ICS 280e, Fall 2000
Lecture Notes for October 26, 2000

Feder and Motwani's randomized $(k, 2)$-CSP algorithm

David Eppstein, ICS, UC Irvine

Problem

We want to solve $(k, 2)$-CSP; that is, we are given a set of n variables x_i ($1 \leq i \leq n$), each of which can be given one of k values. We are also given a set of constraints; each constraint specifies a pair of values from different variables. The goal is to find an assignment of values to all the variables, such that no constraint has both its specified values used.

Tomás Feder and Rajeev Motwani of Stanford Univ., in an unpublished manuscript dated September 1998, gave a very simple randomized algorithm for this problem. Its analysis, however, is not so simple. Their algorithm is still the best known when k is sufficiently large $(k \geq 11)$.

It is open, and would be of interest, to find a way of derandomizing their algorithm, giving a deterministic algorithm with a similar running time. The set cover based methods we've seen for derandomizing other randomized generate-and-test algorithms don't seem to work here.

Algorithm

Like the simple randomized $(3, 2)$-CSP algorithm we saw last time, this algorithm combines ideas from both generate-and-test and backtracking algorithms. The outer loop is like a generate-and-test algorithm, repeating an inner loop enough times to make up for the low probability of finding an answer. The inner loop is similar to a recursive backtracking search, but only follows a single randomly chosen path through the search tree.

Repeat K times:

1. Choose a random permutation of the variables
2. For each variable (in the random order) choose randomly a value consistent with the previous choices

Analysis

All logs in this section should be assumed to be base 2.

Lemma 1 *Let a and b be nonnegative integers. Then* $\log(a + b - 1) \leq \log(a) + \log(b)$.

Proof: Taking powers of two on both sides, the inequality becomes $a + b - 1 \leq ab$. If a or b is one, both sides are equal; otherwise, assume without loss of generality that $a > b \geq 2$. Then the left side is less than $2a$ while the right side is at least $2a$. $\square$

Given a solution $\mathbf{x}$ to the given CSP problem, let $\text{poss}(i, \mathbf{x})$ denote the number of different values which we could assign to variable x_i (leaving the other variables fixed) and still come up with a correct solution. Also define

$$\text{value}(\mathbf{x}) = \prod_i \frac{1}{\text{poss}(i, \mathbf{x})}$$

so in some sense $\text{value}(\mathbf{x})$ is a measure of how isolated the solution is – a single isolated solution will have $\text{value}(\mathbf{x}) = 1$ while a solution in which most of the variables are free to change will have a very small value of $\text{value}(\mathbf{x})$. If x is not a valid solution, we define $\text{value}(\mathbf{x})$ to be zero.

Lemma 2

$$\sum_x \text{value}(\mathbf{x}) \geq 1.$$

Proof: Consider the search tree used in the algorithm, and prune the tree so that the only remaining leaves are the actual solutions to the CSP.

Consider assigning weights to nodes in the tree as follows. Initially, we assign unit weight to the root and zero weight elsewhere. Then, whenever there is a nonzero weight at a non-leaf node, we divide that weight equally among the node's children. Thus, the weight all ends up at the leaves, and the total weight remains equal to one.

At any leaf, the assigned weight is $\prod 1/d_i$, where d_i is the degree of the ancestor corresponding to variable x_i. But $d_i \geq \text{poss}(i, \mathbf{x})$ so the leaf's weight is at least $\text{value}(\mathbf{x})$. Thus $\sum \text{value}(\mathbf{x}) \geq \sum \prod 1/d_i = 1$. □

Lemma 3 *Let $F(\mathbf{x})$ denote the expected value of $-\log P(\mathbf{x})$, where $P(\mathbf{x})$ is the probability that CSP solution $\mathbf{x}$ is found by an iteration of the algorithm, and the expectation is over the choice of random permutations. Then*

$$F(\mathbf{x}) \leq \frac{n \log(k!)}{k} + \log(\text{value}(\mathbf{x})).$$

Proof:

$$\begin{aligned}
F(\mathbf{x}) &= -\text{E}(\log(\Pr(\text{correct value chosen for each } x_i))) \\
&= -\text{E}(\log(\prod_i \Pr(\text{correct value chosen for } x_i))) \\
&= -\text{E}(\sum_i (\log(\Pr(\text{correct value chosen for } x_i))) \\
&= -\sum_i (\text{E}(\log(\Pr(\text{correct value chosen for } x_i))) \\
&= -\sum_i (\text{E}(\log(\frac{1}{\text{number of choices for } x_i}))) \\
&\leq \sum_i \frac{1}{k+1-\text{poss}(i,\mathbf{x})} \sum_{j=\text{poss}(i,\mathbf{x})}^{k} \log(j) \\
&\leq \sum_i \frac{1}{k} \sum_{j=1}^{k} \log(j-1+\text{poss}(i,\mathbf{x})) \\
&\leq \sum_i \frac{1}{k} \sum_{j=1}^{k} (\log(j) + \log(\text{poss}(i,\mathbf{x}))) \\
&= \frac{n \log(k!)}{k} + \log(\text{value}(\mathbf{x})).
\end{aligned}$$

Here the first inequality comes from the worst case when all the values not in $\mathrm{poss}(i, \mathbf{x})$ are eliminated by single constraints to distinct variables. A value eliminated by multiple constraints, or two values eliminated by the same variable, only decrease the expectation. The second inequality comes from an averaging argument (we are taking a weighted average of terms like $\log(j)$, and throwing in some extra larger terms). The third inequality is Lemma 1. $\Box$

Lemma 4 *Let X be a nonnegative random variable. Then the probability that $X \leq \mathrm{E}[X] + 1$ is at least $1/(\mathrm{E}[X] + 1)$.*

Proof: Suppose to the contrary that $X > \mathrm{E}[X] + 1$ with probability at least $1 - 1/(\mathrm{E}[X] + 1)$. Then

$$\mathrm{E}[X] > (1 - \frac{1}{\mathrm{E}[X] + 1})(\mathrm{E}[X] + 1) = \mathrm{E}[X],$$

a contradiction. $\Box$

Theorem 1 *If there is a solution to a CSP problem, then with constant probability such a solution will be found within $\mathcal{O}(k!^{n/k}\, n \log k)$ iterations of the given algorithm*

Proof: We have seen that

$$-\mathrm{E}[\log P(\mathbf{x})] \leq \frac{n \log(k!)}{k} + \log(\mathrm{value}(\mathbf{x})).$$

Also note that $-\mathrm{E}[\log P(\mathbf{x})] \leq \max - \log P(\mathbf{x}) \leq n \log k$, so, by Lemma 4, the probability of choosing a permutation for which $\log P(\mathbf{x})$ is within one of its expectation is at least $1/(n \log k + 1)$. Putting these two facts together and exponentiating, we find that the probability of reaching solution $\mathbf{x}$ on any particular iteration is at least $k!^{-n/k} \mathrm{value}(\mathbf{x})/(n \log k + 1)$. Summing over all $\mathbf{x}$, and using the fact that $\sum \mathrm{value}(\mathbf{x}) \geq 1$, gives probability at least $k!^{-n/k}/(n \log k + 1)$ of reaching any solution in a given iteration. If the number of iterations is inverse to this probability, we will have a constant probability of seeing a solution in some iteration. $\Box$

The same analysis technique can be used to show, more generally, that if variable x_i has k_i possible values, then with constant probability the algorithm finds a solution within $\mathcal{O}(\prod_i k_i!^{1/k_i} \sum_i \log k_i)$ iterations.

References

[1] T. Feder and R. Motwani. Worst-case time bounds for coloring and satisfiability problems. Manuscript, September 1998.