CS 270 Algorithms **Spring 2021**

Lecture 4: Centroid and Ellipsoid Algorithm

Lecturer: Prasad Raghavendra Scribe: Jonathan Pan, Jason Xiong, Yunduan Lin

4.1 Centroid Algorithm

4.1.1 Definitions

Definition 4.1 (Centroid)**.** *The centroid c of a convex set $\mathcal{K}$ is*

$$c = \frac{\int_{x\in\mathcal{K}} x\,dx}{vol(\mathcal{K})} = \frac{\int_{x\in\mathcal{K}} x\,dx}{\int_{x\in\mathcal{K}} 1\,dx}$$

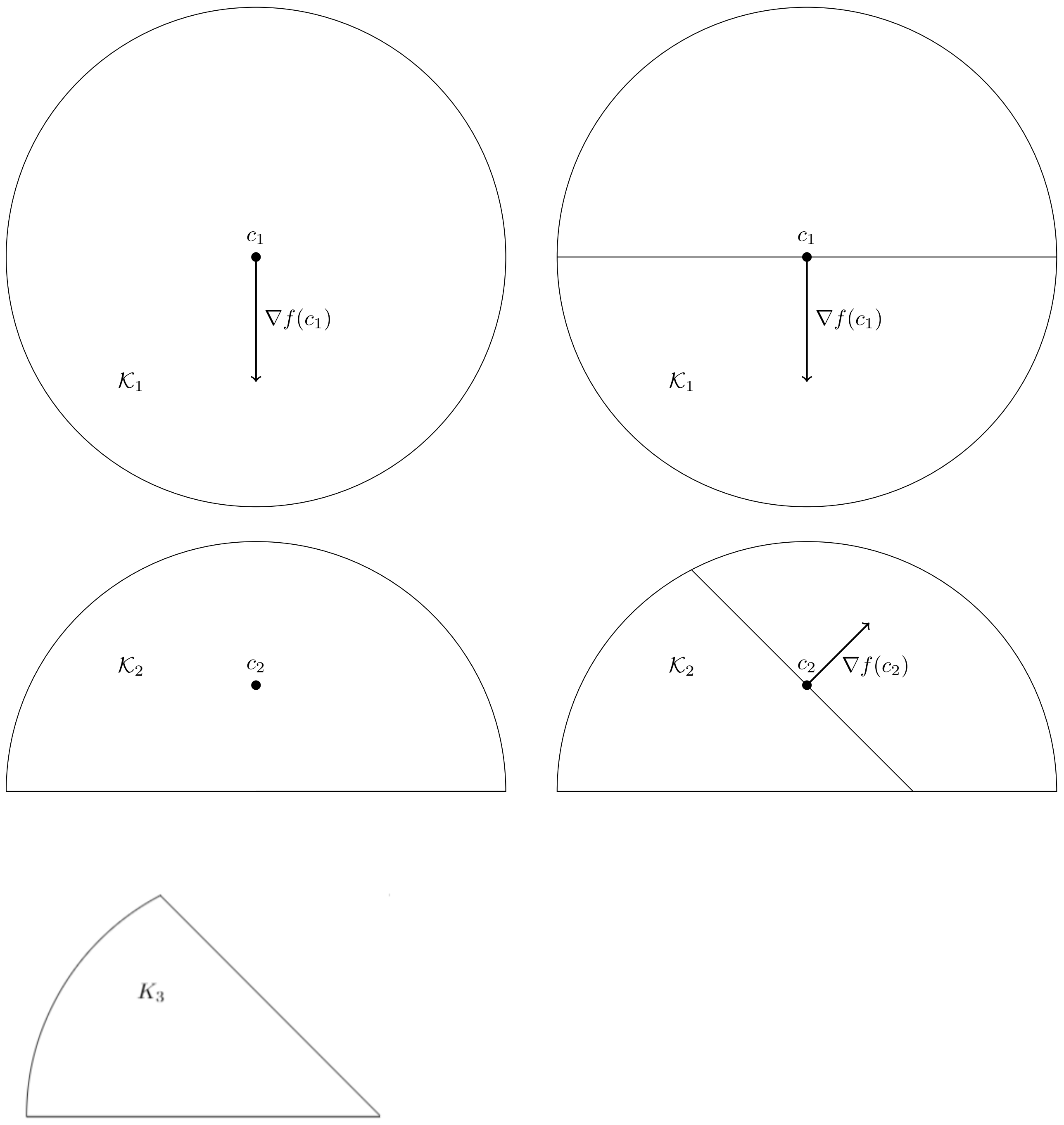
Definition 4.2 ($\mathcal{K}^\delta$)**.** $\mathcal{K}^\delta = \{(1-\lambda)x^* + \lambda x \mid x \in \mathcal{K}, \lambda \in [0,\delta]\}$ *The set $\mathcal{K}^\delta$ represents the set of points in the convex set scaled down by δ around the point x^*.*

4.1.2 Algorithm

Algorithm 1 Centroid Algorithm

```
1: procedure CentroidAlg($\nabla f, \mathcal{K}, T$)
2:     Initialize $\mathcal{K}_1 \leftarrow \mathcal{K}$
3:     for $t = 1$ to $T$ do
4:         $c_t \leftarrow$ centroid of $\mathcal{K}_t$
5:         $\mathcal{K}_{t+1} \leftarrow \mathcal{K} \cap \{y \mid \langle \nabla f(c_t), y - c_t \rangle \le 0\}$
6:     end for
7:     return $\min\{f(c_1), \cdots, f(c_T)\}$
8: end procedure
```

4.1.3 Example Visualization of the Centroid Algorithm for Two Iterations



4.1.4 Centroid Algorithm Runtime Analysis

Lemma 4.3 (Grünbaum's Lemma)**.** *$\forall$ convex sets $\mathcal{K}$, $\forall$ half space H passing through the centroid of $\mathcal{K}$*

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

Note that this bound is tight. Equality holds when the convex set is a circular cone in n-dimensions and the hyperplane that passes through the centroid of the cone is parallel to the cone's base.

Theorem 4.4 (Runtime of Centroid Algorithm)**.** *Let $f(x) : \mathcal{K} \to [-B, B]$, $B \geq 0$ and $x^* = \operatorname{argmin}_x f(x)$. If $\hat{x}$ is the result of the Centroid Algorithm after T timesteps,*

$$f(\hat{x}) - f(x^*) \leq 4B \cdot \exp\left(\frac{-T}{3n}\right)$$

Proof. Consider a $\mathcal{K}^\delta$. Pick any arbitrary $y \in \mathcal{K}^\delta$. Then, $y = (1-\lambda)x^* + \lambda x$ for some $x \in \mathcal{K}$ and $\lambda \in [0, \delta]$.

$$f(y) \leq (1-\lambda) f(x^*) + \lambda f(x) \leq f(x^*) + \lambda(f(x) - f(x^*)) \leq f(x^*) + 2\delta B$$

$$f(y) - f(x^*) \leq 2\delta B$$

Notice that

$$vol(\mathcal{K}^\delta) = \delta^n \cdot vol(\mathcal{K})$$

since $\mathcal{K}^\delta$ is $\mathcal{K}$ scaled down by a factor of δ with the center point of the scaling at x^* and the set lives n-dimensional space.

Next, Grünbaum's Lemma's tells us that in each iteration of the algorithm, the volume of the convex set $\mathcal{K}_t$ that is being maintained decreases by a factor of at least $(1 - \frac{1}{e})$. Thus,

$$vol(\mathcal{K}_t) \leq \left(1 - \frac{1}{e}\right)^t vol(\mathcal{K})$$

For $\delta = 2(1 - \frac{1}{e})^{\frac{T}{n}}$,

$$vol(\mathcal{K}^\delta) = 2^n \left(1 - \frac{1}{e}\right)^T vol(\mathcal{K})$$

and after T iterations of the algorithm, we will have

$$vol(\mathcal{K}_T) \leq \left(1 - \frac{1}{e}\right)^T vol(\mathcal{K}) \leq vol(\mathcal{K}^\delta)$$

Thus, sometime between iterations 1 and T, there will be a t such that $\mathcal{K}^\delta \subseteq \mathcal{K}_t$, but $\mathcal{K}^\delta \nsubseteq \mathcal{K}_{t+1}$.

This means that in iteration t, the algorithm eliminated points from $\mathcal{K}_t$ that were also in $\mathcal{K}^\delta$ to form $\mathcal{K}_{t+1}$. Denote the set of such points as $\mathcal{K}^\delta \cap (\mathcal{K}_t \setminus \mathcal{K}_{t+1})$.

Consider any point $y \in \mathcal{K}^\delta \cap (\mathcal{K}_t \setminus \mathcal{K}_{t+1})$. Since $y \in \mathcal{K}_t \setminus \mathcal{K}_{t+1}$ (the set of points that were removed in step t), $\langle \nabla f(c_t), y - c_t \rangle > 0$. By the definition of convexity, note that

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

$$\implies f(y) - f(c_t) \geq \langle \nabla f(c_t), y - c_t \rangle \geq 0$$

$$\implies f(c_t) \leq f(y)$$

The above statement shows that any point y eliminated in iteration t are less optimal than c_t. Combining this with the fact that $y \in \mathcal{K}^\delta$,

$$\begin{aligned} f(c_t) - f(x^*) &\leq 2B\delta \\ &= 2B\left(2\left(1 - \frac{1}{e}\right)^{\frac{T}{n}}\right) \\ &= 4B\left(1 - \frac{1}{e}\right)^{\frac{T}{n}} \\ &\leq 4B(e^{-\frac{1}{3}})^{\frac{T}{n}} \\ &= 4B \cdot \exp\left(\frac{-T}{3n}\right) \end{aligned}$$

Since the algorithm returns $\hat{x} = \text{argmin}_{1 \leq i \leq T} f(c_i)$ after T iterations,

$$f(\hat{x}) - f(x^*) \leq f(c_t) - f(x^*) \leq 4B \cdot \exp\left(\frac{-T}{3n}\right)$$

4.1.5 Comments About the Centroid Algorithm

1. The error of the algorithm for $T > 3n \ln \frac{4B}{\epsilon}$ iterations and $\epsilon < 1$ is

$$f(\hat{x}) - f(x^*) \leq \epsilon$$

which can be proved by plugging in $T \geq 3n \ln \frac{4B}{\epsilon}$ into Theorem 4.4. Thus, the Centroid Algorithm runs in $O(poly(log(\frac{1}{\epsilon}))$ time. However, the algorithm can be challenging to implement in practice since it is hard to maintain information about the convex set $\mathcal{K}_t$ (the algorithm must store all of the half-spaces $\{y \,|\, \langle \nabla f(c_t), y - c_t \rangle \leq 0\}$). Moreover, calculating the centroid of a general convex set can be difficult (sampling algorithms are often used). The Centroid Algorithm also assumes feasibility since it needs a point inside the feasible set to start running. This makes it not suitable for linear programming since determining feasibility for a linear program is nontrivial. Many of these problems are alleviated with the Ellipsoid Algorithm which will be presented below.

4.2 Ellipsoid Algorithm

4.2.1 Definitions

Definition 4.5 (Ball)**.** *Ball(C, R) is a ball centered at C with radius R represented by the set of points*

$$\{x \mid \|x - C\| \leq R\}$$

Definition 4.6 (Ellipsoid)**.** *An ellipsoid is an invertible linear transformation of the unit ball, Ball(0, 1). For the invertible linear transformation $L \in \mathbb{R}^{nxn}$, the ellipsoid is represented by the set*

$$\{Lx \mid x \in Ball(0, 1)\}$$

The ellipsoid corresponding to the transformation L centered at point c can be expressed as

$$E(c, Q) = \{x \mid (x - c)^T Q(x - c) \leq 1\}$$

where $Q = (LL^T)^{-1}$.

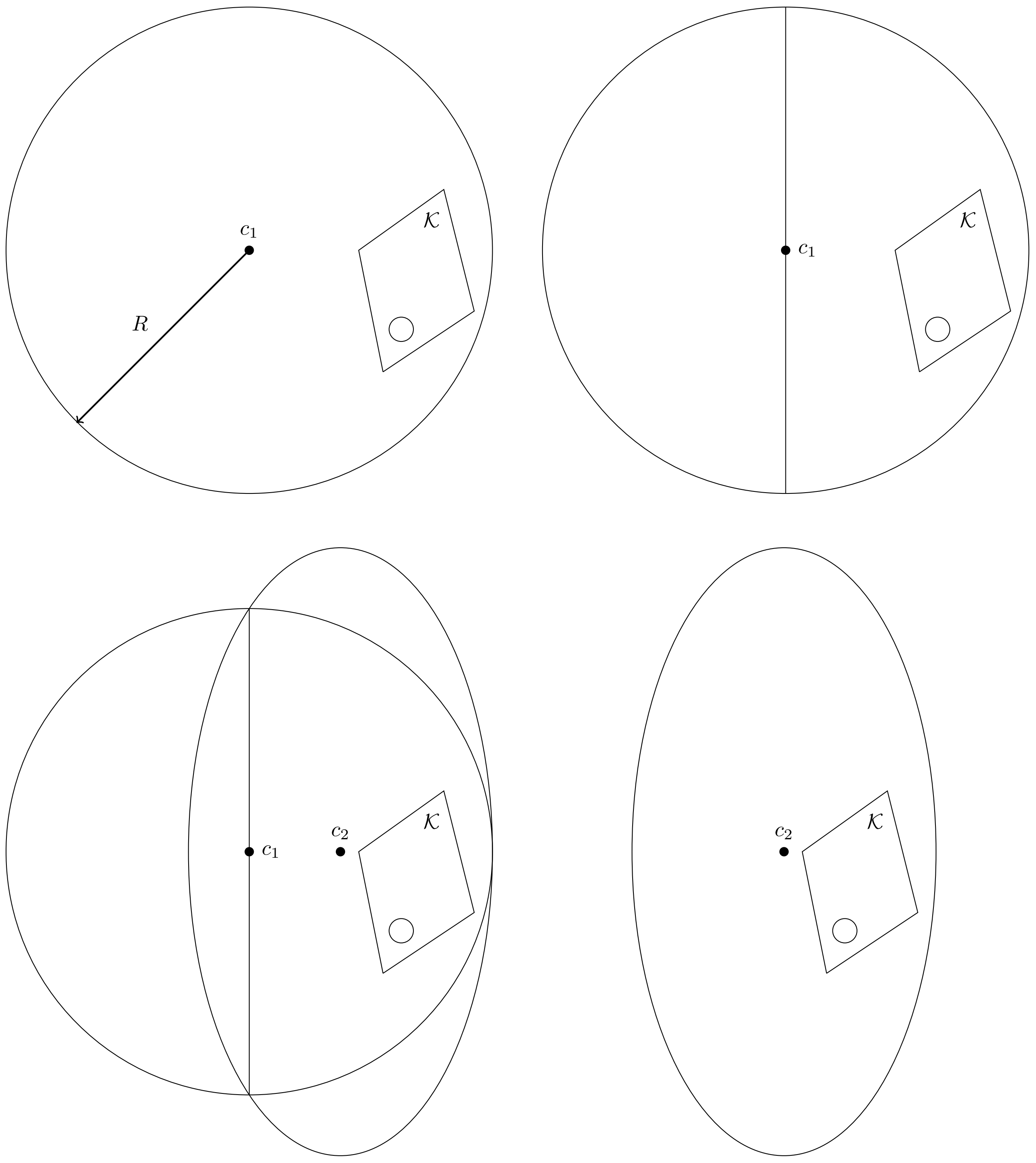
Definition 4.7 (Separation Oracle)**.** *A separation oracle for a convex set $\mathcal{K}$ is an algorithm that takes in a point $z \in \mathbb{R}^n$ and does one of the following:*
*1) If $z \in \mathcal{K}$: Output **Yes***
*2) If $z \notin \mathcal{K}$: Output **No** as well as a hyperplane separating z from $\mathcal{K}$, given by $a \in \mathbb{R}^n, b \in \mathbb{R}$ such that $\langle a, z\rangle < b$ and $\langle a, x\rangle \geq b \; \forall \, x \in \mathcal{K}$*

4.2.2 Algorithm

Algorithm 2 Ellipsoid Algorithm for LP Feasibility

```
1: procedure EllipsoidAlg(Separation Oracle, $R$)
2:     Initialize $(E_0, c_0) \leftarrow (\text{Ball}(0, R), 0)$
3:     for $t = 1$ to $2n(n+1)\ln(\frac{R}{r})$ do
4:         Ask Separation Oracle if $c_t \in \mathcal{K}$
5:         If Yes, return $c_t$
6:         If No, $P \leftarrow$ the separating hyperplane for $c_t$ and $\mathcal{K}$
7:         $H \leftarrow$ half space produced by P that contains $\mathcal{K}$
8:         $(E_{t+1}, c_{t+1}) \leftarrow$ (an ellipse that contains $E_t \cap H$, center of $E_{t+1}$)
9:     end for
10:    return Infeasible
11: end procedure
```

4.2.3 Example Visualization of the Ellipsoid Algorithm for One Iteration



4.2.4 Ellipsoid Algorithm Runtime Analysis

Lemma 4.8 (Linear Transformations and Volumes)**.** *$\forall A \subseteq \mathbb{R}^n$, if L is a linear transformation and $L(A)$ is the linear transformation of A where $L(A) = \{La \mid a \in A\}$, then $vol(L(A) = |det(L)|vol(A)$.*

Theorem 4.9 (Runtime of Ellipsoid Algorithm)**.** *Let $\mathcal{K}$ be a convex set that is feasible for the optimization problem. If $\exists\, c \in \mathbb{R}^n, r \in \mathbb{R}$ such that $Ball(c, r) \subseteq \mathcal{K}$ and $\exists\, R \in \mathbb{R}$ such that $\mathcal{K} \subseteq Ball(0, R)$, the Ellipsoid Algorithm can find a feasible point (if it exists) in $2(n+1)\ln\frac{R}{r}$ iterations.*

Proof. To find the ellipsoid E_{t+1} from E_t, we first use a linear transformation L^{-1} that maps the ellipsoid E_t to a $Ball(0, 1)$ and the halfspace of the ellipsoid that contains $\mathcal{K}$ into a half-ball H. One example of a simple half-ball is $H = \{x \mid x \in Ball(0, 1), x_1 \geq 0)\}$. Once we use this linear transformation, we can find a simple ellipsoid E_0 that contains the half-ball and satisfies the property that

$$\frac{vol(E_0)}{vol(Ball(0, 1))} \leq e^{-\frac{1}{2(n+1)}}$$

Now, we can apply the transformation L on E_0 to get E_{t+1} (notice that applying L to $Ball(0, 1)$ gives back E_t). By Lemma 4.8,

$$\frac{vol(E_{t+1})}{vol(E_t)} = \frac{|det(L)|vol(E_0)}{|det(L)|Ball(0, 1)} \leq e^{-\frac{1}{2(n+1)}}$$

Since the volume of the initial ball will decrease by at least a factor of $\frac{1}{e}$ after $2(n + 1)$ iterations, the volume decreases by at least a factor of $\frac{r}{R}$ after $2(n + 1)\ln\frac{R}{r}$ iterations. The algorithm terminates as soon as the separation oracle declares c_t to be in $\mathcal{K}$. If the problem is feasible, this happens once $vol(E_t) \leq vol(Ball(c, r))$ since at this point (which takes at most $2(n + 1)\ln\frac{R}{r}$ iterations), the ellipsoid must be fully contained in the ball that exists in the feasible set. Otherwise, none of the ellipsoid centers were in $\mathcal{K}$ and the algorithm finishes by returning that the problem was infeasible.

4.2.5 Using the Ellipsoid Algorithm To Solve Linear Programs

To see how the Ellipsoid Algorithm can solve linear programs, we will first focus on how the Ellipsoid Algorithm can solve linear program feasibility.

Given a linear program with the constraints $Ax \leq b$, denote the feasible set for the linear program as $\mathcal{K}$. Note that in order to run the algorithm, it must be the case that $\exists c \in \mathbb{R}^n, r \in \mathbb{R}$ such that $Ball(c, r) \subseteq \mathcal{K}$. If the previous condition is satisfied, one can input a separation oracle for $\mathcal{K}$ and a $R \in \mathbb{R}$ such that $\mathcal{K} \subseteq Ball(0, R)$ into the Ellipsoid Algorithm to solve for whether the linear program is feasible and a feasible point if one exists.

In each iteration of the Ellipsoid Algorithm, the separation oracle for the feasible set of the linear program will go through all of the constraints and will return "Yes" if the current centroid c_t meets all of the constraints. Otherwise, WLOG, say c_t does not satisfy the ith constraint, in order words, $a_i^T c_t > b_i$ where a_i^T, is the ith row of A and b_i is the ith element of b. The separation oracle will then return "No" along with the hyperplane $\{x \mid a_i^T x = b_i\}$. This will be a valid separating hyperplane since c_t satisfies $a_i^T c_t > b_i$ and all feasible points $y \in \mathcal{K}$ must satisfy the constraint $a_i^T y \leq b_i$. Continuing the process, the Ellipsoid Algorithm will eventually return a feasible point or determine that the feasible set is empty in at most $2n(n+1)ln(\frac{R}{r})$ iterations.

One can use the above algorithm that solves linear program feasibility to solve linear program optimization by adding constraints to the linear program that upper bound the objective function and using binary search on the upper bound to find the minimum of the objective function.

4.2.6 Comments About The Ellipsoid Algorithm

1. The Ellipsoid Algorithm is easier to implement than the Centroid Algorithm because we do not need to keep track of an arbitrary convex set but instead just the ellipsoid parameters c and Q.

2. The Ellipsoid Algorithm can be used to solve linear programs optimally in polynomial time in the number of variables and the bit complexity of the numbers returned by the separation oracle (the runtime does not depend on the number of constraints). Specifically, solving $\{\max c^T x \mid Ax \leq b\}$ has a runtime of poly$(n, \langle A \rangle, \langle b \rangle, \langle c \rangle)$, where n is the number of variables and $\langle A \rangle$, $\langle b \rangle$, and $\langle c \rangle$ are the number of bits it takes to represent A, b, and c respectively. If the number of operations were instead poly(x, y) where $A \in \mathbb{Q}^{x \times y}$, the algorithm would be called strongly polynomial time. However, the question of whether such an algorithm exists for solving linear programs is still open.

3. The simplex algorithm performs better in practice than the Ellipsoid Algorithm. However, the Ellipsoid Algorithm is a powerful tool in proving certain problems, especially those that involve linear programs, are solvable in polynomial time.