

In this lecture, we discuss some polynomial-time algorithms for Linear Programming problems (along with some that are not in polynomial time). In particular, we touch on several LP algorithms, and examine in-depth the Center-of-Gravity and Ellipsoid algorithms.

## 1 LP Approaches

Before we give a high-level idea of how Ellipsoid works, let us just mention the main approaches to solve LPs. For now, let  $K = \{x \mid Ax \geq b\} \subseteq \mathbb{R}^n$ , and we want to minimize  $c^\top x$  such that  $x \in K$ .

**Simplex:** This is the first algorithm for solving LPs that most of us see. Developed by George Dantzig in 1947, at a high level it starts at a vertex of the polyhedron  $K$ , and moves from vertex to vertex without decreasing the objective function value, until it reaches an optimal vertex. (The convexity of  $K$  ensures that such a sequence of steps is possible.) The strategy to choose the next vertex is called the pivot rule. Unfortunately, for most pivot rules people have proposed, there are examples on which the following the pivot rule takes exponential number of steps. One early description of the method appears in [DOW55].

**Ellipsoid:** The first polynomial-time algorithm for LP solving proposed by Khachiyan [Kha79]; this is discussed in detail in Section 3. Check out [this historical survey](#) by Bland, Goldfarb and Todd.

**Interior Point:** The second approach to get a polynomial-time algorithm for LPs; proposed by Karmarkar [Kar84]. This is widely used in practice, and has many nice theoretical properties too. We will discuss this in a later lecture if there is time and interest.

**Geometric Algorithms for LPs:** These approaches are geared towards solving LPs fast when the number of dimensions is small. Suppose we let  $d$  denote the number of dimensions and  $m$  be the number of constraints, these algorithms often allow a poor runtime in  $d$ , at the expense of getting a good dependence on  $m$ . As an example, [this algorithm](#) of Seidel's has a runtime of  $O(m \cdot d!) = O(m \cdot d^{d/2})$  [Sei91]. A different algorithm of Clarkson has a runtime of  $O(d^2 m) + d^{O(d)} O(\log m)^{O(\log d)}$ . Perhaps the best of these algorithms is one by Matousek, Sharir, and Welzl [MSW96], which has a runtime of

$$O(d^2 m) + e^{O(\sqrt{d \log d})}.$$

For details and references, see [this survey](#) by Dyer, Megiddo, and Welzl.

**Center of Gravity Algorithm:** This is discussed in detail in Section 2.

## 2 The Center-of-Gravity Algorithm

In this section, we discuss the center-of-gravity algorithm in the context of constrained convex minimization. Besides being interesting in its own right, I find it a good lead-in to Ellipsoid, since it gives some intuition about high-dimensional bodies and their volumes.

Given a convex body  $K \subseteq \mathbb{R}^n$ , and a convex function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , we want to approximately minimize  $f(x)$  over  $x \in K$ . First, recall that the *center of gravity* of a set  $K$  is defined as the point  $c \in \mathbb{R}^n$  such that

$$c := \frac{\int_{x \in K} x \, dx}{\text{vol}(K)} = \frac{\int_{x \in K} x \, dx}{\int_{x \in K} dx},$$

where  $\text{vol}(K)$  is the volume of the set  $K$ .<sup>1</sup>

The following lemma captures the one fact about the center-of-gravity that we use in our algorithm.

**Lemma 17.1** (Grünbaum [Grü60]). *For any convex set  $K \in \mathbb{R}^n$  with a center of gravity  $c \in \mathbb{R}^n$ , and any halfspace  $H = \{x \mid a^\top(x - c) \geq 0\}$  passing through  $c$ ,*

$$\frac{1}{e} \leq \frac{\text{vol}(K \cap H)}{\text{vol}(K)} \leq \left(1 - \frac{1}{e}\right).$$

## 2.1 The Algorithm

In 1965, Levin [Lev65] and Newman [New65] independently (on opposite sides of the iron curtain) proposed the following algorithm.

---

**Algorithm 1** Center-of-Gravity( $K, f, T$ )

---

```

 $K_0 \leftarrow K$ 
for  $t = 0, 1, \dots, T$  do
    at step  $t$ , let  $c_t$  = the center of gravity of  $K_t$ 
    calculate the gradient  $\nabla f(c_t)$ 
     $K_{t+1} = K_t \cap \{x \mid \langle \nabla f(c_t), x - c_t \rangle \leq 0\}$ 
end for
return  $\hat{x} := \arg \min_{t \in \{0, 1, \dots, T\}} f(c_t)$ 

```

---

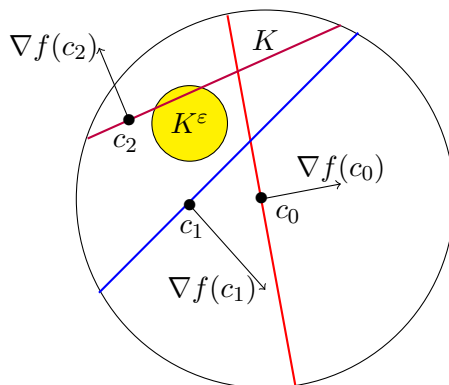


Figure 17.1: Sample execution of first three steps of the Center-of-Gravity algorithm.

---

<sup>1</sup>This is the natural analog of the center of gravity of  $n$  points  $x_1, x_2, \dots, x_N$ , which is defined as  $\frac{\sum_i x_i}{N}$ . See this blog post for a discussion about the center-of-gravity of an arbitrary measure  $\mu$  defined over  $\mathbb{R}^n$ .

## 2.2 Analysis of Center of Gravity

**Theorem 17.2.** *Let  $B \geq 0$  such that  $f : K \rightarrow [-B, B]$ . If  $\hat{x}$  is the result of the algorithm, and  $x^* = \arg \min_{x \in K} f(x)$ , then*

$$f(\hat{x}) - f(x^*) \leq 2B \left(1 - \frac{1}{e}\right)^{\frac{T}{n}} \leq 2B \cdot \exp(-T/3n).$$

Hence, for any  $\delta \leq 1$ ,  $f(\hat{x}) - f(x^*) \leq \delta$  as long as  $T \geq 3n \ln \frac{2B}{\delta}$ .

*Proof.* By Grünbaum’s lemma, it immediately follows that at any time  $t$ ,  $\text{vol}(K_t) \leq \text{vol}(K) \cdot (1 - \frac{1}{e})^t$ . Now for some  $\varepsilon \leq 1$ , define the body  $K^\varepsilon = \{(1 - \varepsilon)x^* + \varepsilon x \mid x \in K\}$ . The following facts are immediate:

- $\text{vol}(K^\varepsilon) = \varepsilon^n \cdot \text{vol}(K)$ .
- The value of  $f$  on any point  $y = (1 - \varepsilon)x^* + \varepsilon x \in K^\varepsilon$  is

$$\begin{aligned} f(y) = f((1 - \varepsilon)x^* + \varepsilon x) &\leq (1 - \varepsilon)f(x^*) + \varepsilon f(x) \leq (1 - \varepsilon)f(x^*) + \varepsilon B \\ &\leq f(x^*) + \varepsilon(B - f(x^*)) \leq f(x^*) + 2\varepsilon B. \end{aligned}$$

- Consider step  $t$  such that  $K^\varepsilon \subseteq K_t$  but  $K^\varepsilon \not\subseteq K_{t+1}$ . Let  $y \in K^\varepsilon \cap (K_t \setminus K_{t+1})$  be a point that is “cut off”. Then  $f(c_t) < f(y)$ . Indeed,

$$f(y) \geq f(c_t) + \langle \nabla f(c_t), y - c_t \rangle$$

and  $\langle \nabla f(c_t), y - c_t \rangle > 0$  since  $y \in K_t \setminus K_{t+1}$ .

If we define  $\varepsilon := (1 - 1/e)^{T/n}$ , then after  $T$  steps either  $c_T \in K^\varepsilon$ , or some point of  $K^\varepsilon$  has been cut off at some step  $t$ . In either case, we know that some center in  $\{c_t\}_{t \in [T]}$  achieves  $f(c) \leq 2B\varepsilon$ , which gives us the first claim. The second claim follows by substituting  $T \geq 3n \ln \frac{2B}{\delta}$  into the first claim, and simplifying.  $\square$

## 2.3 Comments

Observe that if we want an error of  $\delta$  for minimizing a convex function, the number of steps  $T$  depends on  $\log(1/\delta)$ ; compare this to gradient descent which requires  $1/\delta^2$  steps. This is an exponentially better convergence rate! A useful way to think of this: improving the precision from  $\delta$  to  $\delta/2$  is like getting one more bit of precision. Getting one more bit of precision in gradient descent requires quadrupling the number of steps, but in this algorithm it requires increasing by an additive constant. This is why such a convergence depending as  $\log 1/\delta$  is called *linear convergence* in the convex optimization literature.

One downside with this approach: now the number of steps explicitly depends on the number of dimensions  $n$ . Contrast with gradient descent, where the number of steps depended on other factors (a bound on the gradient, the diameter of the polytope), but not explicitly on the dimension.

Finally, there’s the all-important question: *how do you compute the center of gravity?* This is a difficult problem—it is #P-hard, which means it is at least as hard as counting the number of satisfying assignments to a SAT instance. In 2002, Bertsimas and Vempala [BV04] suggested a way to find approximate centers-of-gravity using sampling random points from convex polytopes (which in turn was based on random walks).

## 3 An Overview of the Ellipsoid Algorithm

### 3.1 Theorems about the Ellipsoid Algorithm

The Ellipsoid algorithm is usually attributed to N. Shor [SZ71]; the fact that this algorithm gives a polynomial-time algorithm for linear programming was a breakthrough result due to Khachiyan [Kha80], and was **front page news at the time**. Let us survey some theorem statements that will be most useful to design algorithms. A great source of information about this algorithm is the Gröschel-Lovasz-Schrijver book [GLS88].

The second-most important theorem about the Ellipsoid algorithm is the following. (Recall from Lecture 10 that *b.f.s.* stands for **basic feasible solution**.) In this lecture,  $\langle A \rangle, \langle b \rangle, \langle c \rangle$  denote the number of bits required to represent of  $A, b, c$  respectively.

**Theorem 17.3.** *Given an LP  $\min\{c^\top x \mid Ax \geq b\}$ , the Ellipsoid algorithm produces an optimal b.f.s. for the LP, in time polynomial in  $\langle A \rangle, \langle b \rangle, \langle c \rangle$ .*

One may ask: is the large runtime just because the numbers in  $A, b, c$  may be very large, and hence just doing basic arithmetic on these numbers may require large amounts of time. Unfortunately, that is not the case. Even if we count the number of arithmetic operations we need to perform, the Ellipsoid algorithm performs  $\text{poly}(\max(\langle A \rangle, \langle b \rangle, \langle c \rangle))$  operations. A stronger guarantee would have been that the number of arithmetic operations is  $\text{poly}(m, n)$  (where the matrix  $A \in \mathbb{Q}_{m \times n}$ —such an algorithm would be called a *strongly polynomial-time* algorithm. This remains a major open question.

#### 3.1.1 Separation Implies Optimization

In order to talk about the Ellipsoid algorithm, as well as to state the next (and most important) theorem about Ellipsoid, we need a definition.

**Definition 17.4.** Strong Separation Oracle For a convex set  $K \subseteq \mathbb{R}^n$ , a *strong separation oracle* for  $K$  is an algorithm that takes a point  $\hat{x} \in \mathbb{R}^n$  and correctly outputs one of:

- (i) “ $\hat{x} \in K$ ”, or
- (ii) “ $\hat{x} \notin K$ ”, as well as  $a \in \mathbb{R}^n, b \in \mathbb{R}$  such that  $K \subseteq \{x \mid a^\top x \leq b\}$  but  $a^\top \hat{x} > b$ .

In case (ii), the hyperplane  $a^\top x = b$  is a “separating hyperplane” between  $x$  and  $K$ . Figure 17.2 shows a two-dimensional example of the output of a strong separation oracle in the failure case.

**Theorem 17.5.** *Given any finite LP  $\min\{c^\top x \mid Ax \geq b\}$  with  $x \in \mathbb{R}^n$ , and given access to a strong separation oracle for  $K = \{x \mid Ax \geq b\}$ , the Ellipsoid algorithm produces a b.f.s. for the LP in time  $\text{poly}(n, \max_i \langle a_i \rangle, \max_i \langle b_i \rangle, \langle c \rangle)$ .*

Note that there is no dependence on the number of constraints in the LP; as long as each constraint has a reasonable bit complexity, and we can define a separation oracle for the polytope, we can solve the LP. This is often summarized by saying: “*separation*  $\implies$  *optimization*”.

Let us give two examples of exponential-sized LPs, for which we can give a separation oracles, and hence optimize over them.

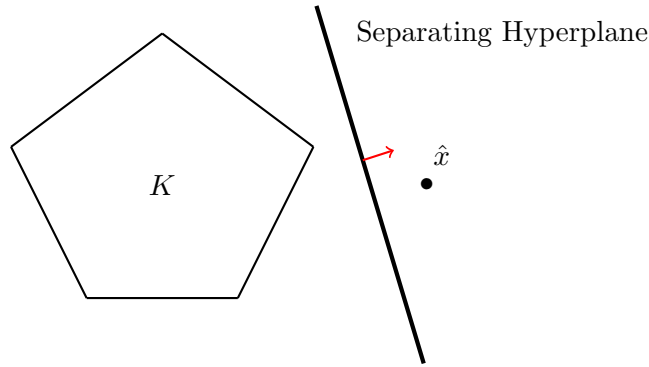


Figure 17.2: Example of separating hyperplanes

### 3.1.2 Exponential-Sized LPs

Given a directed graph  $G = (V, E)$  with edge weights  $w_e \in \mathbb{R}$ , a special vertex  $r \in V$ , we can define the following LP:

$$\begin{aligned} \min \quad & \sum w_e x_e \\ \text{Subject to:} \quad & \sum_{e \in \partial^+ S} x_e \geq 1 & \forall S \subseteq V \text{ s.t. } r \notin S, S \neq \emptyset \\ & 1 \geq x_e \geq 0 & \forall e \in E \end{aligned}$$

A little thought shows that if we consider any arborescence  $T$  in the digraph, its characteristic vector  $\chi_T \in \{0, 1\}^{|E|}$  is a feasible solution for this LP: each set not containing the root must have an arc leaving the set.

How do we solve this LP? It has an exponential number of constraints! We can solve it in  $\text{poly}(|V|, |E|)$  time using Ellipsoid as long as we can implement a strong separation oracle in polynomial time. I.e., given a point  $x \in \mathbb{R}^{|E|}$ , we need to figure out if  $x$  satisfies all the constraints. It is easy to check if  $x_e \in [0, 1]$  for all edges  $e$  (since there are only  $|E|$  many edges), but what about the  $2^{n-1} - 1$  “cut” constraints? Consider the graph  $H$  which has the same nodes and arcs as  $G$ , where arc capacities are  $x_e$ . For each node  $v \neq r$ , find the minimum  $v$ - $r$  cut in  $H$ . There exists some violated set  $S$  if and only if the minimum cut from any node in  $S$  to  $r$  has cut value strictly less than 1. Hence we can find a violated cut using  $n - 1$  min-cut computations. <sup>2</sup>

Here’s another (famous) example: Given a general undirected graph  $G = (V, E)$ , consider the

---

<sup>2</sup>BTW, using ideas from Lecture #3, we can show that the vertices of this polytope are precisely all the arborescences in  $G$ , and hence using Edmonds’ algorithm to find the min-cost arborescence is another way of solving this LP!

following polytope with a variable  $x_e$  corresponding to the edges of the graph.

$$\begin{aligned} & \max \sum w_e x_e \\ \text{Subject to: } & \sum_{e \in \partial v} x_e \leq 1 && \forall v \in V \\ & \sum_{e \in \binom{S}{2}} x_e \leq \left\lfloor \frac{|S|}{2} \right\rfloor && \forall S \subseteq V \\ & x \geq 0 \end{aligned}$$

A little thought shows that if we consider any matching  $M$  in the graph, its characteristic vector  $\chi_M \in \{0, 1\}^{|E|}$  is a feasible solution for this LP. Again, the LP has an exponential number of constraints. Regardless, Theorem 17.5 says that we can solve this LP in time  $\text{poly}(n)$ , as long as we can give a poly-time separation oracle for the polytope. I.e., given a point  $\hat{x} \in \mathbb{R}^{|E|}$ , and suppose  $\hat{x} \notin K$ , then we need to output some separating hyperplane. This time it's a little non-trivial but can be done [this way](#), even though the number of constraints is exponential.

Suppose, instead of minimizing a convex function  $f$  over a convex set  $K$ , we just wanted to find some point  $x \in K$ , or to report that  $K$  was the empty set. This is a *feasibility* problem, and is a simpler problem than optimization. In fact, the GLS book [GLS88] shows that feasibility is not much easier than optimization: under certain conditions the two problems are essentially equivalent.

### 3.2 Ellipsoid for Feasibility

We now consider the actual ellipsoid method. For simplicity, we begin with the problem of determining whether or not an LP is feasible. Let's define a particular feasibility problem. Given some description of a polytope  $K$ , and two scalars  $R, r > 0$ , suppose we are guaranteed that

- (a)  $K \subseteq \text{Ball}(0, R)$ , and
- (b) either  $K = \emptyset$ , or else some ball  $\text{Ball}(c, r) \subseteq K$  for some  $c \in \mathbb{R}^n$ .

The feasibility problem is to figure out which of the two cases in (b) holds, and if  $K \neq \emptyset$  then to also find a point  $x \in K$ . For today we'll assume that  $K$  is given by a strong separation oracle.

**Theorem 17.6.** *Given  $K, r, R$  as above (and a strong separation oracle for  $K$ ), it is possible to solve the feasibility problem using  $O(n \log(R/r))$  calls to the oracle. (This theorem assumes we can perform exact arithmetic on real numbers.<sup>3</sup>)*

The proof of this theorem captures some of the major ideas of the Ellipsoid algorithm. The basic structure is simple: At each step  $t$ , we are given a current ellipsoid  $\mathcal{E}_t$  that is guaranteed to contain the set  $K$  (assuming we have not found a point  $x \in K$  yet). The initial ellipsoid is  $\mathcal{E}_0 = \text{Ball}(0, R)$ .

- At step  $t$ , we ask the oracle: is the center  $c_t$  of  $\mathcal{E}_t$  in  $K$ ?

If the oracle answers “Yes”, we are done. Else the oracle returns a separating hyperplane  $a^\top x = b$ , such that  $a^\top c_t > b$  but  $K \subseteq H_t := \{x : a^\top x \leq b\}$ . Hence we know that  $K$  is contained in the piece of the ellipsoid given by  $\mathcal{E}_t \cap H_t$ . Note that since the half-space  $H_t$  does

---

<sup>3</sup>Where is this assumption used? In computing the new ellipsoid, we need to take square-roots. If we were to round numbers, that could create all sorts of problems. Part of the complication in [GLS88] comes from these issues.

not contain the center  $c_t$  of the ellipsoid, this piece of the ellipsoid is less than half the entire ellipsoid.

The crucial idea is this: we find another (small-volume) ellipsoid  $\mathcal{E}_{t+1}$  that contains this piece  $\mathcal{E}_t \cap H_t$  (and hence also  $K$ ). And we continue.

How do we show that we make progress? The second crucial idea is to show that  $\text{vol}(\mathcal{E}_{t+1})$  is considerably smaller than  $\text{vol}(\mathcal{E}_t)$ . It is possible (see Section 4 and references) to construct an ellipsoid  $\mathcal{E}_{t+1} \supseteq \mathcal{E}_t \cap H_t$  such that

$$\frac{\text{vol}(\mathcal{E}_{t+1})}{\text{vol}(\mathcal{E}_t)} \leq e^{-\frac{1}{2(n+1)}}.$$

Therefore, after  $2(n+1)$  iterations, the ratio of the volumes is down by at least a factor of  $\frac{1}{e}$ , which is exactly the kind of constant-fraction reduction we are looking for. Why? By our assumptions, initially  $\text{vol}(K) \leq \text{vol}(\text{Ball}(0, R))$ ; also if  $K \neq \emptyset$ , then  $\text{vol}(K) \geq \text{vol}(\text{Ball}(0, r))$ . Hence, if after  $2(n+1) \ln(R/r)$  steps, none of the ellipsoid centers have been inside  $K$ , we know that  $K$  must be empty.

### 3.3 Ellipsoid for Convex Optimization

Now we want to solve  $\min\{f(x) \mid x \in K\}$ . Again, assume that  $K$  is given by a strong separation oracle, and we have numbers  $R, r$  as in the previous section.

The general structure will be familiar by now: it combines the ideas from both things we've done.

- Let  $x_0$  be the origin,  $\mathcal{E}_0 = \text{Ball}(0, R)$ ,  $K_0 = K$ .
- At time  $t$ , ask the separation oracle: “Is the center  $c_t$  of ellipsoid  $\mathcal{E}_t$  in the convex body  $K_t$ ?”
  - Yes:** Define half-space  $H_t := \{x \mid \langle \nabla f(c_t), x - c_t \rangle \leq 0\}$ . Observe that  $K_t \cap H_t$  contains all the points in  $K_t$  having value at most  $f(c_t)$ .
  - No:** In this case the separation oracle also gives us a separating hyperplane. This defines a half-space  $H_t$  such that  $c_t \notin H_t$ , but  $K_t \subseteq H_t$ .

In both cases, set  $K_{t+1} \leftarrow K_t \cap H_t$ , and  $\mathcal{E}_{t+1}$  is an ellipsoid containing  $\mathcal{E}_t \cap H_t$ . Since we knew that  $K_t \subseteq \mathcal{E}_t$ , we maintain that  $K_{t+1} \subseteq \mathcal{E}_{t+1}$ .

- Finally, after  $T = 2n(n+1) \ln(R/r)$  rounds either we don't find any point in  $K$ —then we say “ $K$  is empty”—or else we output  $\arg \min\{f(c_t) \mid t \in 0 \dots T \text{ such that } c_t \in K_t\}$ .

One subtle but important question: we make queries to a separation oracle for  $K_t$ , but we are promised only a separation oracle for  $K_0 = K$ . To handle this, suppose we have a strong separation oracle for  $K_t$ . To get a strong separation oracle for  $K_{t+1} = K_t \cap H_t$  we proceed as follows:

Given  $\hat{x} \in \mathbb{R}^n$ , query the oracle for  $K_t$  at  $\hat{x}$ . If  $\hat{x} \notin K_t$ , the separating hyperplane for  $K_t$  also works for  $K_{t+1}$ . Else, if  $\hat{x} \in K_t$ , check if  $\hat{x} \in H_t$ . If so,  $\hat{x} \in K_{t+1} = K_t \cap H_t$ . Otherwise, the defining hyperplane for  $H_t$  is a separating hyperplane between  $\hat{x}$  and  $K_{t+1}$ .

This procedure emits a new strong separation oracle for  $K_{t+1}$ , and we can thus obtain a strong separation oracle for  $K_n$  for any  $n \geq 0$  by induction.

Again, adapting the analysis from the previous sections gives us the following result:

**Theorem 17.7.** *Given  $r, R$  and a strong separation oracle for a convex body  $K$ , and a function  $f : K \rightarrow [-B, B]$ , the Ellipsoid algorithm run for  $T$  steps either correctly reports that  $K = \emptyset$ , or else produces a point  $\hat{x}$  such that*

$$f(\hat{x}) - f(x^*) \leq \frac{2BR}{r} e^{-\frac{t}{2n(n+1)}}.$$

Note the similarity to Theorem 17.2, as well as the differences: the exponential term is slower by a factor of  $2(n+1)$ , which arises because the volume of the ellipsoids shrinks much slower than in Grünbaum's lemma. Also, we lose a factor of  $R/r$  because  $K$  is potentially smaller than the starting body by precisely this factor. (Again, this presentation ignores precision issues, and assumes we can do exact real arithmetic.)

## 4 Getting the New Ellipsoid

This brings us to the final task: given a current ellipsoid  $\mathcal{E}$  and a half-space  $H$  that does not contain its center, find a small ellipsoid  $\mathcal{E}'$  that contains  $\mathcal{E} \cap H$ .

Consider, for example, the general form of an ellipsoid in  $\mathbb{R}^2$ . We can write this as

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1.$$

Or in matrix notation we could also say

$$\begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} 1/a^2 & \\ & 1/b^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq 1$$

More generally, we should think of any ellipsoid,  $\mathcal{E}$ , as some invertible linear transformation,  $L$ , applied to the unit ball  $B(0, 1)$ , and then shifted to the correct center,  $c$ . In other words:

$$\begin{aligned} L(B(0, 1)) &= \{Lx : x^T x \leq 1\} \\ &= \{y : (L^{-1}y)^T (L^{-1}y) \leq 1\} \\ &= \{y : y^T (LL^T)^{-1} y \leq 1\} \\ &= \{y : y^T Q^{-1} y \leq 1\} \end{aligned}$$

Where  $Q^{-1} = LL^T$  is a positive semidefinite matrix. For an ellipsoid centered at  $c$  we simply write

$$\{y : (y - c)^T Q^{-1} (y - c) \leq 1\}.$$

It is helpful to note that for a ball,  $A$ ,

$$\text{vol}(L(A)) = \text{vol}(A) \cdot |\det(L)| = \sqrt{\det(Q)} \text{vol}(A)$$

The problem of finding an ellipsoid  $\mathcal{E}_{t+1}$  given the ellipsoid  $\mathcal{E}_t$  then boils down to finding a matrix  $Q_{t+1}$  and a center  $c_{t+1}$  such that the resulting ellipsoid contain  $\mathcal{E}_t \cap H_t$  and satisfies

$$\frac{\text{vol}(\mathcal{E}_{t+1})}{\text{vol}(\mathcal{E}_t)} \leq e^{-1/2(n+1)}$$

The basic idea is simple:



- Consider the special case where  $\mathcal{E}$  is the ball  $\text{Ball}(0, 1)$ , and  $H = \{x \mid x_1 \geq 0\}$ . We can choose a center  $c_1$  and matrix  $Q_1$  that contains the specified half ball. Using our expression for the volume of an ellipsoid, we can also argue that we have constructed a sufficiently small ellipsoid. In fact, we will aim to choose the minimum volume ellipsoid that meets our requirements.
- We can then see that applying a similar transformation to any ellipsoid will, in fact, meet our requirements, thus telling us how to construct  $\mathcal{E}_{t+1}$  given some  $\mathcal{E}_t$ .

This result is summarized in the following claim:

**Claim 17.8.** *Given the ellipsoid  $\mathcal{E}_t = (c_t, Q_t)$  and the separating hyperplane  $a_t^T(x - c_t) \leq 0$ , the ellipsoid  $\mathcal{E}_{t+1} = (c_{t+1}, Q_{t+1})$  is found by taking*

$$c_{t+1} = c_t - \frac{1}{n+1}h$$

and

$$Q_{t+1} = \frac{n^2}{n^2 - 1} \left( Q_t - \frac{2}{n+1}hh^T \right)$$

where  $h = \sqrt{a_t^T Q_t a_t}$ .

For a more detailed description and proof of this process, see [these notes](#) from our LP/SDP course for details.

## References

- [BV04] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, July 2004. [2.3](#)
- [DOW55] George B Dantzig, Alex Orden, and Philip Wolfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195, 1955. [1](#)
- [GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, Berlin, 1988. [3.1](#), [3.1.2](#), [3](#)
- [Grü60] B. Grünbaum. Partitions of mass-distributions and of convex bodies by hyperplanes. *Pacific J. Math.*, 10:1257–1261, 1960. [17.1](#)
- [Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984. [1](#)
- [Kha79] Leonid Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20(191–196), 1979. [1](#)
- [Kha80] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. [3.1](#)
- [Lev65] A. Y. Levin. On an algorithm for the minimization of convex functions over convex functions. *Soviet Mathematics Doklady*, 16(1244–1247), 1965. [2.1](#)
- [MSW96] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996. [1](#)

- [New65] D. J. Newman. Location of the maximum on unimodal surfaces. *J. ACM*, 12(3):395–398, July 1965. [2.1](#)
- [Sei91] Raimund Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, 6(3):423–434, 1991. [1](#)
- [SZ71] Naum Z Shor and NG Zhurbenko. The minimization method using space dilatation in direction of difference of two sequential gradients. *Kibernetika*, 7(3):51–59, 1971. [3.1](#)