ICS 280e, Fall 2000

Lecture Notes for November, 2000

Schöning's random-restart hill-climbing k-SAT algorithm

David Eppstein, ICS, UC Irvine

The Algorithm

Schöning [2] considered the following very simple algorithm for k-SAT (i.e. boolean formula satisfiability, in which the formula is a disjunction of k-variable clauses):

Repeat K times:

1. Choose a random assignment of truth values for all variables
2. Repeat $3n$ times:
   (a) Find an unsatisfied clause for the current truth assignment
   (b) Choose randomly one of the k variables from the clause and change its assignment from true to false or vice versa

If this sequence of random experiments ever finds a satisfying assignment, we know that the formula is satisfiable and can halt. Each trial can be performed in time $\mathcal{O}(mn)$, where m is the number of clauses and n the number of variables, so the overall running time is K times a polynomial. The question is, how big does K need to be to have high probability of finding a satisfying assignment?

To analyze this algorithm, assume that the formula is satisfiable, and let A^* be some particular satisfying assignment (choose one arbitrarily if there is more than one). Then, for any other truth assignment A, define $d(A)$ to be the Hamming distance from A to A^*; that is, the number of variables that would have to be flipped to get to the satisfying assignment A^*.

What happens to $d(A)$ in the inner loop of the algorithm? At each step, we pick an unsatisfied clause of the formula. Since this clause is satisfied by A^*, we know that A and A^* differ on at least one of the k variables of this clause. By flipping a randomly chosen variable, we know that with probability at least $1/k$ we choose one of the variables where they differ, and reduce d by one. With probability at most $(k-1)/k$, though, we can increase d by one.

Drunkard's Walk

So, the behavior of d in the inner loop can be described by a random walk, in which we start at some positive integer i and step to $i-1$ with probability $1/k$ or to $i+1$ with probability $(k-1)/k$. If we ever get to zero, we succeed. What is the probability that, starting from i, we ever get to zero?

Let's let $p(i)$ denote this probability. We have a nice recurrence: if we're at position i, then we have probability $1/k$ of moving to $i-1$ (with probability $p(i-1)$ of continuing from there to zero) and probability $(k-1)/k$ of moving to $i+1$ (with probability $p(i+1)$ of continuing from there to zero). So,

$$p(i) = \frac{1}{k}p(i-1) + \frac{k-1}{k}p(i+1).$$

Turning this around, we see that

$$p(i+1) = \frac{k}{k-1}p(i) - \frac{1}{k-1}p(i-1),$$

a recurrence similar to that for the Fibonacci numbers. We could solve this easily if only we knew the base cases. Obviously, $p(0) = 1$, but we still need to calculate $p(1)$.

Consider what can happen if the random walk reaches position one. With probability $1/k$, it moves immediately to position zero and halts. But with probability $(k-1)/k$, it moves to position two. From there, it might continue to remain forever at positions greater than one, or it might (with probability $p(1)$) eventually return to position one. If we ever do return to position one, we again have probability $p(1)$ of ever reaching zero. Thus

$$p(1) = \frac{1}{k} + \frac{k-1}{k}(p(1))^2.$$

This is a quadratic equation, with two roots. We can either use the quadratic formula or direct substitution to see that the roots are actually $1/(k-1)$ and 1. The root we want turns out to be $1/(k-1)$.

Lemma 1

$$p(i) = \frac{1}{(k-1)^i}.$$

Proof: We have seen that this holds for the base cases $p(0) = 1$ and $p(1) = 1/(k-1)$. The result follows by induction, since plugging in this value for $p(i)$ and $p(i-1)$ in the recurrence gives

$$\begin{aligned}
p(i+1) &= \frac{k}{k-1}p(i) - \frac{1}{k-1}p(i-1) \\
&= \frac{k}{k-1} \cdot \frac{1}{(k-1)^i} - \frac{1}{k-1} \cdot \frac{1}{(k-1)^{i-1}} \\
&= \frac{k}{(k-1)^{i+1}} - \frac{k-1}{(k-1)^{i+1}} \\
&= \frac{1}{(k-1)^{i+1}}.
\end{aligned}$$

□

Algorithm Analysis

There are two ways in which the random walk above is not an accurate model of the behavior of $d(A)$ in Schöning's algorithm. First, if we ever find a clause containing two or more variables for which A differs from A^*, then the probability of stepping to $d(A) - 1$ grows and the probability of stepping to $d(A) + 1$ shrinks, but this can only increase the overall probability of finding a satisfying assignment.

More importantly, we terminate the random walk after $3n$ steps, instead of allowing it to continue ad infinitum. Schöning goes through a more careful analysis than the one above, showing that this early termination reduces the probability of reaching zero by a negligable amount.

So, what is the probability that a single iteration of the outer loop reaches a satisfying assignment? After we have selected our initial assignment A, the probability is (modulo the inaccuracies noted above) $p(d(A))$, so the overall probability is just the average of this quantity over all possible sets. Any symmetric difference $A \oplus A^*$ is equally likely, so we get the formula

$$\sum_{i=0}^{n} \frac{1}{2^n} \binom{n}{i} \frac{1}{(k-1)^i} = (2 - \frac{2}{k})^{-n}.$$

For instance, for 3-SAT, we get probability $(3/4)^n$ of finding a satisfying assignment in a single iteration, so the number of iterations we need overall is roughly $(4/3)^n$.

More generally we can apply this technique to any (k,d)-CSP problem, by finding unsatisfied constraints and finding a random new value for a random variable in the constraint. We get probability $1/(d-1)k$ of reducing the Hamming distance by this random choice, and the same analysis above shows that we need roughly $(d(1-1/k))^n$ iterations to find a satisfying assignment.

Theorem 1 *Schöning's algorithm finds a solution to any satisfiable (k,d)-CSP problem, with high probability, in time $(d(1-1/k))^n n^{\mathcal{O}(1)}$.*

Although better algorithms are known for $k=2$, this approach provides the best known algorithms for k-SAT ($d=2$), as well as for other CSP problems where both k and d are greater than two.

It would be of interest to find problems other than CSP for which this random-walk approach is useful. Another direction of research is derandomization – the best known deterministic algorithm for k-SAT can be viewed as a derandomized version of Schöning's algorithm [1] but its time bounds are not as good.

References

[1] E. Dantsin, A. Goerdt, E. A. Hirsch, and U. Schöning. Deterministic algorithms for k-SAT based on covering codes and local search. *Proc. 27th Int. Coll. Automata, Languages and Programming*, 2000. http://logic.pdmi.ras.ru/∼hirsch/abstracts/icalp00.html.

[2] U. Schöning. A probabliistic algorithm for k-SAT and constraint satisfaction problems. *Proc. 40th Symp. Foundations of Computer Science*, IEEE, October 1999, pp. 410–414.