Note's on Schöning's Algorithm for 3-SAT
Monday, April 23, 2007
2:36 PM

In 1999, Uwe Schöning found a surprisingly fast algorithm for finding satisfying assignments of 3-SAT instances. It is a randomized algorithm that runs in expected time \((4/3)^n\).

Assume for now that you are given a collection of clauses, \(C_1, \ldots, C_v\), each of which has 3 terms, for which you know there is a satisfying assignment. We will try to find a satisfying assignment by a local search technique. That is, we will start with some truth assignment, and iteratively modify it in an attempt to convert it into a satisfying assignment. For simplicity, we will assume in this lecture that there is exactly one assignment that satisfies all the clauses, \(x^*\). However, this assumption is not necessary.

Let's say that our current assignment is \(x_1, \ldots, x_n\). If this assignment does not satisfy all the clauses, then there is some unsatisfied clause, say \(C_i\). As \(x^*\) satisfies this clause, one of the variables on which it depends must be different in \(x\). So, here's a reasonable way to try to modify \(x\). Pick an unsatisfied clause, pick a variable at random upon which that clause depends, and flip the value of that variable. This gives us at least a 1/3 chance flipping a variable in which \(x\) and \(x^*\) differ. This might not sound like a very good chance, but it is good when \(x\) and \(x^*\) differ in fewer than 1/3 of their values.

And, it is good enough for the following algorithm to do quite well.

Algorithm 1.
Pick a truth assignment \(x\) at random from \(\{0,1\}^n\)
For \(i = 1, \ldots, n\)
    Let \(C\) be an unsatisfied clause, if there is one.
    Choose a random variable in \(C\), and flip its value.

We will show that with probability at least \((2/3)^n\), this algorithm finds a satisfying assignment, if one exists. Actually, we'll do the proof in the case when the satisfying assignment, \(x^*\), is unique. But, the general proof is not too different.

First, assume that our random \(x\) differs from \(x^*\) in exactly \(u\) variables. Note that the chance of this happening is:

\[
2^{-n \binom{n}{u}}
\]

Now, let's consider the probability that in each of the first \(u\) iterations of the algorithm, it chooses to flip a variable in which \(x\) and \(x^*\) differ. In each step, the probability of this is at least 1/3. So, the probability that it happens in each of the first \(u\) steps is \((1/3)^u\). So, the probability that the initial random \(x\) differs from \(x^*\) in \(u\) variables and then corrects each of those variables over the first \(u\) iterations is

\[
2^{-n \binom{n}{u}} \left(\frac{1}{3}\right)^u
\]

So, the chance that Algorithm 1 finds \(x^*\) is at least
To find an algorithm with a better analysis, we'll be slightly less conservative. Instead of asking for the chance that the algorithm finds the solution in the first $u$ steps, we'll ask for the chance that it finds the solution in the first $3u$ steps. We will say that a step is "good" if it flips a variable in which $x$ and $x^*$ differ, and "bad" if it flips a variable in which $x$ and $x^*$ are the same. We'll also call a step "good" if the algorithm has satisfied all the clauses. If at some iteration, the number of good steps exceeds the number of bad steps by $u$, then $x^*$ has been found. In particular, if $2u$ of the first $3u$ steps are good, then the algorithm finds a satisfying assignment. One can show that the probability that $2u$ of the first $3u$ steps are good is at least

$$\binom{3u}{2u} \left(\frac{1}{3}\right)^{2u} \left(\frac{2}{3}\right)^u = \binom{3u}{2u} \left(\frac{1}{3}\right)^{2u} \left(\frac{2}{3}\right)^u$$

By applying Stirling's formula, one can show that for $u \geq 2$

$$\binom{3u}{u} \geq \frac{1}{\sqrt{5u}} \frac{3^{3u}}{2^{2u}}$$

So, if we define an algorithm 2 that runs for $3n$ iterations instead of $n$ iterations, the probability it finds a satisfying assignment is at least

$$\left(\frac{2}{3}\right)^n$$

This means that the expected number of times we need to run algorithm 1 to find $x^*$ is at most $(3/2)^n$. 

\[
\sum_{u=0}^{n} 2^n \binom{n}{u} 2^{-u} = 2^n \sum_{u=0}^{n} \binom{n}{u} 3^{-u} = 2^n \left(1 + \frac{1}{3}\right)^n = \left(\frac{2}{3}\right)^n
\]
\[
\sum_{u=0}^{n} 2^{-u} \binom{n}{u} \left(\frac{3u}{u} \right) \left(\frac{1}{3} \right)^u \left(\frac{2}{3} \right)^{n-u}
\]
\[\geq \frac{1}{15n} 2^{-n} \sum_{u=0}^{n} \left(\frac{3u}{2^u} \right) \frac{1}{3^{2u}} \frac{2^u}{3^u}
\]
\[= \frac{1}{15n} 2^{-n} \sum_{u=0}^{n} \left(\frac{3}{2}\right)^u = \frac{1}{15n} 2^{-n} \left(1 + \frac{1}{2}\right)^n = \frac{1}{15n} \left(\frac{3}{4}\right)^n
\]

So, the expected number of times we need to call Algorithm 2 to find a satisfying assignment is at most \((4/3)^n \ast (5n)^{1/2}\), which is roughly \((4/3)^n\).

To compare this with naïve iteration through all \(2^n\) truth assignments, note that

\[
\log_{\frac{1}{3}} 2 \approx 2.4
\]

So,

\[
2^n \approx \left(\frac{4}{3}\right)^{2.4n}
\]

That means that Schöning’s algorithm can solve instances with about 2.4 times as many variables as the naïve algorithm.