Pruning Planes in Megiddo’s 3-Dimensional LP algorithm

Geometric Optimization

April 22, 2002

We recall the situation studied in class: We have constraints of the form $z \geq a_i x + b_i + c$ or $z \leq a_i x + b_i + c$ (let’s ignore other types of constraints for now), and we want to find the minimum $z^*$ of the $z$-coordinates of points in the feasible region $K$.

We have a decision procedure that compares $z^*$ with any given value $z_0$. It does so by running a “1-dimensional LP” algorithm on a line $y = y_0$ within the plane $z = z_0$, determining whether the given halfspaces, when restricted to this line, have a nonempty solution.

Figure 1: Example

We want to consider a generic simulation of this procedure on the unknown plane $z = z^*$ and on the unknown line $y = y^*$ within this plane, which contains the lowest point of $K$. (If we assume general position, this is the only feasible point on this line and plane).

Let us assume that we have a generalization of our decision procedure that can also determines on which side of an arbitrary plane the optimum lies. (I skip here details of such a procedure—not too hard to fill in.)

Consider a halfspace, say $z \geq a_i x + b_i + c_i$. Its intersection with the line $z = z^*, y = y^*$ is the ray

$$z^* \geq a_i x + b_i y^* + c_i.$$ 

Assuming that $a_i > 0$, we get the ray

$$x \leq \frac{-b_i y^* + z^* - c_i}{a_i}.$$ 

Recall that we have to compute the maximum of the left endpoints of all rightward-directed such rays, and the minimum of the right endpoints of all leftward-directed rays. To do
Figure 2: Example

this with a parallel algorithm, we simply compare pairs of left endpoints, and pairs of right endpoints, \( n/2 \) pairs in total. Each such comparison compares two expressions of the form

\[
\frac{-b_i y^* + z^* - c_i}{a_i} : \frac{-b_j y^* + z^* - c_j}{a_j},
\]

which amounts to determining the sign of some linear expression

\[
\alpha_{ij} y^* + \beta_{ij} z^* + \gamma_{ij}.
\]

If we project this onto the \( yz \)-plane, we obtain a collection of lines \( \alpha_{ij} y + \beta_{ij} z + \gamma_{ij} = 0 \), and we need to determine the side of each of them that contains the optimum \((y^*, z^*)\).

Figure 3: Example

Here Megiddo uses the following trick.

(a) Find the median of the slopes of these lines. Rotate the \( yz \)-plane, so that half of these lines have positive slopes and half have negative slopes.

(b) Pair the lines, so that in each pair we have one line with a positive slope, and one with a negative slope. Find the intersection points of these pairs of lines. (We have \( n/4 \) points.)
Figure 4: Example

Figure 5: Example

(e) Find the median $z_m$ of the $z$-coordinates of the intersection points. Test whether $z^*$ is larger or smaller than $z_m$. Suppose, without loss of generality, that $z^* > z_m$.

(d) Find the median $y_m$ of the $y$-coordinates of those intersection points, whose $z$-coordinate lie on the other side of $z_m$ (in the current case, on the side $z < z_m$). Test whether $y^*$ is larger or smaller than $y_m$ (using the generalized decision procedure that we have assumed above). Suppose, without loss of generality, that $y^* > y_m$.

(e) Now look at the points whose $z$- and $y$-coordinates both lie in the ‘wrong’ sides; that is, $z < z_m$ and $y < y_m$. Let $p$ be such a point, the intersection of $\ell^+$ with a positive slope and of $\ell^-$ with a negative slope. Observe that we know which side of $\ell^-$ contains the optimum (it is the top-right side). Since $\ell^-$ itself was a line with the property that for each point $(y^*, z^*)$ on it, the values of two endpoints of two specific rays on the line $y = y^*$ in the plane $z = z^*$ coincide, it follows that we now know that one of these endpoints must lie to the left of the other at the optimum value $(y^*, z^*)$, so we can delete one of the rays without affecting the result of the generic decision procedure.

Note that, after step (c) we have $n/8$ points and after step (d) we have $n/16$ points. In step (e), for each of these remaining points, we throw away one halfspace. So we managed to delete $n/16$ constraints, and we can stop the whole process and restart it from scratch with the remaining $15n/16$ constraints.
Altogether, everything can be implemented in linear time, so the whole algorithm also takes linear time (as we argued for the 2-dimensional case in the basic course).