CSC2411 - Linear Programming and Combinatorial Optimization*
Lecture 8: Ellipsoid Algorithm

Notes taken by Shizhong Li

March 15, 2005

Summary: In the spring of 1979, the Soviet mathematician L.G.Khachian discovered a polynomial algorithm for LP called *Ellipsoid Algorithm*. This discovery classifies the linear programming problem in class P for the first time. After the high level introduction of the algorithm in the last class, we turn to discuss it in details in this lecture. *Ellipsoid Algorithm*.

1 Sketch of the Ellipsoid Algorithm

1.1 Pseudocode of Ellipsoid Algorithm

Algorithm 1.1 (Ellipsoid Algorithm).

input: A, b
output: $x \in P$ or "P is empty" where $P = \{x \in R^n | Ax < b\}$
init: $k = 0$ /* iteration counter */
  $R = n2^L$ /* initial radius to look at */
  $N = 16n(n+1)L$ /* maximum number of iterations needs to perform */
  $E_0 = B(0, R)$ /* initial search space */
  if $k = N$ **then**
   announce "P is empty"
  if $c_k =$center(E_k) satisfies $Ac_k < b$ **then**
   output x
  else
   find inequality $(a_i, x) < b_i$ that is violated by c_k, so $(a_i, c_k) \geq b_i$ while for $(a_i, x) < b_i, \forall x \in P$
   find E_{k+1} with the following properties
    (i) $E_{k+1} \supset E_k \cap \{x | (a_i, x) < b\}$
    (ii) Vol$(E_{k+1}) \leq e^{-\frac{1}{2(n+1)}} \cdot$ Vol(E_k)
  endif

* Lecture Notes for a course given by Avner Magen, Dept. of Computer Sciecne, University of Toronto.

1.2 Proof of the correctness

Proof. if the alogrithm the stops and finds a feasible solution, this is clearly fine.
If the algorithm stops and does not find a feasible solution, we have to show that $P = \emptyset$.
Notice that in the algorithm, Vol$(E_{k+1}) \leq e^{-\frac{1}{2(n+1)}} \cdot$ Vol(E_k), we have Vol$(E_N) \leq e^{-\frac{N}{2(n+1)}} \cdot$ Vol(E_0)
Assume P is not empty after N iterations. We know from last class, Vol$(P) \geq 2^{-O(nL)}$ and now Vol$(E_0) \leq (2R)^n = (2n)^n \cdot 2^{nL}$.
Thus Vol$(E_N) \leq$ Vol$(E_0) \cdot e^{-\frac{N}{2(n+1)}} \leq (2n)^n \cdot 2^{nL} \cdot e^{-\frac{N}{2(n+1)}}$.
For a large enough constant k, $N = kn^2L$, Vol$(E_N) \leq e^{n^2+n+nL-\frac{kn^2L}{2(n+1)}} \leq e^{-O(nL)} =$ Vol(P).
However, $P \subset E_N$, it is a contradiction. $\square$

2 How to find the next Ellipsoid

Definition 2.1. Ellipsoid is the image under an offine map of a unit ball $B(0,1)$ in R^n.

$$\begin{aligned} E &= T(B) \\ &= \{T(x) | x \in B\} \\ &= \{Ax + c | ||x|| \leq 1\} \\ &= \{y | ||A^{-1}(y-c)|| \leq 1\} \\ &= \{y | (y-c)^t (A^{-1})^t A^{-1} (y-c) \leq 1\} \\ &= \{y | (y-c)^t Q^{-1} (y-c) \leq 1\} \text{ where } Q = AA^t \end{aligned}$$

Q is $n \times n$ symmetric matrix which is *Positive Definite*, that is: $\forall x \in R^n$ and $x \neq 0, x^tQx > 0$

For example, $B(0,1)$ is the ellipsoid with $Q = I$, $B(0,r)$ is with $Q = r^2I$.

Theorem 2.2. *For an ellipsoid B and E where $E = T(B)$, T is an affine transformation, then Vol$(E) =$ Vol$(T(B)) = \sqrt{det(Q)} \cdot$ Vol(B).*

Theorem 2.3. *(Löwner John): Let $K \in R^n$ be convex, then there is a (unique) ellipsoid E containing K and of minimal volume, further $\frac{1}{n}E \subset K \subset E$.*

See Figure[**??**],K is called *LJ ellipsoid*. In our case, K is half-ellipsoid, and we will show how to find the LJ ellipsoid of K.

We use linear transformation to transform the ellipsoids in Figure[**??**] to Figure[**??**]. Notice that we transform the original ellipsoid E to a unit ball $B(0,1)$, $\frac{1}{2}E$ is tranformed to the northern halfball G, and the minimal volume ellipsoid E' is transformed to F.

First we need to prove we do not change the ratio of the volume between E' and E by the transformation, that is $\frac{\text{Vol}(F)}{\text{Vol}(B)} = \frac{\text{Vol}(E')}{\text{Vol}(E)}$.

Proof. Let T be the mapping, that is $E = T(B)$ and $\frac{1}{2}E = T(G)$, thus $B = T^{-1}(E)$, $G = T^{-1}(\frac{1}{2}E)$, and the half space of the intersect is $\{x \geq 0\}$. Suppose F is the

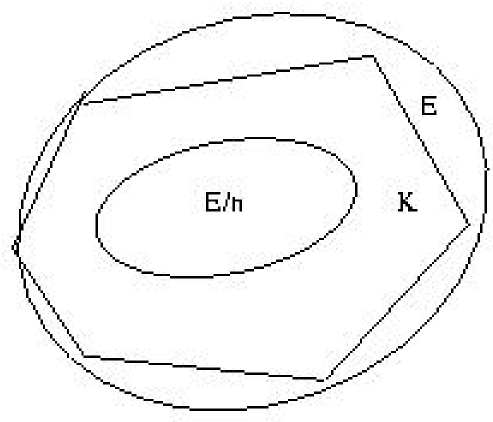

Figure 1: $K \in R^n$ is a convex, E is the ellipsoid containing K with the minimal volume.

ellipsoid we found, then $F = T^{-1}(E')$. By Theorem **??**, we have $\text{Vol}(F) = \text{Vol}(E') \cdot \det(T^{-1})$ and $\text{Vol}(B) = \text{Vol}(E) \cdot \det(T^{-1})$, then we have $\frac{\text{Vol}(F)}{\text{Vol}(B)} = \frac{\text{Vol}(E') \cdot \det(T^{-1})}{\text{Vol}(E) \cdot \det(T^{-1})} = \frac{\text{Vol}(E')}{\text{Vol}(E)}$. $\square$

How to find the new ellipsoid E in Figure[**??**]? Recall Definition [**??**], we have $E = \{|(x-c)^t Q^{-1}(x-c) \leq 1\}$, where $c = (0, 0, ..., 0, t)$ is the center of E. We transform E from the unit ball $B(0,1)$, thus by Theorem **??**, we have $\text{Vol}(E) = \sqrt{\det(Q)} \cdot \text{Vol}(B(0,1))$. We require $\frac{1}{2}B$ to be contained in E.
We notice that we can get E by shrinking the unit ball $B(0,1)$ in vertical direction and shift it upward, stretching $B(0,1)$ symmetrically orthogonal to the vertical direction. We also notice that E is symmetic in all directions, that means Q is a diagonal matrix. Thus we can write Q as:

$$Q = \begin{pmatrix} \alpha & & & \\ & \cdots & & \\ & & \alpha & \\ & & & \beta \end{pmatrix}$$

where last row corresponds to the vertical direction.

Now we can set our goals in formula:
Goal: find α, β, c so that $\frac{1}{2}B \subset E$ and $\text{Vol}(E)$ must be minimized, that is $\det(Q)$ is minimized (by Theorem **??**). We have the following equations:

$\min(\alpha^{n-1}\beta)$	(by $\min(\det(Q))$, since $\text{Vol}(E) = \sqrt{\det(Q)} \cdot \text{Vol}(B(0,1))$)
$(1-t)^2\beta^{-1} = 1$	(for $x = (0, 0, .., 0, 1), (x-c)^t Q^{-1}(x-c) = 1$)
$\alpha^{-1}\lvert\vec{x_1}\rvert^2 + \beta^{-1}c^2 = 1$	(for $x = (\vec{x_1}, 0)$ and $\lvert\vec{x_1}\rvert^2 = 1, (x-c)^t Q^{-1}(x-c) = 1$)

From $(1-t)^2\beta^{-1} = 1$, we can get $\beta = (1-t)^2$. From $\alpha^{-1}|\vec{x_1}|^2 + \beta^{-1}c^2 = 1$

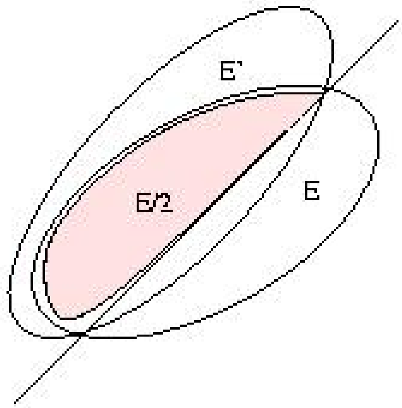

Figure 2: Original ellipsoid E, we need a new ellipsoid E' to contain $\frac{1}{2}E$ with minimal volume.

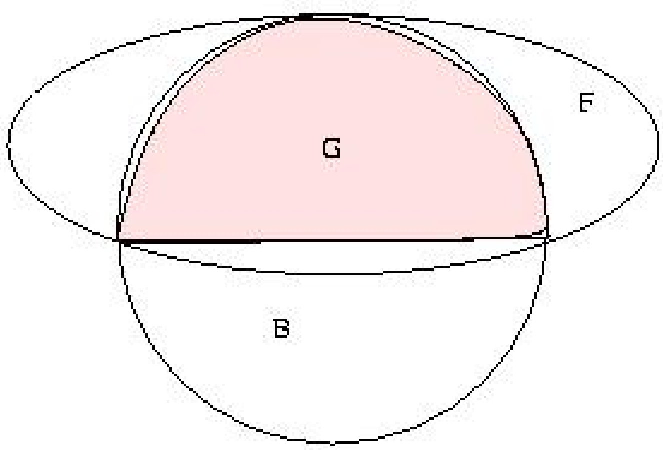

Figure 3: We use linear transformation to transform E to a unit ball B, now $\frac{1}{2}E$ is the northern half-ball G, F is transformed from E'.

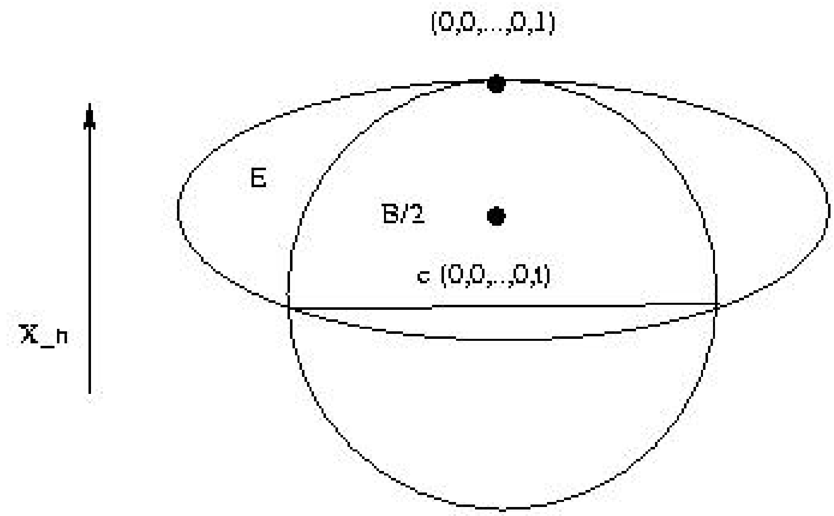

Figure 4: We need to find an ellipsoid E to contain $\frac{1}{2}B$ which is the northern half ball. E has the minimal volume, and c is the center of E.

and $|\vec{x_1}|^2 = 1$, we get $\frac{1}{\alpha} + \frac{t^2}{\beta} = 1$. Then we substitute $\beta = (1-t)^2$, we can get $\alpha = \frac{(1-t)^2}{1-2t}$.
We denote $F = \alpha^{n-1}\beta$, substitute α, β, then $F = (\frac{(1-t)^2}{1-2t})^{n-1}(1-t)^2$. To get $\min(F)$, we force $\frac{dF}{dt} = 0$. $\frac{dF}{dt} = 2n(1-t)^{2n-1}(-1)(1-2t)^{1-n} + (1-t)^2 n(1-n)(1-2t)^{-n}(-2) = 0 \implies t(n+1) - 1 = 0 \implies t = \frac{1}{n+1}$. We substitute back $\alpha = \frac{(1-t)^2}{1-2t} = \frac{n^2}{n^2-1}$ and $\beta = (1-t)^2 = \frac{n^2}{(n+1)^2}$.
Thus we optimized $\min(\alpha^{n-1}\beta)$ when $\alpha = \frac{n^2}{n^2-1}$,$\beta = \frac{n^2}{(n+1)^2}$ and $t = \frac{1}{n+1}$.

$$\begin{aligned} \det(Q) &= \alpha^{n-1}\beta \\ &= (1 + \frac{1}{n^2-1})^{n-1}(1 - \frac{2n+1}{(n+1)^2}) \\ &\leq (e^{\frac{1}{n^2-1}})^{n-1} e^{\frac{-2}{n+1}} \qquad \text{(for } x > 0, 1 + x \leq e^x \forall x \text{)} \\ &= e^{-\frac{1}{n+1}} \end{aligned}$$

Hence the factor of shrinkage in the volume is $\sqrt{\det(Q)} = e^{-\frac{1}{2(n+1)}}$.

Above we restrict B as a ball, but we can relax B to any ellipsoid. In general, if $E_k = E(Q_k, c_k)$, then $E_{k+1} = E(Q_{k+1}, c_{k+1})$, where

$Q_{k+1} = \frac{n^2}{n^2-1}(Q_k - \frac{2}{n+1} v v^t)$ where vv^t is the matrix $a_{ij} = v_i \cdot v_j$

$c_{k+1} = c_k - \frac{1}{n+1} v$

$v = \frac{Q_k a_i}{\sqrt{a_i^t Q_k a_i}}$ where a_i is the vector defining the violated constant.

3 Implementation issues

3.1 Rounding

The fact that not all $\sqrt{a_i^t Q_k a_i}$ are rational numbers and the fact that the numbers defining the ellipsoid can grow too fast, leads to a modification of the above, namely rounding all the numbers. We round the entries of Q and all other numbers in the transformation while:

1. Q must be positive definite
2. the "rounded" ellipsoid must still contain $\frac{1}{2}E_k$
3. the volume of the rounded ellipsoid is not much bigger than the original one

For this rounding process, we have the following parameters:
P: for the accuracy needed, the number of digits we round after.
ϵ: the scaling factor of the ellipsoid after rounding.

For example, if we select $N = 50n^2L, P = 8N, \epsilon = 1 + \frac{1}{4n^2}$, it will give a satisfactory result.

3.2 Ellipsoid Algorithm is *not* a strongly polynomial algorithm

Definition 3.1. elementary arithmetic operations: We count *addition*, *subtraction*, *multiplication*, *division*, *comparison* as one step rather than the number of moves of a head on *Turing machine*. We call it **arithmetic model**.

Definition 3.2. strongly polynomial time: We say that an algorithm runs in *strongly polynomial time* if the algorithm is a polynomial space algorithm and performs a number of elementary arithmetic operations which is bounded by a polynomial in the number of input numbers. Thus a strongly polynomial algorithm is a polynomial space algorithm (in standard Turing machine model) and a polynomial time algorithm in the arithmetic model.

We say Ellipsoid Algorithm runs in polynomial time at most $N = 16n(n+1)L$ iterations, each iteration is polynomial time. However, $L = \log(|P|)$ where P is the nonzero coefficients in A, b, c. It means the running time is not only depends on the number of input numbers $m \times n$, but also depends on P which is the value of the input numbers. By the definition of *strongly polynomial time*, a strongly polynomial algorithm must perform a number of elementary arithmetic operations bounded by a polynomial in the **number** of input numbers. Obviously, Ellipsoid Algorithm performs arithmetic operations bounded also by the value of the input numbers. Thus, Ellipsoid Algorithm is not a *strongly polynomial algorithm*.

3.3 Abstraction for the algorithm oracle

Definition 3.3. oracle: we can imagine an *oracle* as a device that can solve problem for us. We make no assumption on how a solution is found by the oracle.

Definition 3.4. oracle algorithm: is an algorithm which can "ask questions" from an oracle and can use the answers supplied.

Definition 3.5. polynomial transformation: suppose we have two decision problems Π and Π', a polynomial transformation is an algorithm which given an encoded instance σ of Π, produces in polynomial time an encoded instance σ' of Π' such that the following holds: For every instance σ of Π, the answer is "yes" if and only if the answer to σ' is "yes".

Clearly, if there is a polynomial algorithm to solve Π' then by polynomially transforming any instance of Π to an instance of Π' there is also a polynomial algorithm to solve Π.
Optimization problems are not decision problems. But in minimization(maximization) problems, we can ask "Is there a feasible solution whose value is at least(most) Q?". If we can solve this "yes/no" question in polynomial time(i.e Ellipsoid Algorithm), we can continue asking "Is there a feasible solution whose value is at least(most) $\frac{Q}{2}$?"..."Is there a feasible solution whose value is at least(most) $\frac{Q}{2^n}$?". Thus we are using binary search to find the optimal solution. The number of iterations is $n = \log(Q)$, it is the length of input Q. Obviously we run polynomial times of iterations, each iteration is also polynomial. Thus we can solve an optimization problem in polynomial time through a polynomial oracle. For Ellipsoid Algorithm to work, we need answers to the following queries:

1. Is $x \in P$?, we call it "Membership Oracle"
2. If *NOT*, a plane separating x from P, we call it "Separation Oracle"

So Ellipsoid Algorithm really tells us that given those oracles to a problem and guarantees of not too large initial search space and not too small possible volume for $P \neq \emptyset$, we get a polynomial solution.

4 Tutorial

4.1 Yao's min-max principle

From last week's lecture, we know Yao's min-max principle:

$$min_{A \in \mathcal{A}} E[C(\mathcal{I}_p, A)] \leq max_{I \in \mathcal{I}} E[C(I, A_q)]$$

where $\mathcal{A}$ is a set of algorithms, $\mathcal{I}$ is a set of inputs, $C(I, A)$ is the running time of algorithm A with input I.

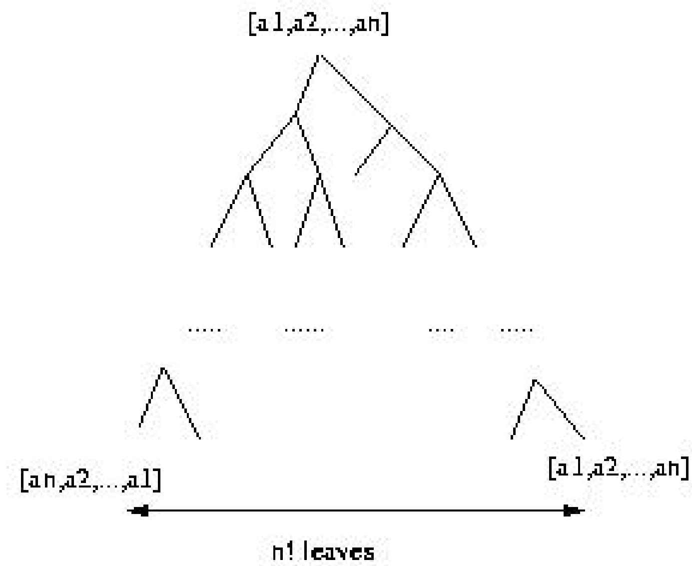

Figure 5: A decision tree for sorting problem. For n input numbers, we have $n!$ permutations. Hence, there are $n!$ leaves in the tree.

The expected runnning time of the best deterministic algorithm on some input distribution is a lower bound for the expected running time of the best randomized algorithm on an arbitrary input.

We have the following theorem:

Theorem 4.1. *The worst-case expected time of any randomized sorting algorithm is* $\Omega(nlog(n))$.

Proof. See Figure[**??**], it is a sorting problem with a random permutation of n numbers. Since there are $n!$ possible permutations of the n input numbers, the decision tree must have $n!$ leaves. Different permutation goes to the different branch of the decision tree.

Since $\frac{n!}{2} > 2^{\alpha n \log(n)}$ for some $0 < \alpha < 1$, there are at least $\frac{1}{2}$ of the leaves are at the distance $\Omega(nlog(n))$ from the root. Thus the expected running time of any deterministic algorithm on this random input is $\Omega(n\log(n))$. $\square$

4.2 Example of Ellipsoid Algorithm

We show an example to run Ellipsoid Algorithm: $Ax \leq b$

1. $-x_1 + 0.2x_2 \leq -8$

2. $x_1 + x_2 \leq 4$

3. $0.3x_1 - x_2 \leq 9$

Step 1:

Set $n = 2, c_0 = (0,0)^t, Q_0 = \begin{pmatrix} 13^2 & 0 \\ 0 & 13^2 \end{pmatrix}$. Check whether $c_0 = (0,0)^t$ satisifies the inequalities above. We find (1) is not satisfied. Thus let $a = (-1, 0.2)^t$, we get

$$v = \frac{Q_0 a}{\sqrt{a^t Q_0 a}} = (-12.7475, 2.5495)^t.$$

$$c_1 = c_0 - \frac{v}{n+1} = (0,0)^t - \frac{(-12.7475, 2.5495)^t}{2+1} = (4.2492, -0.8498)^t.$$

$$Q_1 = \frac{n^2}{n^2-1}(Q_0 - \frac{2}{n+1}vv^t) = \begin{pmatrix} 80.8889 & 28.8889 \\ 28.8889 & 219.5556 \end{pmatrix}.$$

Step 2:

We check whether $c_1 = (4.2492, -0.8498)^t$ satisfies the inequalities. We find (1) is not satisfied. Thus let $a = (-1, 0.2)^t$, we get

$$v = \frac{Q_1 a}{\sqrt{a^t Q_1 a}} = (-8.4984, 1.6997)^t.$$

$$c_2 = c_1 - \frac{v}{n+1} = (4.2492, -0.8498)^t - \frac{(-8.4984, 1.6997)^t}{2+1} = (7.0820, -1.4164)^t.$$

$$Q_2 = \frac{n^2}{n^2-1}(Q_1 - \frac{2}{n+1}vv^t) = \begin{pmatrix} 43.6543 & 51.3580 \\ 51.3580 & 290.1728 \end{pmatrix}$$

...

Step 8:

We find $c_8 = (7.4929, -6.4397)^t$. This is a solution satisfies the inequalities. So we are done.

References

[1] Martin Grötschel, László Lovász,Alexander Schrijver. Geometric Algorithms and Combinatorial Optimization, 2nd Corrected Edition. `Springer-Verlag`, pages 26-33 1993.