18.415/6.854 Advanced Algorithms September 24, 2001

Lecture 4

Lecturer: Michel X. Goemans *Scribe: Reina Riemann*

1 LP Duality

Last time, we proved the strong duality theorem. Let P and D be the following pair of dual linear programs:

$$\begin{array}{ll} (P) & z^* = \min\{c^T x : Ax = b, x \geq 0\}, \\ (D) & w^* = \max\{b^T y : A^T y \leq c\}. \end{array}$$

Theorem 1 (Strong duality) *If P or D is feasible then $z^* = w^*$.*

1.1 Rules for Taking Dual Problems

We can also express the strong duality theorem for linear programs in other forms. This is described in the main lecture notes. For example, if the primal linear program is:
$z = \min\ c_1^T x_1 + c_2^T x_2 + c_3^T x_3$ s.t.

$$\begin{array}{rcl} A_{11}x_1 + A_{12}x_2 + A_{13}x_3 & = & b_1 \\ A_{21}x_1 + A_{22}x_2 + A_{23}x_3 & \geq & b_2 \\ A_{31}x_1 + A_{32}x_2 + A_{33}x_3 & \leq & b_3 \\ x_1 \geq 0\ ,\ x_2 \leq 0\ ,\ x_3 \text{ UIS}, & & \end{array}$$

where A_{ij} is a mtrix of size $m_i \times n_j$ (and thus $x_j \in \mathbb{R}^{n_j}$) and UIS denotes no restrictions ("unrestricted in sign") on these variables, then the dual is:
$w = \max\ b_1^T y_1 + b_2^T y_2 + b_3^T y_3$
s.t.

$$\begin{array}{rcl} A_{11}^T y_1 + A_{21}^T y_2 + A_{31}^T y_3 & \leq & c_1 \\ A_{12}^T y_1 + A_{22}^T y_2 + A_{32}^T y_3 & \geq & c_2 \\ A_{13}^T y_1 + A_{23}^T y_2 + A_{33}^T y_3 & = & c_3 \\ y_1 \text{ UIS}\ ,\ y_2 \geq 0\ ,\ y_3 \leq 0, & & \end{array}$$

where thus $y_i \in \mathbb{R}^{m_i}$, and strong duality holds between them.

2 Size of LP

In order to be able to discuss the complexity for solving a linear program, we need first to discuss the size of the input. We assume that every integer data is given in binary encoding, thus for $n \in \mathbb{Z}$, we need

$$size(n) = 1 + \lceil \log_2(|n| + 1) \rceil$$

bits, for $v \in \mathbb{Z}^p$, we need

$$size(v) = \sum_{i=1}^{p} (v_i)$$

bits, and for $A \in \mathbb{Z}^{nxm}$, we need

$$size(A) = \sum_{i=1}^{n} \sum_{j=1}^{m} (a_{i,j}).$$

bits. As a result, to represent all the data of a linear program, we need a size equal to

$$size(LP) = size(b) + size(c) + size(A).$$

The above size is not very convenient when proving the complexity of a linear programming algorithm. Instead, we will be considering another size, defined by

$$m + n + size(det_{max}) + size(b_{max}) + size(c_{max}),$$

where $\det_{max} = \max |\det(A')|$ over all submatrices A' of A, $b_{max} = \max_i |b_i|$ and $c_{max} = \max_j |c_j|$.

In the next proposition, we show that L is smaller than $size(LP)$, which implies that if we can prove that an algorithm has a running time polynomially bounded in terms of L then it will certainly be polynomial in $size(LP)$ as well.

Proposition 2 $L <$ size(LP).

The proof of the proposition is in the lecture notes. In the proof, the following key lemma is needed.

Lemma 3 *If* $A \in \mathbb{Z}^{n \times n}$ *then* $|det(A)| \leq 2^{size(A)-n^2} - 1.$

Proof: Recall that for $A = [a_1, a_2, ..., a_k]$, $|\det(A)|$ can be visualized as the volume of the parallelipiped spanned by the column vectors. Hence,

$$1 + |det(A)| \leq 1 + \prod_{i=1}^{n} \|a_i\| \leq \prod_{i=1}^{n} (1 + \|a_i\|) \leq \prod_{i=1}^{n} 2^{size(a_i)-n} = 2^{size(A)-n^2}.$$

□

From the definition of L, the following remark follows; this is what we will need mostly when analyzing running times or sizes.

Remark 1 $det_{max} * b_{max} * c_{max} * 2^{m+n} < 2^L.$

3 Complexity of LP

Here is the decision problem corresponding to linear programming.

LP: Given A, b, c, λ, is there an $x : Ax = b, x \geq 0, c^T x \leq \lambda$?

To show that LP is in NP, we need to be able to provide a concise (i.e. polynomially bounded in the size of the input) certificate for yes instances. A feasible point of cost less or equal to λ will clearly be a certificate, but will it be concise?

Claim 4 *$LP \in NP$*

We now show that if we take not just any feasible solution, but a basic feasible solution, then its size will be polynomially bounded in the size of the input.

Theorem 5 *Let x be a vertex (or basic feasible solution) of $Ax = b, x \geq 0$. Then $x_i = \frac{p_i}{q}$. for i=1,...,n where $p_i, q \in \mathbb{N}$ and $p_i < 2^L$ and $q < 2^L$.*

Proof: Since x is a vertex, then x is a basic feasible solution with basis B such that $x_B = A_B^{-1} b$ and $x_N = 0$ (notice that A_B is square). By Cramer's rule:

$$x_B = A_B^{-1} b = \frac{1}{\det(A_B)} cof(A_B) b,$$

where $cof(A)$ is a matrix whose entries are all determinants of submatrices of A. Letting $q = \det(A_B)$, we get that $q \leq det_{\max} < 2^L$ and $p_i \leq m\, det_{\max}\, b_{\max} < 2^L$. □

Now, to prove Claim 4, for yes instances, the certificate will be a vertex of $\{x : Ax = b, x \geq 0\}$ such that $c^T x \leq \lambda$. However, to be precise, we also have to deal with the case in which the LP is unbounded, since in that case, there might not be any such vertex. But in that case, we can give a certificate of unboundedness by (i) exhibiting a vertex of $\{x : Ax = b, x \geq 0\}$ (showing it is not empty, and it is concise by the above theorem) and (ii) showing that the dual feasible region $\{y : A^T y \leq c\}$ is empty by using Farkas' lemma and exhibiting a vertex of $Ax = b, x \geq 0, c^T x = -1$ which is also concise by the above theorem.

Thanks to duality, we can also show that:

Claim 6 *$LP \in co-NP$.*

Indeed, for no instances, we can use strong duality and exhibit a basic feasible solution of $A^T y \leq c$ s.t. $b^T y > \lambda$ (or show that $\{x \geq 0 : Ax = b\}$ is empty using Farkas' lemma). In the case when $\{x : Ax = b, x \geq 0\}$ is feasible, the correctness follows from strong duality saying that

$$\min\{c^T x : Ax = b, x \geq 0\} = \max\{b^T y : A^T y \leq c\}.$$

Thus, $LP \in NP \cap co-NP$ which makes it likely to be in P. And indeed, LP was shown to be polynomially solvable through the ellipsoid algorithm.

The Ellipsoid algorithm was proposed by the Russian mathematician Shor in 1977 for general convex optimization problems, and applied to linear programming by Khachyan in 1979.

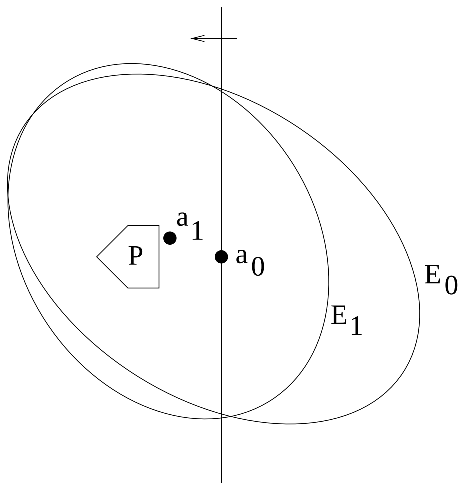

Figure 1: One iteration of the ellipsoid algorithm.

4 The Ellipsoid Algorithm

The problem being considered by the ellipsoid algorithm is:

Given a bounded convex set $P \in \mathbb{R}^n$ find $x \in P$.

We will see that we can reduce linear programming to finding an x in $P = \{x \in \mathbb{R}^n : Cx \leq d\}$.

The ellipsoid algorithm works as follows. We start with a big ellipsoid E that is guaranteed to contain P. We then check if the center of the ellipsoid is in P. If it is, we are done, we found a point in P. Otherwise, we find an inequality $c^T x \leq d_i$ which is satisfied by all points in P (for example, it is explicitly given in the description of P) which is not satisfied by our center. One iteration of the ellipsoid algorithm is illustrated in Figure 1. The ellipsoid algorithm is the following.

- Let E_0 be an ellipsoid containing P
- while center a_k of E_k is not in P do:
  - Let $c^T x \leq c^T a_k$ be such that $\{x : c^T x \leq c^T a_k\} \supseteq P$
  - Let E_{k+1} be the minimum volume ellipsoid containing $E_k \cap \{x : c^T x \leq c^T a_k\}$
  - $k \leftarrow k + 1$

The ellipsoid algorithm has the important property that the ellipsoids contructed shrink in volume as the algorithm proceeds; this is stated precisely in the next lemma. This means that if the set P has positive volume, we will eventually find a point in P. We will need to deal with the case when P has no volume (i.e. P has just a single point), and also discuss when we can stop and be guaranteed that either we have a point in P or we know that P is empty.

Lemma 7 $\frac{Vol(E_{k+1})}{Vol(E_k)} < e^{-\frac{1}{2n}}$.

Before we can state the algorithm more precisely, we need to define ellipsoids.

Definition 1 *Given a center a, and a positive definite matrix A, the ellipsoid $E(a, A)$ is defined as $\{x \in \mathbb{R}^n : (x-a)^T A^{-1}(x-a) \leq 1\}$.*

One important fact about a positive definite matrix A is that there exists B such that $A = B^{-1}B$, and hence $A^{-1} = B^{-1}(B^{-1})^T$. Ellipsoids are in fact just affine tansformations of unit spheres. To see this, consider the (bijective) affine transformation $T : x \to y = (B^{-1})^T(x-a)$. It maps $E(a, A) \to \{y : y^T y \leq 1\} = E(0, I)$.

We first consider the simple case in which the ellipsoid E_k is the unit sphere and that the inequality we generate is $x_1 \geq 0$. We claim that the ellipsoid containing $E_k \cap \{x : x_1 \geq 0\}$ is

$$E_{k+1} = \left\{ x : \left(\frac{n+1}{n}\right)^2 \left(x - \frac{1}{n+1}\right)^2 + \frac{n^2-1}{n^2} \sum_{i=2}^{n} x_i^2 \leq 1 \right\}$$

Indeed, if we consider an $x \in E_k \cap \{x : x_1 \geq 0\}$, we see that

$$\begin{aligned}
&\left(\frac{n+1}{n}\right)^2 \left(x - \frac{1}{n+1}\right)^2 + \frac{n^2-1}{n^2} \sum_{i=2}^{n} x_i^2 \\
= \;& \frac{n^2+2n+1}{n^2} x_1^2 - \left(\frac{n+1}{n}\right)^2 \frac{2x_1}{n+1} + \frac{1}{n^2} + \frac{n^2-1}{n^2} \sum_{i=2}^{n} x_i^2 \\
= \;& \frac{2n+2}{n^2} x_1^2 - \frac{2n+2}{n^2} x_1 + \frac{1}{n^2} + \frac{n^2-1}{n^2} \sum_{i=1}^{n} x_i^2 \\
= \;& \frac{2n+2}{n^2} x_1(x_1 - 1) + \frac{1}{n^2} + \frac{n^2-1}{n^2} \sum_{i=1}^{n} x_i^2 \\
\leq \;& \frac{1}{n^2} + \frac{n^2-1}{n^2} \leq 1.
\end{aligned}$$

In this specific case, we can prove easily lemma 7 (we prove a slight weakening of it by just showing an upper bound of $e^{-1/2(n+1)}$):

Proof: The volume of an ellipsoid is proportional to the product of its side lengths. Hence the ratio between the unit ellipsoid E_k and E_{k+1} is

$$\begin{aligned}
\frac{Vol(E_{k+1})}{Vol E_k} &= \frac{(\frac{n}{n+1})(\frac{n^2}{n^2-1})^{\frac{n-1}{2}}}{1} \\
&= \left(\frac{n}{n+1}\right)\left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}} \\
&\leq e^{-\frac{1}{n+1}} e^{\frac{n-1}{(n^2-1)2}} = e^{-\frac{1}{n+1}} e^{\frac{1}{2(n+1)}} = e^{-\frac{1}{2(n+1)}},
\end{aligned}$$

where we have used the fact that $1 + x \leq e^x$ for all x. □