Schöning's Algorithm for 3SAT

The 3SAT problem is NP-complete, and it is believed to have only exponential time algorithm. It is still interesting to see what is the best exponential time algorithm we can get.

If the formula has n variables and m clauses, then the algorithm that simply tries all possible assignments has running time $O((n+m)\cdot 2^n)$. We will show that this can be improved to roughly $O((1.334)^n)$. This result is due to Uwe Schöning, and it is from 1999.

The following observation is simple but quite useful. Let ϕ be a formula, a^* be an assignment that satisfies ϕ, a be an assignment that does not satisfy ϕ, and C be one of the clauses of ϕ not satisfied by a. Then a and a^* differ in at least one of the three variables of C (possibly, they may differ in two or in all of three). Then, if we pick at random one of the three variables of C and flip the value of that variable in a, we have a probability at least 1/3 of getting a new assignment that is closer to a^*, and a probability at most 2/3 of getting one that is further from a^*. (Where closeness of assignments is measured by the number of variables where they differ.)

Suppose now that a and a^* differ in k variables, and consider an algorithm that given a keeps flipping the value of a randomly chosen variable occurring in the first unsatisfied clause, as long as any unsatisfied clause remains. (As in Algorithm S in Figure 1.)

We can see that there is a probability at least $(1/3)^k$ that the algorithm will find a^*, or another satisfying assignment, in k or fewer steps.

If we pick a to be a random assignment, its "distance" k from a^* will typically be around $n/2$, and so the probability that the algorithm finds a satisfying assignment within about $n/2$ steps is at least about $(1/3)^{n/2}$. If we repeat the algorithm $100\cdot 3^{n/2}$ times we will have a high probability of finding a satisfying assignment. So, even if some details are missing, we have essentially described a 3SAT algorithm that runs in time $O((n+m)\cdot(\sqrt{3})^n)$, which is about $(1.78)^n$ and better than 2^n.

Exercise 1 *Show that, in fact, there is also a* deterministic *3SAT algorithm running in time $O((n+m)\cdot(\sqrt{3})^n)$.*

In order to improve the running time from $(1.78)^n$ to $(1.334)^n$ we will improve the analysis in two ways. First, we show that if a is at distance k from a satisfying assignment, and if we set $t=3k$ (instead of $t=k$) in algorithm S, then the probability of finding a satisfying assignment is at least $(1/2)^k$. This is much better then the lower bound $(1/3)^k$ that we got before by considering the case of k consecutive correct choices. Second, instead of restricting only to the case $k=n/2$, we will consider the contribution of all possible values of k to the total proability of correctness of algorithm S.

Claim 1 *If, in algorithm S, $t=3k$ and we pick an assignment a that differs in k variables from a satisfying assignment a^*, then there is a probability at least $\Omega\left(\frac{1}{\sqrt{k}}\left(\frac{1}{2}\right)^k\right)$ that the algorithm finds a satisfying assignment.*

Algorithm S

- Input: 3SAT formula $\phi = C_1 \wedge \cdots \wedge C_m$
- Pick an assignment a uniformly at random
- Repeat at most t times
  - If a satisfies ϕ, return a
  - Else
    - Let C be the first clause not satisfied by a
    - Pick at random a variable x occurring in C
    - Flip the value of x in a

Figure 1: The basic probabilistic algorithm for 3SAT

The analysis is similar to the analysis of the 2SAT algorithm in Note 8, in that we reduce the analysis of the algorithm to the study of a Markov chain.
At each step of the algorithm, consider the distance between a and a^*. The following facts are clearly true:

- The distance is an integer between 0 and n;
- The algorithm succeeds in finding a satisfying assignment if the distance ever reaches zero;
- Every time a variable is flipped, the distance to a^* either increases by one or decreases by one;
- Every time a variable is flipped, there is a probability at least 1/3 that the distance decreases and a probability at most 2/3 that the distance increases.

We can thus model the progress of our algorithm as a Markov chain M arranged as a path, with vertices labelled 0 to n. For every vertex i there is an edge having probability 2/3 that moves to $i+1$ and an edge having probability 1/3 that moves to $i-1$, except for vertex n that has only an edge with probability 1 that moves to $n-1$, and for vertex 0 that has a self-loop with probability 1. The vertex k is the start vertex.
As in the case of the 2SAT analysis, this Markov chain does not model our 3SAT algorithm exactly: the distance between a and a^* possibly moves towards 0 faster in the algorithm than in the Markov chain. But if the Markov chain has a probability p of reaching 0 within t steps starting from vertex k, then it is certainly true that the algorithm has a probability at least p of finding a satisfying assignment within t steps starting from an assignment at distance k from a^*.
In order to study the probability of reaching vertex 0 in our Markov chain, we define yet another Markov chain M' that makes a possibly even slower progress towards zero. The new Markov chain has a vertex for every integer, and an edge with probability 2/3 from i to $i+1$ and an edge with probability 1/3 from i to $i-1$.

Notice that if there is a probability p of going from k to 0 in M' in t steps, then there is a probability at least p of going from k to 0 in M in t or fewer steps. Indeed the only differences between M and M' are that M' may go into negative numbers, but it can do so only after reaching zero first, or otherwise it can take values biggers than n, while M "bounces back" from n and so it consistently stays closer to 0 than M'.

So, what is the probability of going from k to 0 in M' in t steps? If we go from k to 0, we must have made $k+i$ steps in the right direction and i steps in the wrong direction, where $t = k + 2i$. There are $\binom{k+2i}{i}$ ways to do $k+i$ steps in one direction and i in the other, and each of them has probability $(1/3)^{k+i} \cdot (2/3)^i$, and the overall probability is

$$\binom{k+2i}{i} \cdot \left(\frac{1}{3}\right)^{k+i} \cdot \left(\frac{2}{3}\right)^{i}$$

The binomial coefficient gets larger for larger i, but the other factor gets smaller for larger i. It turns out that the probability is optimized for $i = k$.

Then, we have that the probability of going from k to 0 in M' in $3k$ steps is

$$\binom{3k}{k} \cdot \left(\frac{1}{3}\right)^{2k} \cdot \left(\frac{2}{3}\right)^{k} \tag{1}$$

Now we use (a weak version of) Stirling's approximation to estimate the binomial coefficient. We estimate $n! = \Theta(\sqrt{n} \cdot (n/e)^n)$. Then

$$\binom{3k}{k} = \frac{(3k)!}{k!(2k)!} = \Theta\left(\frac{\sqrt{3k} \cdot (3k/e)^{3k}}{\sqrt{k} \cdot (k/e)^k \cdot \sqrt{2k} \cdot (2k/e)^{2k}}\right) = \Theta\left(\frac{1}{\sqrt{k}} \cdot \frac{3^{3k}}{2^{2k}}\right)$$

By substituting this estimation into (1) we get that the probability of going from k to 0 in $3k$ steps in M' is at least $\Omega((1/\sqrt{k}) \cdot (1/2)^k)$, and the probability of going from k to 0 in $3k$ or fewer steps in M is also at least that much. This proves our first claim.

Claim 2 *If we set $t = 3n$ in algorithm S, where n is the number of variables of ϕ, and ϕ is satisfiable, then there is a probability at least $\Omega\left(\frac{1}{\sqrt{n}}\left(\frac{3}{4}\right)^n\right)$ that the algorithm finds a satisfying assignment.*

When we pick a at random, there is a probability $\binom{n}{k} \cdot 2^{-n}$ that a is at distance k from a^*. Conditioned on this event, the probability of finding a satisfying assignment is at least $c \cdot (1/\sqrt{k}) \cdot 2^{-k}$, for some constant c, as proved in Claim 1.

Overall, the probability of finding a satisfying assignment in $3n$ or fewer steps is at least

$$c \cdot \sum_k \frac{1}{\sqrt{k}} \cdot \frac{1}{2^k} \cdot \binom{n}{k} \cdot \frac{1}{2^n} \leq \frac{c}{\sqrt{n}} \sum_k \binom{n}{k} \frac{1}{2^{n+k}} = \frac{c}{\sqrt{n}} \left(\frac{3}{4}\right)^n$$

where the last step follows by considering the binomial expansion of $(1/2 + 1/4)^n$.

Now it follows that if we repeat $100 \cdot (1/c) \cdot \sqrt{n} \cdot (4/3)^n$ times algorithm S with $t = 3n$ we have a very high probability of finding a satisfying assignment for ϕ if one exists. The total running time is $O(n^{1.5}(n+m)(4/3)^n)$.