On the stupid algorithm for satisfiability

Ian P. Gent[*]

Report APES-03-1998, April 1998[†]

Abstract

I introduce an extremely stupid algorithm for the satisfiability problem. Where random 3-SAT problems are generated randomly to agree with a predetermined truth assignment, I show that the algorithm almost certainly solves problems with N variables and $O(N \ln N)$ clauses very easily. This helps confirm two pieces of folklore. First, that random problems generated in this way are usuually easy. Second, that extrapolating theoretical results beyond their range of direct application must be done with some care.

1 On "On the greedy algorithm for satisfiability"

In [1], Koutsoupias and Papadimitriou study the greedy algorithm for satisfiability (as they call it). Actually it's not an algorithm, but a heuristic. This is a very simple algorithm.

> Choose a truth assignment at random. Then choose a variable such that flipping it increases the number of satisfied clauses: do not allow sideways moves. Repeat until no improvement is possible. If this is a solution, report success. Otherwise give up.

The study then mathematically shows that for 3-SAT, this process will find a solution for 'almost all' soluble 3-SAT problems.

In the introduction, Koutsoupias and Papadimitriou state that (my emphasis)

> Naturally, there is nothing surprising or original about an NP-complete problem with a good average-case algorithm under the most natural probabilistic distribution. What is perhaps a little surprising is that the problem is so *fundamental*, the algorithm so *simple*, and the proof so *easy*.

Here, I introduce a *simpler* algorithm, and show with an *easier* proof a stronger result on the same *fundamental* problem class.

The proof will actually follow very closely parts of the proof given by K&P, but missing bits of their proof out that I don't need in my context.

First I introduce the framework used by K&P. Given that the problem is soluble, there is some truth assignment $\hat{T}$ which solves it. A crucial phrase is:

> "If all clauses satisfied by $\hat{T}$ are chosen independently at random with probability p, then ..."

In other words, the random model of clause generation is to generate clauses consistent with a given model. This is known to give very easy problems, though that is not the focus of this note. In fact it turns out that when they talk about "almost all problems" they actually mean the following. Choose $p = 1/2$: then all but an exponentially small fraction of problems are solved first time all

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

[†]APES reports are currently available from `http://www.cs.strath.ac.uk/~apes/apereports.ht ml`

but an exponentially small amount of the time. What is the implication of choosing $p = 1/2$? There are $O(n^3)$ possible 3-SAT clauses, so we have clause sets with n variables and n^3 clauses! All problems with $O(n)$ clauses can easily fit into the exponentially small exceptions to this theorem. It's actually not *quite* as bad as this. For any constant $d > 0$, if $p \geq d/n$ then the greedy algorithm still succeeds all but an exponentially small amount of the time. But this still means that formulas have $O(n^2)$ clauses, and again this allows the result to ignore problems with $O(n)$ clauses.

2 The Stupid Algorithm for Satisfiability

Here is the "stupid algorithm" for satisfiability.

> For each variable in the clause set, note whether it appears more often with positive or negative polarity. Set each variable to true if it appears more often positively and false otherwise.

Note one thing: we do not build up a partial assignment and look for occurrences of a variable in the clauses still to be satisfied. Each variable is based on its occurrences in all clauses, including those already satisfied. This may seem counterintuitive, as it would seem that we should ignore satisfied clauses in attempting to satisfy the remaining clauses. But if we wish to make a judgement about a given variable and whether it should be true or false, we wish to make that judgement based on as much information as possible. Extra information is contained in clauses already satisfied because each variable in each clause was generated independently from the same distribution.

A binomial random experiment is one with two possible results, call them success and failure. Let $B(p, n)$ denote the number of successes out of n experiments if each has a chance p of success. K&P give the following Chernoff bounds that I will need.

$$P[B(p,n) \leq (1-\theta)pn] \leq e^{-\theta^2 np/2} \tag{1}$$

$$P[B(p,n) \geq (1+\theta)pn] \leq e^{-\theta^2 np/3} \tag{2}$$

Recall that all clauses satisfied by $\hat{T}$ are chosen independently with probabibility p. This means that (in theory) the generation method is: consider in turn each possible clause consistent with $\hat{T}$ and include it in the clause set with probability p.

Think of all the clauses containing a given variable V. A set of them (call it V^+) contains V positively, and another set V^- contains V negatively.[1] What are the sizes of these two sets? If we are generating a problem with N variables in a 3-SAT problem, and we have set V positively, there are $(N-1)(N-2)/2$ possible other pairs of variables we could choose. And either variable can be true or false, for 4 possibilities for each pair. There are obviously the same number of other pairs of variables if we have set V negatively. But now there are only 3 possible pairs of values for the other two variables. We cannot set both false since that would make the whole clause inconsistent with $\hat{T}$. So we have

$$\begin{aligned} |V^+| &= 2(N-1)(N-2) \\ |V^-| &= \frac{3}{2}(N-1)(N-2) \end{aligned}$$

Now, the stupid algorithm will get the value of V correct if the random generation happens to pick more clauses out of the set V^+ than out of V^-. This is obviously typically going to happen, since V^+ is larger than V^-. Can we say anything more mathematical than that?

One simple observation (again borrowed from K&P) is as follows. Consider any number k. If we pick more than k elements out of V^+ and fewer than k elements out of V^- we have obviously

[1] Technically, positively or negatively with respect to the truth assignment $\hat{T}$. For simplicity and without loss of generality, I will now assume that $\hat{T}$ is the assignment of every variable to True. I will also assume that clauses always have three different variables in them.

picked more elements out of V^+ than V^-. What is a suitable value for k? Obviously one that splits the difference between the expected number of clauses picked out of V^+ and V^-. So I will use

$$k = \frac{7}{4}(N-1)(N-2)$$

So we have that the probability that the stupid algorithm guesses variable V wrong is no more than the the probability that *either* we pick fewer than k clauses out of V^+ *or* we pick more than k clauses out of V^-. The probability of this is bounded by the sum of the two constituent probabilities. Writing W_V for the event that the stupid algorithm sets variable V wrongly, we have

$$P[W_V] \leq P[B(p,|V^+|) < k] + P[B(p,|V^-|) > k]$$

Now we have that $k = (1-\frac{1}{8})|V^+|$ and $k = (1+\frac{1}{6})|V^-|$. Putting appropriate values of θ into the Chernoff bounds we get:

$$\begin{aligned} P[W_V] &\leq e^{-\frac{1}{64}|V^+|p/2} + e^{-\frac{1}{36}|V^-|p/3} \\ &= e^{-p(N-1)(N-2)/64} + e^{-p(N-1)(N-2)/72} \\ &\leq 2e^{-p(N-1)(N-2)/72} \end{aligned}$$

Of course the probability that the stupid algorithm gets any variable wrong is no more than the sum of the probabilities that it gets each one wrong. (The events are not independent, but the sum provides an upper bound on the probability). Since there are N variables, the sum is just N times each one, so we have:

$$P[\text{stupid algorithm gets all vars right}] \geq 1 - 2Ne^{-p(N-1)(N-2)/72}$$

Suppose we want only a small probability ϵ that the stupid algorithm will make a mistake. In order to make $\epsilon = 2Ne^{-p(N-1)(N-2)/72}$ we should set Then we should have

$$\begin{aligned} \epsilon &= 2Ne^{-p(N-1)(N-2)/72} \\ \epsilon/2N &= e^{-p(N-1)(N-2)/72} \\ \ln(\epsilon/2N) &= -p(N-1)(N-2)/72 \\ p &= \frac{-72\ln(\epsilon/2N)}{(N-1)(N-2)} \end{aligned}$$

Since the total number of possible clauses consistent with $\hat{T}$ is 7 times N choose 3, this gives an average number of clauses C given by

$$\begin{aligned} E(C) &= 7N(N-1)(N-2)/6p \\ &= -84N\ln(\epsilon/2N) \\ &= -84N(\ln\epsilon - \ln 2 - \ln N) \\ &= 84N(\ln N + \ln(\frac{1}{\epsilon}) + \ln 2) \end{aligned}$$

If one, for example, wishes ϵ to decline exponentially with N, for example as e^{-N} we still have

$$E(C) = 84N(2\ln N + \ln 2)$$

In other words, if we have only $O(N\ln N)$ clauses the stupid algorithm will be successful all but an exponential amount of the time. Compare this with the $O(N^2)$ clauses needed by K&P.

3 Conclusions

By a simpler proof than Koutsoupias and Papadimitriou's I have proved that a simpler algorithm than Koutsoupias and Papadimitriou's performs provably well over a larger class of SAT problems than Koutsoupias and Papadimitriou proved their work for.

As you may guess by the name, I do not propose the stupid algorithm as a sensible algorithm for SAT problems. Instead, I suggest two things.

- Problems generated naively to be consistent with a given truth assignment are dead easy, as is known in the folklore. As I quoted above, they call this distribution the "most natural" probabilistic distribution. Most natural perhaps, but unrealistic.
- One has to be very careful indeed assuming theoretical results have some applicability beyond the range of the stated theorems. Koutsoupias and Papadimitriou clearly implied in their paper that their results gave some backing to the likely success of algorithms like GSAT, when they said "It is plausible that the greedy algorithm, or simple modifications, behaves superbly even in this [3-SAT phase transition] environment." The statement is true, but entirely unrelated to Koutsoupias and Papadimitriou's results. Its truth is known because of extensive experiments by the Mitchell, Selman & Levesque [2] and many others. If there were some link, then the stupid algorithm would presumably be even better than GSAT.

Acknowledgements

I am a member of the cross-site APES Research Group[2]. I thank fellow members from both Leeds and Strathclyde Universities. I also thank Tad Hogg for encouraging me to make this report available.

References

[1] Elias Koutsoupias and Christos H. Papadimitriou. On the greedy algorithm for satisfiability. *Information Processing Letters*, 43:53–55, August 1992.

[2] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of AAAI-92*, pages 459–465, 1992.

[2] `http://www.cs.strath.ac.uk/~apes`