz, & Cohen, 1996), and Survey Propagation (Mézard & Zecchina, 2002).

However, if we wish to test *incomplete* algorithms such as WalkSAT and Survey Propagation (SP), we need a source problems that are hard but satisfiable. In contrast, above a critical density $r \approx 4.27$, the random formulas defined above are almost certainly unsatisfiable. Random formulas at this threshold appear to be quite hard for complete solvers (Cheeseman, Kanefsky, & Taylor, 1991; Mitchell, Selman, & Levesque, 1992; Hogg, Huberman, & Williams, 1996); but for precisely this reason, it is not feasible to generate large problems at the threshold and then filter out the unsatisfiable ones. While other classes of satisfiable CSPs have been proposed, such as the quasigroup completion problem (Shaw, Stergiou, & Walsh, 1998; Kautz, Ruan, Achlioptas, Gomes, Selman, & Stickel, 2001; Achlioptas, Gomes, Kautz, & Selman, 2000), we would like to have problems generators that are "native" to 3-SAT.

A natural way to generate random satisfiable 3-SAT formulas is to choose a random truth assignment $A \in \{0,1\}^n$, and then choose m clauses uniformly and independently from among the $7\binom{n}{3}$ clauses satisfied by A. The problem with this is that simply rejecting clauses that conflict with A causes an unbalanced distribution of literals; in particular, on average a literal will agree with its value in the hidden assignment $4/7$ of the time. Thus, especially when there are many clauses, a simple majority heuristic or local search will quickly find A. More sophisticated versions of this "hidden assignment" scheme (Asahiro, Iwama, & Miyano, 1996; Van Gelder, 1993) improve matters somewhat but still lead to biased samples. Thus the question is how to avoid this "attraction" to the hidden assignment,

One approach (Achlioptas et al., 2004) is to choose clauses uniformly from among those that are satisfied by both A and its complement $\overline{A}$. This is inspired by recent work on random k-SAT and Not-All-Equal SAT (Achlioptas & Moore, 2002b), in which symmetry with respect to complementation reduces the variance of the number of solutions; the idea is that A and $\overline{A}$ cancel each others' attractions out, making either one hard to find. Indeed, the resulting formulas appear to take DPLL solvers exponential time and, in general, to be just as hard as random 3-SAT formulas with no hidden assignment. On the other hand, WalkSAT solves these formulas in polynomial time, since after a few variables are set in a way that agrees with one of the hidden assignments, neighboring variables develop correlations consistent with these (Barthel, Hartmann, Leone, Ricci-Tersenghi, Weigt, & Zecchina, 2002).

In this paper, we pursue an alternate approach, inspired by Achlioptas and Preres, who reweighted the satisfying assignments in a natural way (Achlioptas & Peres, 2003). We hide just one assignment, but we bias the distribution of clauses as follows:

1. Predefine a constant $q < 1$ and generate a random truth assignment $A \in \{0,1\}^n$
2. Do rn times: choose a random k-tuple of variables, and choose from among the clauses in which $t > 0$ literals are satisfied by A with probability proportional to q^t.

This penalizes the clauses which are "more satisfied" by A, and reduces the extent to which variable occurrences are more likely to agree with A. (Note that the naive formulas discussed above amount to the case $q = 1$.) As we will see below, by choosing q appropriately we can rebalance the distribution of literals, so that each variable is as likely to appear positively as often as negatively and no longer points toward its value in A. By reducing q further, we can even make it more likely that a variable occurrence *disagrees* with A, so that the formula becomes "deceptive" and points away from the hidden assignment.

We call these formulas "q-hidden," to distinguish them from the naive "1-hidden" formulas discussed above, the "2-hidden" formulas studied by Achlioptas, Jia and Moore (Achlioptas et al., 2004), and the "0-hidden" formulas consisting of random 3-SAT formulas with no hidden assignment. Like these other families, our q-hidden formulas are readily amenable to all the mathematical tools that have been developed for studying random k-SAT formulas, including moment calculations and the method of differential equations. Below we calculate the expected density of satisfying assignments as a function of their distance from A, and analyze the behavior of the Unit Clause (`UC`) algorithm on q-hidden formulas. We then present experiments on several complete and incomplete solvers. For certain values of q, we find that our q-hidden formulas are just as hard or harder for DPLL algorithms as 0-hidden formulas or 2-hidden formulas, and are much harder than naive 1-hidden formulas. In addition, we find that local search algorithms like `WalkSAT` find our formulas much harder than any of these other families, taking exponential as opposed to polynomial time. Moreover, the running time of `WalkSAT` increases sharply as our formulas become more deceptive.

2. The Expected Density of Solutions and the Bias of Local Search

For $\alpha \in [0,1]$, let X_α be the number of satisfying truth assignments in a random q-hidden k-SAT formula that agree on a fraction α of the variables with the hidden assignment A; that is, their Hamming distance from A is $(1-\alpha)n$. We wish to calculate the expectation $\mathbf{E}[X_\alpha]$.

By symmetry, we can take A to be the all-true assignment. In that case, a clause with $t > 0$ positive literals is chosen with probability $q^t/((1+q)^k - 1)$ (here we normalize the probabilities by summing over the $\binom{k}{t}$ clauses for all $t > 0$). Let B be a truth assignment where αn of the variables are true and $(1-\alpha)n$ are false. Then, analogous to the calculation by Achlioptas, Jia and Moore (Achlioptas et al., 2004), we use linearity of expectation, independence between clauses, the selection of the literals in each clause with replacement, and Stirling's approximation for the factorial to obtain (where $\sim$ suppresses terms polynomial in n):

$$\begin{aligned}
\mathbf{E}[X_\alpha] &= \binom{n}{\alpha n} \Pr[B \text{ satisfies a random clause}]^m \\
&= \binom{n}{\alpha n} \left(1 - \sum_{t=1}^{k} \binom{k}{t} \frac{q^t (1-\alpha)^t \alpha^{k-t}}{(1+q)^k - 1}\right)^m \\
&\sim f_{k,r,q}(\alpha)^n
\end{aligned}$$

where

$$f(\alpha) = \frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}} \left(1 - \frac{(q(1-\alpha)+\alpha)^k - \alpha^k}{(1+q)^k - 1}\right)^r \quad .$$

Looking at Figure 1, we see that the behavior of f near $\alpha = 1/2$ changes dramatically as we vary q. For $q = 1$ (i.e., naive 1-hidden formulas), $f'(1/2)$ is positive. On the other

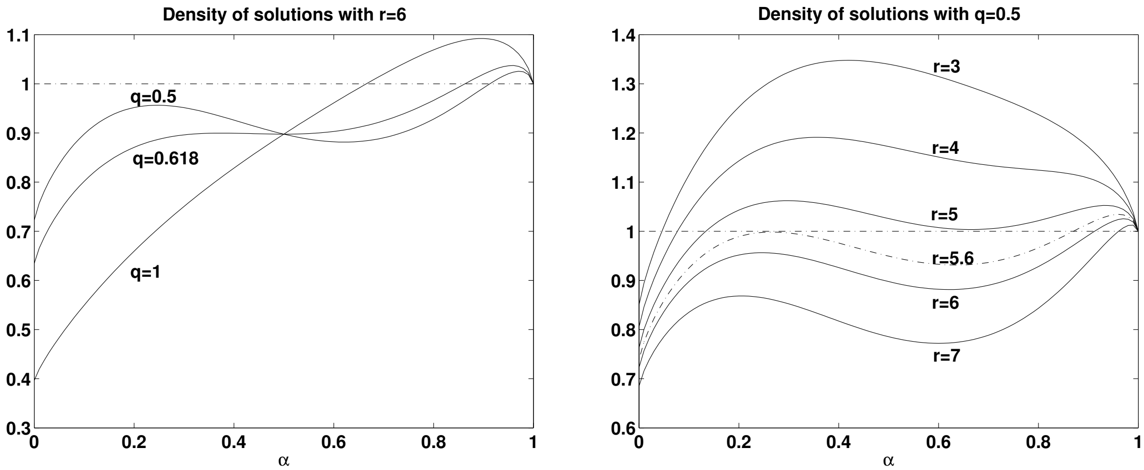

Figure 1: The nth root $f(\alpha)$ of the expected number of solutions which agree with the hidden assignment on a fraction α of the variables. Here $k=3$. The left part of the figure shows $f(\alpha)$ for $q=1$, $q=0.618$ and $q=0.5$ at $r=6$. The right part shows $f(\alpha)$ for $q=0.5$ and varying r. Note that at $r=5.6$, we have $f(\alpha)<1$ for all $\alpha \leq 1/2$.

hand, if q is the positive root q^* of

$$1-(1-q)(1+q)^{k-1}=0 \tag{1}$$

then $f'(1/2)=0$. We call the resulting q^*-hidden formulas *balanced*; for $k=3$, q^* is the golden ratio $(\sqrt{5}-1)/2=0.618...$

This choice of q affects local search algorithms such as `WalkSAT` in the following way. If we start with a random assignment B, a step of `WalkSAT` chooses a random unsatisfied clause, and satisfies a literal ℓ chosen randomly from that clause. The expected change in the Hamming distance $d(A,B)$ is then the probability that ℓ agrees with A, minus the probability that it doesn't. Since all the clauses are equally likely to be unsatisfied by a random assignment, this is the expectation of $2t/k-1$ in a random clause, namely

$$\mathbf{E}[\Delta d(A,B)] = \sum_{t=1}^{k} \binom{k}{t} \frac{q^t(2t/k-1)}{(1+q)^k-1} = \frac{1-(1-q)(1+q)^{k-1}}{(1+q)^k-1} \ .$$

Thus is zero when (1) holds, in which case `WalkSAT` is equally likely to move toward or away from A. Thus, analogous to the calculation by Achlioptas and Peres (Achlioptas & Peres, 2003), when $q=q^*$ a given literal is equally likely to agree or disagree with A, and `WalkSAT` has no information about in which direction the hidden assignment lies. (This argument applies to the first $o(n^{1/2})$ steps of `WalkSAT`, since until then it is unlikely to have seen any variable twice).

For smaller values of q such as $q = 0.5$ shown in Figure 1, $f'(1/2)$ becomes negative, and we expect a local search algorithm starting at a random assignment to move *away* from A. Indeed, $f(\alpha)$ has a local maximum at some $\alpha < 1/2$, and for small r there are solutions with $\alpha < 1/2$. When r is sufficiently large, however, $f(\alpha) < 1$ for all $\alpha < 1/2$, and as $n \to \infty$ the probability any of these "alternate" solutions exist is exponentially small. We conjecture that for each $q \leq q^*$ there is a threshold $r_c(q)$ at which with high probability the only solutions are those close to A. Setting $\max\{f(\alpha) \mid \alpha \leq 1/2\} = 1$ yields an upper bound on $r_c(q)$, which we show in Figure 4 below. For instance, the dotted line in Figure 1 shows that $r_c(0.5) \leq 5.6$.

We call such formulas *deceptive*, since local search algorithms such as `WalkSAT`, DPLL algorithms such as `zChaff` that use a majority heuristic in their splitting rule, and message-passing algorithms such as `SP` will presumably search in the wrong direction, and take exponential time to cross the local minimum in $f(\alpha)$ to find the hidden assignment. Our experiments below appear to confirm this intuition. In addition, all three types of algorithms appear to encounter the most difficulty at roughly the same density $r_c(q)$, where we conjecture the "alternate" solutions disappear.

3. Unit Clause Heuristic and DPLL Algorithms

Unit Clause (`UC`) is a linear-time heuristic which permanently sets one variable in each step as follows: if there are any unit clauses, satisfy them; otherwise, pick a random literal and satisfy it. For random 3-SAT formulas, `UC` succeeds with constant probability for $r < 8/3$, and fails with high probability for $r > 8/3$ (Chao & Franco, 1986). `UC` can be thought as the first branch of a simple DPLL algorithm S, whose splitting rule takes a random unset variable and tries its truth values in random order; thus `UC` succeeds if S succeeds without backtracking. On the other hand, it was showed that S's expected running time is exponential in n for any $r > 8/3$ (Cocco & Monasson, 2004; Cocco, Monasson, Montanari, & Semerjian, 2005); also Achlioptas, Beame and Molloy used lower bounds on resolution complexity to show that S takes exponential time with high probability if $r > 3.81$ (Achlioptas, Beame, & Molloy, 2001). In general, it appears that simple DPLL algorithms begin to take exponential time at exactly the density where the corresponding linear-time heuristic fails.

In this section, we analyze the performance of `UC` on our q-hidden formulas. Specifically, we show that in the balanced case where $q = q^*$, `UC` fails for $r > 8/3$ just as it does for 0-hidden formulas. Based on this, we conjecture that the running time of S, and other simple DPLL algorithms, is exponentially large for our formulas at the same density as for 0-hidden ones.

We analyze the behavior of `UC` on arbitrary initial distributions of 3-SAT clauses using the method of differential equations. For simplicity we assume that A is the all-true assignment. A *round* of `UC` consists of a "free step," in which we satisfy a random literal, and the ensuing chain of unit-clause propagations. For $0 \leq i \leq 3$ and $0 \leq j \leq i$, let $S_{i,j} = s_{i,j}n$ be the number of clauses of length i with j positive literals and $i-j$ negative ones, and let $s_i = \sum_j s_{i,j}$. Let $X = xn$ be the number of variables set so far, and let m_T and m_F be the expected number of variables set true and false in a round. Then we can model the discrete stochastic process of the $S_{i,j}$ with the following differential equations for the $s_{i,j}$:

$$\begin{aligned} \frac{\mathrm{d}s_{3,j}}{\mathrm{d}x} &= -\frac{3s_{3,j}}{1-x} \\ \frac{\mathrm{d}s_{2,j}}{\mathrm{d}x} &= -\frac{2s_{2,j}}{1-x} + \frac{m_F(j+1)s_{3,j+1} + m_T(3-j)s_{3,j}}{(m_T+m_F)(1-x)} \end{aligned} \tag{2}$$

The unit clauses are governed by a two-type branching process, with transition matrix

$$M = \frac{1}{1-x}\begin{pmatrix} s_{2,1} & 2s_{2,0} \\ 2s_{2,2} & s_{2,1} \end{pmatrix} .$$

As in the calculation by Achlioptas and Moore (Achlioptas & Moore, 2002a), as long as the largest eigenvalue of M is less than 1, the branching process is subcritical, and summing over the round gives

$$\begin{pmatrix} m_F \\ m_T \end{pmatrix} = (I-M)^{-1} \cdot \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} .$$

We then solve the equation (2) with the initial conditions $s_{3,0} = 0$ and

$$s_{3,j} = \binom{3}{j} \frac{q^j}{(1+q)^3 - 1}$$

for $0 < j \leq 3$. In the balanced case $q = q^*$, we find that UC succeeds on q-hidden formulas with constant probability if and only if $r < 8/3$, just as for 0-hidden formulas. The reason is that, as for 2-hidden formulas, the expected number of positive and negative literals are the same throughout the process. This symmetry causes UC to behave just as it would on random 3-SAT formulas without a hidden assignment.

We note that for $q < q^*$, UC succeeds at slightly higher densities, at which it can find one of the "alternate" solutions with $\alpha < 1/2$. At higher densities where these alternate solutions disappear, our experimental results below show that these "deceptive" formulas take DPLL algorithms exponential time, and for $r > r_c(q)$ they are harder than 0-hidden formulas of the same density.

4. Experimental Results

4.1 DPLL

In this section we discuss the behavior of DPLL solvers on our q-hidden formulas. We focus on `zChaff` (Zhang, 2002); the behavior of `OKsolver` (Kullmann, 2002) is similar. Figure 2 shows `zChaff`'s running time on 0-hidden, 1-hidden, 2-hidden, and q-hidden formulas for various values of q.

Balanced formulas, i.e. with $q = q^* = 0.618...$, appear to be about as hard as 0-hidden ones, including above the satisfiability threshold $r \approx 4.27$ where 0-hidden formulas become unsatisfiable. Like 0-hidden formulas, these q^*-hidden formulas appear to peak in complexity near the satisfiability threshold. This is consistent with the picture given in the previous two sections: namely, that these "balanced" formulas make it impossible for algorithms to feel the attraction of the hidden assignment. In contrast, naive 1-hidden formulas are far easier, since the attraction to the hidden assignment is strong.

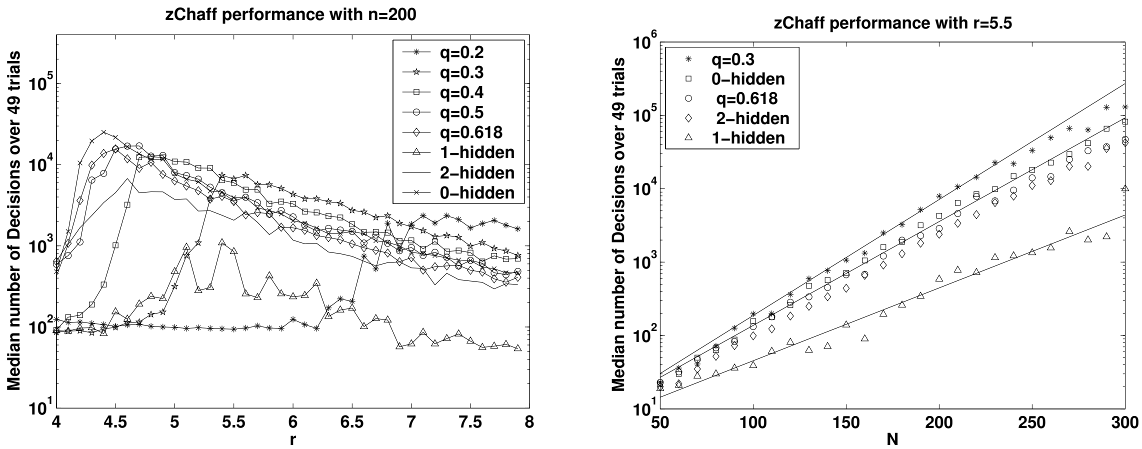

Figure 2: The left part of the figure shows `zChaff`'s median running time over 49 trials on 0-hidden, 1-hidden, 2-hidden and q-hidden formulas with $n = 200$ and r ranging from 4.0 to 8.0. The right part shows the median running time with $r = 5.5$ and n ranging from 50 to 300. Note that 0-hidden formulas are almost always unsatisfiable for $r > 4.27$.

Deceptive formulas, i.e. with $q < q^*$, appear to have two phases. At low density they are relatively easy, and their hardness peaks at a density $r_c(q)$. Above $r_c(q)$ they take exponential time; as for 0-hidden formulas, as r increases further the coefficient of the exponential decreases as the clauses generate contradictions more quickly.

We believe that this peak $r_c(q)$ is the same threshold density defined earlier (see Figure 4 below) above which the only solutions are those close to the hidden assignment. The situation seems to be the following: below $r_c(q)$, there are "alternate" solutions with $\alpha < 1/2$, and `zChaff` is led to these by its splitting rule. Above $r_c(q)$, these alternate solutions disappear, and `zChaff` takes exponential time to find the vicinity of the hidden assignment, since the formula deceptively points in the other direction. Moreover, for a fixed r above $r_c(q)$ these formulas become harder as q decreases and they become more deceptive.

To illustrate this further, the right part of Figure 2 shows `zChaff`'s median running time on 0-hidden formulas, 1-hidden formulas, 2-hidden formulas, and q-hidden formulas for $q = q^*$ (balanced) and $q = 0.3$ (deceptive). We fix $r = 5.5$, which appears to be above $r_c(q)$ for both these values of q. At this density, the 0-hidden, 2-hidden, and balanced q-hidden formulas are all comparable in difficulty, while 1-hidden formulas are much easier and the deceptive formulas appear to be somewhat harder.

4.2 `SP`

Survey Propagation or `SP` (Mézard & Zecchina, 2002) is a recently introduced incomplete solver based on insights from the replica method of statistical physics and a generalization of belief propagation. We tested `SP` on 0-hidden formulas and q-hidden formulas for different values of q, using $n = 10^4$ and varying r. For 0-hidden formulas, `SP` succeeds up to $r = 4.25$, quite close to the satisfiability threshold. For q-hidden formulas with $q = q^*$, SP fails at 4.25 just as it does for 0-hidden formulas, suggesting that it finds these formulas exactly as hard as 0-hidden ones even though they are guaranteed to be satisfiable. For naive 1-hidden formulas, SP succeeds at a significantly higher density, up to $r = 5.6$.

Presumably the naive 1-hidden formulas are easier for SP since the "messages" from clauses to variables, like the majority heuristic, tend to push the algorithm towards the hidden assignment. In the balanced case $q = q^*$, this attraction is successfully suppressed, causing SP to fail at essentially the same density as for 0-hidden formulas, close to the satisfiability threshold, even though our q-hidden formulas continue to be satisfiable at all densities. In contrast, the 2-hidden formulas proposed by Achlioptas, Jia and Moore (Achlioptas et al., 2004) are solved by SP up to a somewhat higher density $r \approx 4.8$. Thus it seems that the reweighting approach of q-hidden formulas does a better job of confusing SP than hiding two complementary assignments does.

For $q < q^*$, SP succeeds up to somewhat higher densities, each of which matches quite closely the value $r_c(q)$ at which zChaff's running time peaks (see Figure 4 below). Building on our conjecture that this is the density above which the only solutions are those close to the hidden assignment, we guess that SP succeeds for $r < r_c(q)$ precisely because the local gradient in the density of solutions pushes it towards the "alternate" solutions with $\alpha < 1/2$. Above $r_c(q)$, these solutions no longer exist, and SP fails because the clauses send deceptive messages, demanding that variables be set opposite to the hidden assignment.

4.3 WalkSAT

We conclude with a local search algorithm, WalkSAT. For each formula, we did up to 10^4 restarts, with 10^4 steps per attempt, where each step does a random or greedy flip with equal probability. In the left part of Figure 3 we measure WalkSAT's performance on 1-hidden, 2-hidden, and q-hidden formulas with various values of q. We use $n = 200$ and r range from 4 to 8. Even for these relatively small formulas, we see that for the three most deceptive values of q, there is a density at which the median running time jumps to 10^8 flips. For instance, q-hidden formulas with $q = 0.4$ appear to be unfeasible for WalkSAT for, say, $r > 5$.

We believe that, consistent with the discussion above, local search algorithms like WalkSAT greedily follow the gradient in the density of solutions $f(\alpha)$. For $q < q^*$, this gradient is deceptive, and lures WalkSAT away from the hidden assignment. At densities below $r_c(q)$, there are many alternate solutions with $\alpha < 1/2$ and WalkSAT finds one of them very easily; but for densities above $r_c(q)$, the only solutions are those near the hidden assignment, and WalkSAT's greed causes it to wander for an exponentially long time in the wrong region. This picture is supported by the fact that, as Figure 4 shows below, the density at which WalkSAT's running time jumps upward closely matches the thresholds $r_c(q)$ that we observed for zChaff and SP.

The right part of Figure 3 looks at WalkSAT's median running time at a fixed density as a function of n. We compare 1-hidden and 2-hidden formulas with q-hidden ones with $q = q^*$ and two deceptive values, 0.5 and 0.3. We choose $r = 5.5$, which is above $r_c(q)$ for all three values of q. The running time of 1-hidden and 2-hidden formulas is only polynomial (Achlioptas et al., 2004; Barthel et al., 2002). In contrast, even in the balanced case $q = q^*$, the running time is exponential, and the slope of this exponential increases dramatically as we decrease q and make the formulas more deceptive. We note that it might be possible to develop a heuristic analysis of `WalkSAT`'s running time in the deceptive case (Semerjian & Monasson, 2003; Cocco et al., 2005).

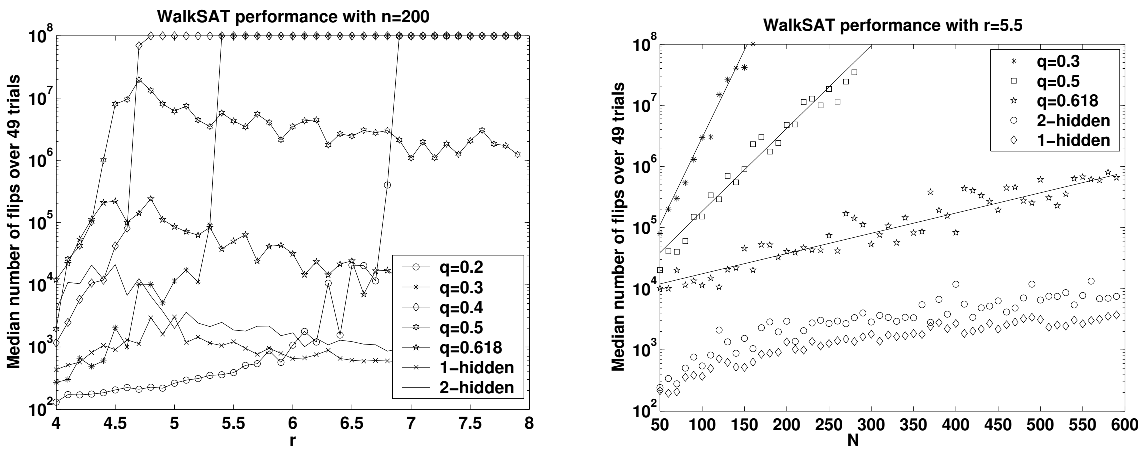

Figure 3: The left part of the figure shows `WalkSAT`'s median running time over 49 trials with $n = 200$ and r ranging from 4 to 8; the right part shows the median running time with $r = 5.5$ and n ranging from 50 to 600.

5. The Threshold Density

As we have seen, there appears to be a characteristic density $r_c(q)$ for each value of $q \le q^*$ at which the running time of DPLL algorithms like `zChaff` peaks, at which `WalkSAT`'s running time becomes exponential, and at which `SP` ceases to work. We conjecture that in all three cases, the key phenomenon at this density is that the solutions with $\alpha < 1/2$ disappear, leaving only those close to the hidden assignment. Figure 4 shows our measured values of $r_c(q)$, and indeed they are quite close for the three algorithms. We also show the analytic upper bound on $r_c(q)$ resulting from setting $\max\{f(\alpha) \mid \alpha \le 1/2\} = 1$, above which the expected number of solutions with $\alpha \le 1/2$ is exponentially small.

6. Conclusions

We have introduced a simple new way to hide solutions in 3-SAT problems that produces instances that are both hard and satisfiable. Unlike the 2-hidden formulas proposed by Achlioptas, Jia and Moore (Achlioptas et al., 2004) where the attraction of the hidden assignment is cancelled by also hiding its complement, here we eliminate this attraction by reweighting the distribution of clauses as proposed by Achlioptas and Peres (Achlioptas & Peres, 2003). Indeed, by going beyond the value of the parameter q that makes our q-hidden formulas balanced, we can create *deceptive* formulas that lead algorithms in the wrong direction. Experimentally, our formulas are as hard or harder for DPLL algorithms

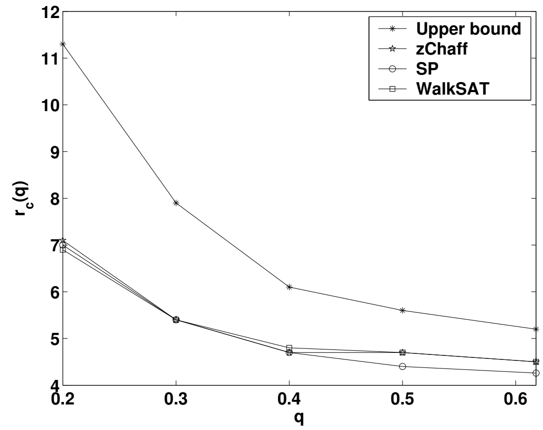

Figure 4: The density $r_c(q)$ at which the running time of `zChaff` peaks, `WalkSAT` peaks or exceeds 10^8 flips, and `SP` stops working. We conjecture all of these events occur because at this density the alternate solutions with $\alpha < 1/2$ disappear, leaving only those close to the hidden assignment. Shown also is the analytic upper bound described in the text.

as 0-hidden formulas, i.e., random 3-SAT formulas without a hidden assignment; for local search algorithms like `WalkSAT`, they are much harder than 0-hidden or 2-hidden formulas, taking exponential rather than polynomial time. Our formulas are also amenable to all the mathematical tools developed for the study of random 3-SAT; here we have calculated their expected density of solutions as a function of distance from the hidden assignment, and used the method of differential equations to show that `UC` fails for them at the same density as it does for 0-hidden formulas.

We close with several exciting directions for future work:

1. Confirm that there is a single threshold density $r_c(q)$ at which a) the alternate solutions far from the hidden assignment disappear, b) the running time of DPLL algorithms is maximized, c) `SP` stops working, and d) the running time of `WalkSAT` becomes exponential;

2. Prove that simple DPLL algorithms take exponential time for $r > r_c(q)$, in expectation or with high probability;

3. Calculate the variance of the number of solutions as a function of α, and giving improved upper and lower bounds on the distribution of solutions and $r_c(q)$.

Acknowledgments

H.J. is supported by an NSF Graduate Fellowship. C.M. and D.S. are supported by NSF grants CCR-0220070, EIA-0218563, and PHY-0200909. C.M. thanks Tracy Conrad and Rosemary Moore for their support.

References

Achlioptas, D., Beame, P., & Molloy, M. (2001). A sharp threshold in proof complexity. In *Proc. STOC*, pp. 337–346.

Achlioptas, D., Gomes, C., Kautz, H., & Selman, B. (2000). Generating satisfiable problem instances. In *Proc. AAAI*, pp. 256–261.

Achlioptas, D., Jia, H., & Moore, C. (2004). Hiding satisfying assignments: two are better than one. In *Proc. AAAI*, pp. 131–136.

Achlioptas, D., & Moore, C. (2002a). Almost all graphs with average degree 4 are 3-colorable. In *Proc. STOC*, pp. 199–208.

Achlioptas, D., & Moore, C. (2002b). The asymptotic order of the random k-SAT threshold. In *Proc. FOCS*, pp. 779–788.

Achlioptas, D., & Peres, Y. (2003). The threshold for random k-SAT is $2^k(\ln 2 - o(k))$. In *Proc. STOC*, pp. 223–231.

Asahiro, Y., Iwama, K., & Miyano, E. (1996). Random generation of test instances with controlled attributes. *DIMACS Series in Disc. Math. and Theor. Comp. Sci.*, *26*.

Barthel, W., Hartmann, A., Leone, M., Ricci-Tersenghi, F., Weigt, M., & Zecchina, R. (2002). Hiding solutions in random satisfiability problems: A statistical mechanics approach. *Phys. Rev. Lett.*, *88*(188701).

Chao, M., & Franco, J. (1986). Probabilistic analysis of two heuristics for the 3-satisfiability problem. *SIAM J. Comput.*, *15*(4), 1106–1118.

Cheeseman, P., Kanefsky, R., & Taylor, W. (1991). Where the really hard problems are. In *Proc. IJCAI*, pp. 163–169.

Cocco, S., & Monasson, R. (2004). Heuristic average-case analysis of the backtrack resolution of random 3-satisfiability instances. *Theor. Comp. Sci.*, *320*, 345–372.

Cocco, S., Monasson, R., Montanari, A., & Semerjian, G. (2005). Approximate analysis of search algorithms with "physical" methods. In Percus, A., Istrate, G., & Moore, C. (Eds.), *Computational Complexity and Statistical Physics*. Oxford University Press.

Hogg, T., Huberman, B., & Williams, C. (1996). Phase transitions and complexity. *Artificial Intelligence*, *81*.

Johnson, D., Aragon, C., McGeoch, L., & Shevon, C. (1989). Optimization by simulated annealing: an experimental evaluation. *Operations Research*, *37*(6), 865–892.

Kautz, H., Ruan, Y., Achlioptas, D., Gomes, C., Selman, B., & Stickel, . (2001). Balance and filtering in structured satisfiable problems. In *Proc. IJCAI*, pp. 351–358.

Kullmann, O. (2002). Investigating the behaviour of a SAT solver on random formulas. Tech. rep. CSR 23-2002, University of Wales Swansea.

Mézard, M., & Zecchina, R. (2002). Random k-satisfiability: from an analytic solution to a new efficient algorithm. *Phys. Rev. E*, *66*, 056126.

Mitchell, D., Selman, B., & Levesque, H. (1992). Hard and easy distributions of SAT problems. In *Proc. AAAI*, pp. 459–465.

Morris, P. (1993). The breakout method for escaping from local minima. In *Proc. AAAI*, pp. 40–45.

Selman, B., Kautz, H., & Cohen, B. (1996). Local search strategies for satisfiability testing. In *Proc. 2nd DIMACS Challange on Cliques, Coloring, and Satisfiability*.

Semerjian, G., & Monasson, R. (2003). A study of pure random walk on random satisfiability problems with "physical" methods. *LNCS*, *2919*, 120–134.

Shaw, P., Stergiou, K., & Walsh, T. (1998). Arc consistency and quasigroup completion. In *Proc. ECAI, workshop on binary constraints*.

Van Gelder, A. (1993). Problem generator `mkcnf.c`. In *Proc. DIMACS*. Challenge archive.

Zhang, L. (2002). zChaff.. ee.princeton.edu/˜chaff/zchaff.php.