16. A randomized Algorithm for Linear Programming II

Lecture on Thursday 19th November, 2009 by Bernd Gärtner <gaertner@inf.ethz.ch>

Now we get to the analysis of algorithm LP from the last lecture, and this will also reveal why the random choice is important.

We will analyze the algorithm in terms of the expected number of *violation tests* $a_h x^* \leq b_h$, and in terms of the expected number of *basis computations* $x^*(\emptyset, R)$ that it performs. This is a good measure, since these are the dominating operations of the algorithm. Moreover, both violation test and basis computation are "cheap" operations in the sense that they can be performed in time $f(d)$ for some f.

More specifically, a violation test can be performed in time $O(d)$; the time needed for a basis computation is less clear, since it amounts to solving a small linear program itself. Let us suppose that it is done by brute-force enumeration of all vertices of the bounded polyhedron defined by the at most $3d$ (in)equalities

$$a_h x = b_h, h \in R$$

and

$$-M \leq x_i \leq M, i = 1, \ldots, d.$$

16.1 Violation Tests

Lemma 16.1 *Let* $T(n, j)$ *be the maximum expected number of violation tests performed in a call to* $\mathcal{LP}(Q, R)$ *with* $|Q| = n$ *and* $|R| = d - j$*. For all* $j = 0, \ldots, d$*,*

$$\begin{aligned} T(0, j) &= 0 \\ T(n, j) &\leq T(n-1, j) + 1 + \frac{j}{n} T(n-1, j-1), \quad n > 0. \end{aligned}$$

Note that in case of $j = 0$, we may get a negative argument on the right-hand side, but due to the factor $0/n$, this does not matter.

Proof. If $|Q| = \emptyset$, there is no violation test. Otherwise, we recursively call $\mathcal{LP}(Q \setminus \{h\}, R)$ for some h which requires at most $T(n-1, j)$ violation tests in expectation. Then there is one violation test within the call to $\mathcal{LP}(Q, R)$ itself, and depending on the outcome, we perform a second recursive call $\mathcal{LP}(Q \setminus \{h\}, R \cup \{h\})$ which requires an expected number of at most $T(n-1, j-1)$ violation tests. The crucial fact is that the probability of performing a second recursive call is at most j/n.

To see this, fix some $S \subseteq Q, |S| \leq d - |R| = j$ such that $x^*(Q, R) = x^*(S, R)$. Such a set S exists by Lemma 15.4. This means, we can remove from Q all constraints except the ones in S, without changing the solution.

If $h \notin S$, we thus have

$$x^*(Q, R) = x^*(Q \setminus \{h\}, R),$$

meaning that we have already found $x^*(Q,R)$ after the first recursive call; in particular, we will then have $a_h x^* \leq b_h$, and there is no second recursive call. Only if $h \in S$ (and this happens with probability $|S|/n \leq j/n$), there can be a second recursive call. □

The following can easily be verified by induction.

Theorem 16.2

$$T(n,j) \leq \sum_{i=0}^{j} \frac{1}{i!} j!n.$$

Since $\sum_{i=0}^{j} \frac{1}{i!} \leq \sum_{i=0}^{\infty} \frac{1}{i!} = e$, we have $T(n,j) = O(j!n)$. If $d \geq j$ is constant, this is $O(n)$.

16.2 Basis Computations

To count the basis computations, we proceed as in Lemma 16.1, except that the "1" now moves to a different place.

Lemma 16.3 *Let* $B(n,j)$ *be the maximum expected number of basis computations performed in a call to* $\mathcal{LP}(Q,R)$ *with* $|Q| = n$ *and* $|R| = d - j$. *For all* $j = 0, \ldots, d$,

$$\begin{aligned} B(0,j) &= 1 \\ B(n,j) &\leq B(n-1,j) + \frac{j}{n} B(n-1,j-1), \quad n > 0. \end{aligned}$$

Interestingly, $B(n,j)$ turns out to be much smaller than $T(n,j)$ which is good since a basic computation is much more expensive than a violation test. Here is the bound that we get.

Theorem 16.4

$$B(n,j) \leq (1 + H_n)^j = O(\log^j n),$$

where H_n *is the* n*-th Harmonic number.*

Proof. This is also a simple induction, but let's do this one since it is not entirely obvious. The statement holds for $n = 0$ with the convention that $H_0 = 0$. It also holds for $j = 0$, since Lemma 16.3 implies $B(n,0) = 1$ for all n. For $n, j > 0$, we inductively obtain

$$\begin{aligned} B(n,j) &\leq (1 + H_{n-1})^j + \frac{j}{n}(1 + H_{n-1})^{j-1} \\ &\leq \sum_{k=0}^{j} \binom{j}{k} (1 + H_{n-1})^{j-k} (\frac{1}{n})^k \\ &= (1 + H_{n-1} + \frac{1}{n})^j = (1 + H_n)^j. \end{aligned}$$

The second inequality uses the fact that the terms $(1 + H_{n-1})^j$ and $\frac{j}{n}(1 + H_{n-1})^{j-1}$ are the first two terms of the sum in the second line. □

16.3 The Overall Bound

Putting together Theorems 16.2 and 16.4 (for $j = d$, corresponding to the case $R = \emptyset$), we obtain the following

Theorem 16.5 *A linear program with* n *constraints in* d *variables (*d *a constant) can be solved in time* $O(n)$*.*