Massachusetts Institute of Technology | Handout 19
18.433: Combinatorial Optimization | December 2nd, 2005
Michel X. Goemans

Lecture notes on the ellipsoid algorithm

The simplex algorithm was the first algorithm proposed for linear programming, and although the algorithm is quite fast in practice, no variant of it is known to be polynomial time. The Ellipsoid algorithm is the first polynomial-time algorithm discovered for linear programming. 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. Contrary to the simplex algorithm, the ellipsoid algorithm is not very fast in practice; however, its theoretical polynomiality has important consequences for combinatorial optimization problems as we will see.

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

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

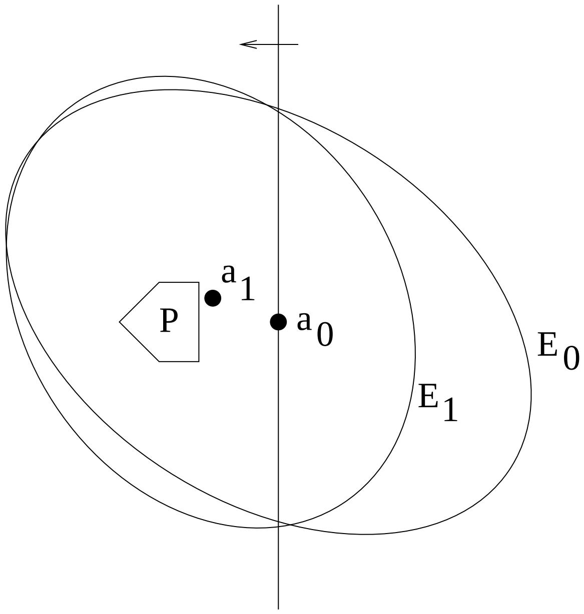

Figure 1: One iteration of the ellipsoid algorithm.

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

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

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

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

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 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. △

Now consider the slightly more general case in which the ellipsoid is also the unit sphere but we have an arbitrary constraint $d^T x \leq 0$. We want to find an ellipsoid that contains $E(0, I) \cap \{x : d^T x \leq 0\}$ (we let $\|d\| = 1$; this can be done by scaling both sides), it is easy to verify that we can take $E_{k+1} = E(-\frac{1}{n+1}d, F)$, where $F = \frac{n^2}{n^2-1}(I - \frac{2}{n+1}dd^T)$, and the ratio of the volumes is $\leq \exp\left(-\frac{1}{2(n+1)}\right)$.

Now we deal with the case where E_k is not the unit sphere. We take advantage of the fact that linear transformations preserve ratios of volumes.

$$\begin{array}{ccc}
E_k & \xrightarrow{T} & E(0,1) \\
 & & \downarrow \\
E_{k+1} & \xleftarrow{T^{-1}} & E'
\end{array} \tag{1}$$

Let a_k be the center of E_k, and $c^T x \leq c^T a_k$ be the halfspace through a_k that contains P. Therefore, the half-ellipsoid that we are trying to contain is $E(a_k, A) \cap \{x : c^T x \leq c^T a_k\}$.

Let's see what happens to this half-ellipsoid after the transformation T defined by $y = T(x) = (B^{-1})^T(x-a)$. This transformation transforms $E_k = E(a_k, A)$ to $E(0, I)$. Also,

$$\{x : c^T x \leq c^T a_k\} \xrightarrow{T} \{y : c^T(a_k + B^T y) \leq c^T a_k\} = \{y : c^T B^T y \leq 0\} = \{y : d^T x \leq 0\}, \quad (2)$$

where d is given by the following equation.

$$d = \frac{Bc}{\sqrt{c^T B^T Bc}} = \frac{Bc}{\sqrt{c^T Ac}} \quad (3)$$

Let $b = B^T d = \frac{Ac}{\sqrt{c^T Ac}}$. This implies:

$$\begin{aligned} E_{k+1} &= E\left(a_k - \frac{1}{n+1}b, \frac{n^2}{n^2-1}B^T\left(I - \frac{2}{n+1}dd^T\right)B\right) && (4)\\ &= E\left(a_k - \frac{1}{n+1}b, \frac{n^2}{n^2-1}\left(A - \frac{2}{n+1}bb^T\right)\right) && (5) \end{aligned}$$

To summarize, here is the Ellipsoid Algorithm:

1. Start with $k = 0$, $E_0 = E(a_0, A_0) \supseteq P$, $P = \{x : Cx \leq d\}$.
2. While $a_k \notin P$ do:
   - Let $c^T x \leq d$ be an inequality that is valid for all $x \in P$ but $c^T a_k > d$.
   - Let $b = \frac{A_k c}{\sqrt{c^T A_k c}}$.
   - Let $a_{k+1} = a_k - \frac{1}{n+1}b$.
   - Let $A_{k+1} = \frac{n^2}{n^2-1}(A_k - \frac{2}{n+1}bb^T)$.

Claim 2 $\frac{Vol(E_{k+1})}{Vol(E_k)} \leq \exp\left(-\frac{1}{2(n+1)}\right)$.

After k iterations, $Vol(E_k) \leq Vol(E_0)\exp\left(-\frac{k}{2(n+1)}\right)$. If P is nonempty then the Ellipsoid Algorithm should find $x \in P$ in at most $2(n+1)\ln\frac{Vol(E_0)}{Vol(P)}$ steps.

In general, for a full-dimensional polyhedron described as $P = \{x : Cx \leq d\}$, one can show that $\frac{Vol(E_0)}{Vol(P)}$ is polynomial in the encoding length for C and d. We will not show this in general, but we will focus on the most important situation in combinatorial optimization when we are given a set $S \subseteq \{0,1\}^n$ (not explicitly, but for example as the incidence vectors of all matchings in a graph) and we would like to optimize over $P = conv(S)$. We will make the assumption that P is full-dimensional; otherwise, one can eliminate one or several variables and obtain a smaller full-dimensional problem.

From feasibility to Optimization First, let us show how to reduce such an optimization problem to the problem of finding a feasible point in a polytope. Let $c^T x$ with $c \in \mathbb{R}^n$ be our objective function we would like to minimize over P. Assume without loss of generality that $c \in \mathbb{Z}^n$. Instead of optimizing, we can check the non-emptiness of

$$P' = P \cap \{x : c^T x \leq d + \frac{1}{2}\}$$

for $d \in \mathbb{Z}$ and our optimum value corresponds to the smallest such d. As $S \subseteq \{0,1\}^n$, d must range in $[-nc_{max}, nc_{max}]$ where $c_{max} = \max_i c_i$. To find d, we can use binary search (and check the non-emptiness of P' with the ellipsoid algorithm). This will take $O(\log(nc_{max})) = O(\log n + \log c_{max})$ steps, which is polynomial.

Starting Ellipsoid. Now, we need to consider using the ellipsoid to find a feasible point in P' or decide that P' is empty. As starting ellipsoid, we can use the ball centered at the vector $(\frac{1}{2}, \frac{1}{2}, \cdots, \frac{1}{2})$ and of radius $\frac{1}{2}\sqrt{n}$ (which goes through all $\{0,1\}^n$ vectors). This ball has volume $Vol(E_0) = \frac{1}{2^n}(\sqrt{n})^n Vol(B_n)$, where B_n is the unit ball. We have that $Vol(B_n) = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)}$, which for the purpose here we can even use the (very weak) upper bound of $\pi^{n/2}$. This shows that $\log(Vol(E_0)) = O(n \log n)$.

Termination Criterion. We will now argue that if P' is non-empty, its volume is not too small. Assume that P' is non-empty, say $v_0 \in P' \cap \{0,1\}^n$. Our assumption that P is full-dimensional implies that there exists $v_1, v_2, \cdots, v_n \in P \cap \{0,1\}^n = S$ such that the "simplex" $v_0, v_1, \cdots, v_n$ is full-dimensional. The v_i's may not be in P'. Instead, define w_i for $i = 1, \cdots, n$ by:

$$w_i = \begin{cases} v_i & \text{if } c^T v_i \leq d + \frac{1}{2} \\ v_0 + \alpha(v_i - v_0) & \text{otherwise} \end{cases}$$

where $\alpha = \frac{1}{2nc_{max}}$. This implies that $w_i \in P'$ as

$$c^T w_i = c^T v_0 + \alpha c^T (v_i - v_0) \leq d + \frac{1}{2nc_{max}} nc_{max} = d + \frac{1}{2}.$$

We have that P' contains $C = conv(\{v_0, w_1, w_2, \cdots, w_n\})$ and $Vol(C)$ is $\frac{1}{n!}$ times the volume of the parallelipiped spanned by $w_i - v_0 = \beta_i(v_i - v_0)$ (with $\beta_i \in \{\alpha, 1\}$) for $i = 1, \cdots, n$. This paralleliped has volume equal to the product of the β_i (which is at least α^n) times the volume of a parallelipiped with integer vertices, which is at least 1. Thus,

$$Vol(P') \geq Vol(C) = \frac{1}{n!}\left(\frac{1}{2nc_{max}}\right)^n.$$

Taking logs, we see that the number of iterations of the ellipsoid algorithm before either discovering that P' is empty or a feasible point is at most

$$\log(Vol(E_0)) - \log(Vol(P')) = O(n \log n + n \log c_{max}).$$

This is polynomial.

Separation Oracle. To run the ellipsoid algorithm, we need to be able to decide, given $x \in \mathbb{R}^n$, whether $x \in P'$ or find a violated inequality. The beauty here is that we do not necessarily need a complete and explicit description of P in terms of linear inequalities. We will see examples in which we can even apply this to exponential-sized descriptions. What we need is a *separation oracle* for P: Given $x^* \in \mathbb{R}^n$, either decide that $x^* \in P$ or find an inequality $a^T x \leq b$ valid for P such that $a^T x^* > b$. If this separation oracle runs in polynomial-time, we have succeeded in finding the optimum value d when optimizing $c^T x$ over P (or S).

Finding an optimum solution. There is one more issue. This algorithm gives us a point x^* in P of value at most $d + \frac{1}{2}$ where d is the optimum value. However, we are interested in finding a point $x \in P \cap \{0,1\}^n = S$ of value exactly d. This can be done by starting from x^* and finding *any extreme* point x of P such that $c^T x \leq c^T x^*$. Details are omitted.

In summary, we obtain the following important theorem shown by Grötschel, Lovász and Schrijver, 1979.

Theorem 3 *Let $S = \{0,1\}^n$ and $P = conv(S)$. Assume that P is full-dimensional and we are given a separation oracle for P. Then, given $c \in \mathbb{Z}^n$, one can find* $\min\{c^T x : x \in S\}$ *by the ellipsoid algorithm by using a polynomial number of operations and calls to the separation oracle.*

Here are two brief descriptions of combinatorial optimization problems that can be solved with this approach.

Minimum Cost Arborescence Problem. We have seen a combinatorial algorithm to solve the minimum cost arborescence problem, and it relied on solving by a primal-dual method the linear program:

$$LP = \min \sum_{a \in A} c_a x_a$$

subject to:

$$(P) \qquad \sum_{a \in \delta^-(S)} x_a \geq 1 \qquad \forall S \subseteq V \setminus \{r\}$$

$$x_a \geq 0 \qquad a \in A.$$

Instead, we could use the ellipsoid algorithm to directly solve this linear program in polynomial-time. Indeed, even though this linear program has an exponential number of constraints (in the size of the graph), a separation oracle for it can be easily defined. Indeed consider x^*. If $x^*_a < 0$ for some $a \in A$, just return the inequality $x_a \geq 0$. Otherwise, for every $t \in V \setminus \{r\}$, consider the minimum $r - t$ cut problem in which the capacity on arc a is given by x^*_a. As we have seen, this can be solved by maximum flow computations. If for some $t \in V \setminus \{r\}$, the minimum $r - t$ cut has value less than 1 then we have found a violated inequality by x^*. Otherwise, we have that $x^* \in P$. We are done.

Maximum Weight Matching Problem. For the maximum weight matching problem in a general graph $G = (V, E)$, Edmonds has shown that the convex hull of all matchings is given by:

$$\begin{aligned} \sum_{e \in E(S)} x_e &\leq \frac{|S|-1}{2} & & S \subset V, |S| \text{ odd} \\ \sum_{e \in \delta(v)} x_e &\leq 1 & & v \in V \\ x_e &\geq 0 & & e \in E. \end{aligned} \tag{6}$$

Given x^*, we can easily check if x^* is nonnegative and if the n constraints $\sum_{e \in \delta(v)} x_e \leq 1$ are satisfied. There is an exponential number of remaining constraints, but one can show that they can be checked by solving a minimum cut computation.