Optimization and Approximation ENS Lyon

Lecture 7

Lecturer: Alantha Newman *November 8, 2016*

1 Heuristics for Integer Programming

In the last lecture, we studied integer polytopes, which are polytopes whose vertices are integral. Two examples of integer polytopes are: (i) the perfect matching polytope in bipartite graphs, and (ii) the perfect matching polytope in general graphs. Often, polytopes of interest (e.g. relaxations of integer programs for NP-hard problems) are not integer polytopes. Nevertheless, we may still want to find optimal integer solutions. This is the first issue we address today. Another issue is that an integer polytope (such as the perfect matching polytope in general graphs) may have an exponential number of constraints. Thus, even though an optimal extreme point would yield an optimal integral solution, it is not practical to use the simplex algorithm to find such a solution. The second issue we address today is how to handle linear programs with exponentially many constraints. Also, recall that even for a linear program with polynomially many constraints, the simplex algorithm is not an efficient algorithm in the worst case. The third issue we address is efficiently finding solutions for linear programs.

1.1 Branch and Bound

Suppose we would like to solve the following integer program.

$$\begin{aligned} \max\ & \mathbf{c}^\intercal \mathbf{x} \\ \text{subject to: } & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \text{ and } \mathbf{x} \text{ integer.} \end{aligned} \tag{P}$$

A heuristic known as *branch and bound* is outlined below. The value of the global variable τ is equal to the value of the best (largest) integer solution found so far. Initially, τ is set to some (possibly very poor) lower bound on the value of an optimal solution for (P). For example, we can initially set τ to be a negative number with huge absolute value. Here, (Q) will denote a specified set of linear constraints. If our goal is to solve (P), then we initially set (P') to include all the constraints from (P) except the integrality constraints and invoke the procedure Branch-and-Bound(P').

```
Branch-and-Bound(Q)

1. Solve (Q) (e.g. use the simplex algorithm).
2. If solution $\mathbf{c}^\intercal\mathbf{x}^* > \tau$:
   (a) If $\mathbf{x}^* \in \mathbb{Z}^n$, then update $\tau$ and $\mathbf{z}^*$, i.e. set $\tau := \mathbf{c}^\intercal\mathbf{x}^*$ and $\mathbf{z}^* := \mathbf{x}^*$.
   (b) Otherwise, choose some coordinate $x_i^* \notin \mathbb{Z}$ and recurse on two subproblems:
       i. Set (Q) := $\{$(Q) and $x_i \leq \lfloor x_i^* \rfloor\}$ and solve Branch-and-Bound(Q).
       ii. Set (Q) := $\{$(Q) and $x_i \leq \lceil x_i^* \rceil\}$ and solve Branch-and-Bound(Q).
3. Otherwise, if $\mathbf{c}^\intercal\mathbf{x}^* \leq \tau$, do nothing (i.e. "prune" or ignore corresponding subtree).
```

When Branch-and-Bound(P') terminates, the variable $\mathbf{z}^*$ will be an optimal integral solution with objective value τ. This procedure simulates a tree traversal, which can be conducted in a depth-first or breadth-first manner. In other words, at Step 2(b), one can instead place the two new subproblems on a queue and then choose a new subproblem from the queue according to some specified criteria. The way the procedure is actually written above corresponds to a depth-first search of the solution tree, and it has been claimed that depth-first search works well in practice [Van01]. What exactly does "works well in practice" mean here? If one can quickly find an integer solution with a large—albeit possibly not optimal—objective value, then many subtrees can be pruned or skipped resulting in a faster running time.

1.2 Cutting Planes

The problem of finding a *minimum cost arborescence* is defined as follows. Given a strongly connected directed graph $G = (V, A)$, a specified root vertex $r \in V$ and a nonnegative cost function on the arcs $c : A \to \mathbb{R}^+$, the goal is to find a subset $T \subseteq A$ of arcs that contains a directed path from r to v for all $v \in V$, where $v \neq r$. We have the following linear programming relaxation for this problem. For a nonempty subset $S \subset V$, let $\delta^+(S)$ denote all edges directed out of S. Let $m = |A|$.

$$\begin{aligned} \min \sum_{e \in A} c_e x_e & \\ x(\delta^+(S)) \geq 1, &\quad \text{for all } S \subset V \text{ such that } r \in S, \\ x_e \geq 0, &\quad \text{for each } e \in A. \end{aligned} \tag{P$_{arbre}$}$$

Like the perfect matching polytope for general graphs, the linear program (P$_{arbre}$) (i) is described by an exponential number of constraints, and (ii) has integer vertices. (We will not prove (ii) now.) How can we find an optimal extreme point for (P$_{arbre}$)? We will show in Section 3 that we can reduce the problem of finding an extreme point with an *optimal* objective value to the problem of finding any extreme point. Now we give a heuristic for finding an extreme point.

```
CUTTING PLANE(P_arbre)

  Initialize (Q) to be the set of nonnegativity constraints from (P_arbre).

  1. Solve the linear program (Q) to obtain an extreme point $\mathbf{x}$ (e.g. use the simplex algorithm).

  2. If $\mathbf{x}$ is feasible for (P_arbre): terminate and output $\mathbf{x}$.

  3. Otherwise, if $\mathbf{x}$ is infeasible for (P_arbre):

          Find a cut $S \ni r$ whose corresponding constraint in (P_arbre) is violated by $\mathbf{x}$.

  4. Add this constraint to (Q) and go to Step 1.
```

To complete the description of this procedure, we need to show how to determine if a point $\mathbf{x} \in \mathbb{R}^m$ belongs to (P$_{arbre}$). Moreover, if $\mathbf{x}$ does *not* belong to (P$_{arbre}$), we need to show how to find a violated constraint in (P$_{arbre}$). An algorithm for determining membership of a point in a polyhedron is called a *membership oracle*. An algorithm for finding a violated constraint for a point that does not belong to a polyhedron is called a *separation oracle*.

Given a point $\mathbf{x} \in \mathbb{R}^m$, how can we determine whether or not $\mathbf{x}$ belongs to (P$_{arbre}$)? We can easily verify that each coordinate of $\mathbf{x}$ is nonnegative. To determine whether or not the cut constraints are satisfied, we can apply an algorithm for finding a minimum s-t cut, where $s = r$ and $t = v$ for all $v \in V \setminus r$. If each one of these $n - 1$ minimum cuts has value at least one, then we can ascertain that $\mathbf{x}$ belongs to (P$_{arbre}$). However, if there is some minimum cut whose value is strictly *less than* one, then this cut corresponds to a violated cut constraint. As is often the case for the problems we consider, a single algorithm serves as both a membership and separation oracle. We discuss separation oracles further in the next section.

The cutting plane approach outlined above to find an extreme point for (P$_{arbre}$) can be applied to find an extreme point in any polyhedron described by an exponential number of constraints, as long as we have a separation oracle for this set of constraints. Let (Q) denote such a polyhedron and initially set (Q') to be some (polynomial-sized) subset of the constraints describing (Q).

```
Cutting Plane(Q)

  1. Solve the linear program (Q') to obtain an extreme point $\mathbf{x}$ (e.g. use the simplex algorithm).
  2. If $\mathbf{x}$ is feasible for (Q): terminate and output $\mathbf{x}$.
  3. Otherwise, if $\mathbf{x}$ is infeasible for (Q):
         Find a constraint in (Q) that is violated by $\mathbf{x}$.
  4. Add this constraint to (Q') and goto Step 1.
```

In the worst case, this procedure may result in a linear program with exponentially many constraints. However, in practice, it is often the case that the final linear program contains only a small subset of all possible constraints.

2 More Examples of Separation Oracles

We now look at two other problems whose linear programs have exponential size and show that they have efficient separation oracles. Among other things, this implies that the cutting plane procedure from the previous section can be applied to find extreme points for these linear programs.

Given a directed graph $G = (V, A)$, the *asymmetric traveling salesman* problem is defined as follows. Find a connected, spanning Eulerian multigraph of G. This problem is NP-hard and therefore the following linear programming relaxation (P$_{ATSP}$) cannot be an integer polyhedron. Nevertheless, the cutting plane procedure from the previous section can be used to find a (possibly non-integral) extreme point for this linear program. For a nonempty subset $S \subset V$, let $\delta^+(S)$ denote all edges directed out of S and let $\delta^-(S)$ denote all edges directed into S. Let $m = |A|$.

$$\begin{aligned} \min \sum_{e \in A} & x_e \\ x(\delta^+(v)) = {} & x(\delta^-(v)), \quad \text{for all } v \in V, \\ x(\delta^+(S)) \geq 1, & \quad \text{for all } S \subset V \text{ such that } S \neq \emptyset, \\ x_e \geq 0, & \quad \text{for each } e \in A. \end{aligned} \tag{P$_{ATSP}$}$$

Given an $\mathbf{x} \in \mathbb{R}^m$, we wish to determine if $\mathbf{x} \in$(P$_{ATSP}$) and if not, we want to find a violated constraint. We can easily check that the degree constraints and the nonnegativity constraints are respected. How can we check the cut constraints? We can essentially use the same algorithm as we used for the minimum cost arborescence problem: for each value of $s, t \in V$ where $s \neq t$, we run an algorithm for the minimum s-t cut problem. If the minimum cut has value at least one, we can conclude that $\mathbf{x}$ belongs to (P$_{ATSP}$). Otherwise, we have found a violated cut constraint.

2.1 Minimum Odd Cuts

Now we will show that there is an efficient separation oracle for the perfect matching polytope on general graphs. Let $G = (V, E)$, let $m = |E|$ and let $n = |V|$. The perfect matching polytope on G was defined in the last lecture.

$$\begin{aligned} \sum_{e \in \delta(v)} x_e = 1, &\quad \text{for each } v \in V, \\ x(\delta(U)) \geq 1, &\quad \text{for each } U \subset V,\ |U| \text{ odd},\ |U| \geq 3, \\ x_e \geq 0, &\quad \text{for each } e \in E. \end{aligned} \tag{Q$^{\text{PM}}$}$$

Given a point $\mathbf{x} \in \mathbb{R}^m$, we can efficiently check the degree and the nonnegativity constraints. But to determine if $\mathbf{x}$ belongs to (Q$^{\text{PM}}$), we need to verify that all *odd* cuts of at least three vertices have value at least one. This is similar to the separation oracles we have already seen except that here we need an algorithm to find a minimum *odd* cut rather than a minimum cut.

Now we consider the following optimization problem: Given a graph $G = (V, E)$ where each edge $e \in E$ has a value $y_e \geq 0$, find a nonempty subset of the vertices $U \subset V$ such that $|U|$ is odd and the edge weight crossing the cut $y(U, V \setminus U)$ is minimum. We will show that this problem can be reduced to the minimum cut problem.

For a subset of vertices $S \subset V$, let $\bar{S} = V \setminus S$. Let $(A, \bar{A})$ denote a minimum odd cut in G, and let $(B, \bar{B})$ denote a minimum cut in G that we have found (via a minimum cut algorithm). Clearly, if $|B|$ or $|\bar{B}|$ is odd, then we have found a minimum odd cut in G and we are finished. Suppose that neither $|B|$ nor $|\bar{B}|$ is odd. Then we claim that we can recurse on B (and $\bar{B}$), i.e. find a minimum cut on the graph induced by the vertices in B. If $A \subset B$ or $A \subset \bar{B}$, or if $\bar{A} \subset B$ or $\bar{A} \subset \bar{B}$, then we can still find the minimum odd cut by this recursion. However, it may be the case that $A \not\subset B$ and $A \not\subset \bar{B}$. Additionally, assume that $\bar{A} \not\subset B$ and $\bar{A} \not\subset \bar{B}$. Then we have the following:

$$A \cap B \neq \emptyset, \quad A \cap \bar{B} \neq \emptyset, \quad \bar{A} \cap B \neq \emptyset, \quad \bar{A} \cap \bar{B} \neq \emptyset.$$

In this case, we apply a tool called *submodularity.* A function $f(\cdot)$ is submodular if the following holds:

$$f(A) + f(B) \quad \geq \quad f(A \cap B) + f(A \cup B).$$

Let $g(\cdot)$ denote the cut function, i.e. $g(S)$ denotes the value of the edges crossing the cut $(S, \bar{S})$. It can be shown that the cut function is submodular (we leave this as an exercise). Since $|A|$ is odd, either $|A \cap B|$ or $|A \cap \bar{B}|$ is odd. Let us suppose that $|A \cap B|$ is odd. Then we have:

$$\begin{aligned} g(A) + g(B) &\geq g(A \cap B) + g(A \cup B) \\ &= g(A \cap B) + g(\bar{A} \cap \bar{B}) \\ &\geq g(A) + g(B), \end{aligned}$$

since $g(A \cap B)$ is at least as large as the minimum odd cut and $g(\bar{A} \cap \bar{B})$ is at least as large as the minimum cut. It follows that:

$$g(A) = g(A \cap B) \text{ and } g(B) = g(\bar{A} \cap \bar{B}),$$

and the set B contains a minimum odd cut. If $|A \cap \bar{B}|$ is odd, then an analogous argument can be made showing that $\bar{B}$ contains a minimum odd cut. The number of times we need to execute the minimum cut algorithm can be shown to be at most $O(n^2)$. This shows that there is an efficient separation oracle for (Q$^{\text{PM}}$).

2.2 Separating Hyperplanes

The concept of a separation oracle is closely related to that of a *separating hyperplane.*

Theorem 1. [Separating Hyperplane Theorem] *Let K be a non-empty closed convex subset of $\mathbb{R}^n$ and let $\mathbf{x}^* \in \mathbb{R}^n$ be a point such that $\mathbf{x}^* \notin K$. Then there exists some hyperplane $\mathbf{a}^\intercal \mathbf{y} = h$ such that $\mathbf{a}^\intercal \mathbf{x}^* \leq h$ and $\mathbf{a}^\intercal \mathbf{x} > h$ for all $\mathbf{x} \in K$.*

If K is a polyhedron (i.e. described by a set of linear inequalities), and we have a separation oracle for K, then this immediately yields a separating hyperplane for a point $\mathbf{x}^* \notin K$. Specifically, the violated constraint found by the separation oracle can be converted into a separating hyperplane. For example, let $K = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}\}$ and let $\mathbf{a}_j$ denote the j^{th} constraint (i.e. the j^{th} row in the matrix $\mathbf{A}$). For some $\mathbf{x}^* \in \mathbb{R}^n$, suppose there is a constraint such that $\mathbf{a}_j^\intercal \mathbf{x}^* < b_j$ but $\mathbf{a}_j^\intercal \mathbf{x} \geq b_j$ for all $\mathbf{x} \in K$. Then setting $h = \mathbf{a}_j^\intercal \mathbf{x}^*$, we observe that $\mathbf{a}_j^\intercal \mathbf{y} = h$ is a separating hyperplane. (Notice that there are other ways to obtain a separating hyperplane from a violated constraint. We use this convention because the resulting hyperplane has the additional property that it goes "through" the point $\mathbf{x}^*$, which—as we will see in Section 4—is a useful property.)

3 Reducing Optimization to Feasibility

In general, the cutting plane procedure is not a provably efficient method for finding an extreme point in a given polyhedron. Thus, our next objective is to present a provably efficient algorithm for this task. Before we do so, we show why an algorithm that finds an extreme point can be used to solve an optimization problem, such as a linear program, in which the goal is to find an *optimal* extreme point.

Suppose we are given a closed convex set K (e.g. a polytope in $\mathbb{R}^n$) and the following optimization problem:

$$\begin{aligned} \max\ & \mathbf{c}^\intercal \mathbf{x} \\ & \mathbf{x} \in K \end{aligned}$$

Solving this optimization problem is equivalent to finding *any* point $\mathbf{x}$ in the convex set K, because we can add another constraint so that the problem becomes:

$$\text{Find } \mathbf{x} \text{ such that } \mathbf{x} \in K \text{ and } \mathbf{c}^\intercal \mathbf{x} \geq t.$$

By conducting a binary search on t, we can find $\mathbf{x} \in K$ such that $\mathbf{c}^\intercal \mathbf{x}$ is maximized. Suppose the maximum value is $\tau = \mathbf{c}^\intercal \mathbf{z}$ attained for some $\mathbf{z} \in K$. Then finding a point with value τ requires at most $\log |\tau|$ iterations of an algorithm that finds a point $\mathbf{x}$ in a given convex set. Moreover, if we want to find an extreme point with value τ, it is sufficient to find a feasible point with value τ, because there always exists an extreme point with the optimal value (see Theorem 11, Lecture 2). So although the simplex algorithm has the nice feature that it always returns an extreme point, it is sufficient to return a feasible point in order to solve a linear program.

4 Finding a Point in a Convex Set

Given a convex set $K \in \mathbb{R}^n$, our goal is now to find a point $\mathbf{x} \in K$. We can solve this problem efficiently if we are given an efficient separation oracle for K. That is, for some $\mathbf{x}^* \in \mathbb{R}^n$, the separation oracle returns "yes" if $\mathbf{x}^* \in K$ and "no" if $\mathbf{x}^* \notin K$. Additionally, if $\mathbf{x}^* \notin K$, the separation oracle provides a separating hyperplane $\mathbf{a}^\intercal \mathbf{y} = h$ such that $\mathbf{a}^\intercal \mathbf{x}^* \leq h$ and $\mathbf{a}^\intercal \mathbf{x} > h$ for all $\mathbf{x} \in K$. One method to find a point in a convex set K (or determine that K is empty) where K is equipped with such a separation oracle is known as the *ellipsoid algorithm.*

4.1 Ellipsoid Algorithm

We have some convex set $K \in \mathbb{R}^n$ in space—we don't know exactly where it is. The idea behind the ellipsoid algorithm is to locate the set by enclosing it in progressively smaller and smaller ellipsoids. We let $Ell(\mathbf{D}, \mathbf{z})$ denote the ellipsoid defined by a positive definite matrix $\mathbf{D}$ and center $\mathbf{z}$. Formally,

$$Ell(\mathbf{D}, \mathbf{z}) = \{\mathbf{x} \mid (\mathbf{x} - \mathbf{z})'\mathbf{D}^{-1}(\mathbf{x} - \mathbf{z}) \leq 1\}.$$

As the first step in the algorithm, we choose radius R large enough so that $E_0 = Ell(R^2\mathbf{I}, 0)$ contains some part of the convex set. (For simplicity, let us assume that the convex set K is entirely contained in E_0.) Now we ask: is the origin feasible? If yes, we are done. If no, we provide a separating hyperplane through the origin and we proceed by finding the smallest (volume) ellipsoid containing the half ellipsoid bounded by E_0 and the separating hyperplane. On the i^{th} iteration of the algorithm, we let $E_i = Ell(\mathbf{D}_i, \mathbf{z}_i)$. We ask, is $\mathbf{z}_i \in K$? If yes, we are done. If no, we return a hyperplane $\mathbf{a}^\intercal\mathbf{y} = h$ $(= \mathbf{a}^\intercal\mathbf{z}_i)$, which goes through the point $\mathbf{z}_i$, such that $\mathbf{a}^\intercal\mathbf{z}_i \leq h$ and $\mathbf{a}^\intercal\mathbf{x} > h$ for all $\mathbf{x} \in K$. We then let E_{i+1} be the minimum volume ellipsoid that contains $E_i \cap \{\mathbf{x} \mid \mathbf{a}^\intercal\mathbf{x} > \mathbf{a}^\intercal\mathbf{z}_i\}$. Since we assume by induction that E_i contains K and K is on the $>$ side of the hyperplane, the new ellipsoid E_{i+1} will also contain K. Although it is not known how to compute the minimum volume ellipsoid containing a given convex set in general, this computation is trivial if the set is an ellipsoid. Furthermore, if the set is a half-ellipsoid—as is the case here—the problem is still easy. We now summarize the algorithm.

```
ELLIPSOID($K$)

  1. Set $E_0 := Ell(R^2\mathbf{I}, \mathbf{z}_0)$, where $\mathbf{z}_0 = 0$.
  2. For $i = 0$ to $t^*$:
       (a) If $\mathbf{z}_i \in K$, terminate and output $\mathbf{z}_i$.
       (b) Otherwise, find a constraint $\mathbf{a}_j^\intercal\mathbf{x} \geq b_j$ in $K$ violated by $\mathbf{z}_i$.
             i. Set $h = \mathbf{a}_j^\intercal\mathbf{z}_i$.
            ii. Let $E_{i+1}$ be the smallest volume ellipsoid containing $E_i \cap \{\mathbf{x} \mid \mathbf{a}_j^\intercal\mathbf{x} \geq h\}$.
           iii. Let $z_{i+1}$ be the point at the center of $E_{i+1}$.
  3. Terminate and declare $K = \emptyset$.
```

Are the ellipsoids getting smaller at each iteration? Is it clear that we are gradually approaching the target set K? What if the convex set in question is so tiny that it takes exponentially many iterations to find $\mathbf{x} \in K$? These questions are all related to the complexity of the ellipsoid algorithm, which we now address. The first crucial observation is that in fact the size of the ellipsoid always drops from one iteration to the next. That is,

$$\frac{Vol(E_{i+1})}{Vol(E_i)} \quad < \quad 1.$$

Let us now see by exactly how much we require the volume to drop from one iteration to the next. Define Γ such that $Vol(E_0) \leq \Gamma$. We claim that if K is nonempty, then K has volume at least γ. (We assume that K is full-dimensional and we will give bounds on Γ and γ in terms of n and the size of the input K shortly.) The number of iterations we run the algorithm for is:

$$t^* \quad = \quad 2(n+1)\log\left(\frac{\Gamma}{\gamma}\right).$$

Suppose that:

$$\frac{Vol(E_{i+1})}{Vol(E_i)} \quad < \quad e^{-\frac{1}{2(n+1)}}. \tag{1}$$

Then we have:

$$\frac{Vol(E_{t^*})}{Vol(E_0)} \quad < \quad e^{-\frac{t^*}{2(n+1)}} \quad = \quad e^{-\log\left(\frac{\Gamma}{\gamma}\right)} \quad = \quad \frac{\gamma}{\Gamma}, \tag{2}$$

which implies that $Vol(E_{t^*}) < \gamma$. This is a contradiction if K is nonempty, since E_{t^*} should contain K (as should all ellipsoids E_i) and K has volume at least γ by assumption. Now we show that (1) does in fact hold.

Lemma 2. (Key Lemma) *Let $E' = Ell(\mathbf{D}', \mathbf{z}')$ be the minimum volume ellipsoid containing $Ell(\mathbf{D}, \mathbf{z}) \cap \{\mathbf{x} \mid \mathbf{a}^\intercal \mathbf{x} \geq \mathbf{a}^\intercal \mathbf{z}\}$. Then,*

$$\frac{Vol(E')}{Vol(E)} \quad < \quad e^{-\frac{1}{2(n+1)}}.$$

Proof. First, we have a few computational details:

$$\mathbf{z}' = \mathbf{z} - \frac{1}{n+1}\frac{\mathbf{Da}}{\sqrt{\mathbf{a}^\intercal \mathbf{Da}}}, \qquad \mathbf{D}' = \frac{n^2}{n^2-1}\left(\mathbf{D} - \frac{2}{n+1}\frac{\mathbf{Daa}^\intercal\mathbf{D}}{\mathbf{a}^\intercal\mathbf{Da}}\right).$$

Every ellipsoid is a linear transformation of a ball, i.e. $Ell(\mathbf{D}, \mathbf{z}) = \mathbf{D}^{1/2}E(\mathbf{I}, 0) + \mathbf{z}$, where $\mathbf{D} = \mathbf{B}^T\mathbf{B}$ because $\mathbf{D}$ is positive definite implying $\mathbf{D}^{1/2} = \mathbf{B}$. In addition, suppose that P is a convex set and B_n is the n-dimensional unit ball. Let $\mathbf{T}$ be an affine transformation of P that maps $\mathbf{x} \in P$ to $\mathbf{T}(\mathbf{x}) = \mathbf{L}(\mathbf{x}) + \mathbf{d}$, where $\mathbf{d} \in \mathbb{R}^n$. Then we have,

$$\frac{Vol(\mathbf{T}(P))}{Vol(P)} = \frac{Vol(\mathbf{L}(P))}{Vol(P)} = \det(\mathbf{L}), \qquad Vol(Ell(\mathbf{D}, \mathbf{z})) = \sqrt{\det(\mathbf{D})}Vol(B_n).$$

Specifically, let $\mathbf{T}(\cdot)$ denote the linear transformation that maps the original ellipsoid E to the unit ball. We claim that the ratio of $Vol(E)$ to $Vol(E')$ is the same after this transformation. Thus, it is enough to consider $D = Ell(\mathbf{I}, 0)$. In this case, we have,

$$\mathbf{z}' = -\frac{1}{(n+1)}\frac{\mathbf{a}}{|\mathbf{a}|}, \qquad \mathbf{D}' = \frac{n^2}{n^2-1}\left(\mathbf{I} - \frac{2}{(n+1)}\frac{\mathbf{aa}^\intercal}{|\mathbf{a}|}\right).$$

The vector $\mathbf{a}$ is the normal vector to the separating hyperplane. For the analysis, we can assume that $|\mathbf{a}| = 1$ (since this does not change the hyperplane). Furthermore, we can assume that $\mathbf{a} = (1, 0, \dots)$, since this leads to the worst case for the following calculation. Thus, we have,

$$\mathbf{z}' = (-\frac{1}{n+1}, 0, \dots), \qquad \mathbf{D}' = \frac{n^2}{n^2-1}\begin{pmatrix} 1 - \frac{2}{n+1} & & 0 \\ & 1 & \\ & & \dots \\ 0 & & \dots \end{pmatrix}.$$

In other words, the center is only shifting along $\mathbf{a}$. If we use calculus to determine the minimum volume, we have,

$$\frac{Vol(E')}{Vol(E)} = \frac{\sqrt{\det \mathbf{D}'}}{\sqrt{\det \mathbf{D}}} = \frac{\sqrt{\det \mathbf{D}'}}{1}.$$

Now we will compute $\sqrt{\det \mathbf{D}'}$.

$$\sqrt{\det \mathbf{D}'} = \left(\frac{n^2}{n^2-1}\right)^{\frac{n}{2}}\left(\frac{n-1}{n+1}\right)^{\frac{1}{2}} = \left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}}\frac{n}{n+1} \leq e^{\frac{1}{n^2-1}\frac{n-1}{2}}e^{-\frac{1}{n+1}} = e^{\frac{1}{2n+2}-\frac{1}{n+1}} < e^{-\frac{1}{2n+2}}.$$

□

To complete the analysis of the ellipsoid algorithm, we need to argue that $\log \frac{\Gamma}{\gamma}$ is polynomial in the size of the input. Given a polyhedron of form $K = \{x \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}\}$, let U be the largest absolute value of the entries in $\mathbf{A}$ and $\mathbf{b}$, which are assumed to be integral (via scaling of rational numbers). It can be shown that E_0, which is a ball of radius R, will contain K if $R = (nU)^n$. This implies that the volume of E_0 is at most $(2R)^n = 2^n(nU)^{n^2}$ Furthermore, it can be shown that K has volume at least $\frac{1}{(nU)^{n^3}}$. (We omit the proofs, which are tedious, but refer the reader to Chapter 8 of [BT97] for a full technical discussion.) Thus, we have:

$$\log\left(\frac{\Gamma}{\gamma}\right) \;=\; \log 2^n (nU)^{n^2+n^3} \;=\; O(n^3 \log U).$$

We conclude that the ellipsoid algorithm makes at most $O(n^4 \log U)$ calls to the separation oracle for K.

4.2 Centroid-Cutting Algorithm

We briefly discuss another method for efficiently solving a linear program, i.e. finding a point in a given polyhedron. At a high level, this method resembles the ellipsoid algorithm, but in several key ways, it is much simpler. First, we present the following theorem due to Grünbaum. The centroid of a convex body K is defined as the average value of all the points in K.

Theorem 3. [Grü60] *Any halfspace that contains the centroid of K contains at least a $\frac{1}{e}$-fraction of the volume of K.*

Let C_0 be a large, axis-aligned cube centered at the origin that is guaranteed to contain K if K is nonempty (and full-dimensional). Define Γ such that $Vol(C_0) \leq \Gamma$. We test the centroid of C_0 (which is the origin) and see if it belongs to K. If so, we are done. If not, we find a separating hyperplane and construct a new convex body (possibly no longer a cube) that is the intersection of the current convex body (i.e. C_0, initially) with this separating hyperplane, and repeat. We summarize this procedure here. Let $t^* = \log\left(\frac{\Gamma}{\gamma}\right)$. Recall that γ is a lower bound on the volume of K.

```
CUTTING-CENTROID($K$)

  1. Let $C_0$ be an axis-aligned cube that contains $K$ with center $\mathbf{z}_0 = 0$.
  2. For $i = 0$ to $t^*$:
     (a) If $\mathbf{z}_i \in K$, terminate and output $\mathbf{z}_i$.
     (b) Otherwise, find a constraint $\mathbf{a}_j^\intercal \mathbf{x} \geq b_j$ in $K$ violated by $\mathbf{z}_i$.
           i. Set $h = \mathbf{a}_j^\intercal \mathbf{z}_i$.
          ii. Let $C_{i+1} = C_i \cap \{\mathbf{x} \mid \mathbf{a}_j^\intercal \mathbf{x} \geq h\}$.
         iii. Let $\mathbf{z}_{i+1}$ be the point at the center of $C_{i+1}$.
  3. Terminate and declare $K = \emptyset$.
```

Given an efficient subroutine for computing the centroid of a convex body, this procedure is efficient. In other words, using the same reasoning as in Equation (2), we see that Theorem 3 implies:

$$\frac{Vol(C_{t^*})}{Vol(C_0)} \;<\; e^{-t^*} \;=\; e^{-\log\left(\frac{\Gamma}{\gamma}\right)} \;=\; \frac{\gamma}{\Gamma},$$

which is a contradiction if K is nonempty. However, the problem of finding the centroid of a convex body is computationally intractable. But we can modify the algorithm to use an efficient procedure for computing an *approximate* centroid. The efficiency of the modified algorithm relies on the following theorem.

Theorem 4. **[BV04]** *Let K be a convex body in $\mathbb{R}^n$ and let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ denote $N = O(n)$ points in K, each chosen uniformly at random. Define $\mathbf{z} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$. Then with high probability, any halfspace that contains $\mathbf{z}$ contains at least a $\frac{1}{3}$-fraction of the volume of K.*

Theorem 4 shows that an approximate centroid can be generated with high probability, using subroutines for generating random points in polytopes, which is a well-studied problem and can be done efficiently. Using Theorem 4, we obtain the following procedure.

```
CUTTING-RANDOM-CENTROID($K$)

  1. Let $C_0$ be axis-aligned cube that contains $K$ with center $\mathbf{z}_0 = 0$.
  2. For $i = 0$ to $t^*$:
     (a) If $\mathbf{z}_i \in K$, terminate and output $\mathbf{z}_i$.
     (b) Otherwise, find a constraint $\mathbf{a}_j^\intercal \mathbf{x} \geq b_j$ in $K$ violated by $\mathbf{z}_i$.
           i. Set $h = \mathbf{a}_j^\intercal \mathbf{z}_i$.
          ii. Let $C_{i+1} = C_i \cap \{\mathbf{x} \mid \mathbf{a}_j^\intercal \mathbf{x} \geq h\}$.
         iii. Generate $N = O(n)$ random points $\mathbf{x}_1, \mathbf{x}_, \ldots, \mathbf{x}_N$ in $C_{i+1}$.
          iv. Let $\mathbf{z}_{i+1} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$.
  3. Terminate and declare $K = \emptyset$.
```

This procedure can be viewed as a reduction from the problem of finding a point in a convex body to the problem of finding an approximate centroid of a convex body. For more on the complexity of the latter problem, see [Rad07].

In the ellipsoid algorithm, each iteration required a non-trivial computation of a new ellipsoid, but determining the center of this ellipsoid (i.e. the point to test for feasibility in K) was trivial. In the above procedure, computing the next convex body is trivial, but determining the approximate centroid requires some sophisticated machinery such as random walks to generate random points.

References

[BT97] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997.

[BV04] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *Journal of the ACM*, 51(4):540–556, 2004.

[Grü60] Branko Grünbaum. Partitions of mass-distributions and of convex bodies by hyperplanes. *Pacific Journal of Mathematics*, 10(4):1257–1261, 1960.

[Kha80] Leonid G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[Lev65] A. Yu. Levin. On an algorithm for the minimization of convex functions. *Soviet Mathematics Doklady*, 6:286–290, 1965.

[Rad07] Luis A. Rademacher. Approximating the centroid is hard. In *Proceedings of the twenty-third Annual Symposium on Computational Geometry*, pages 302–305. ACM, 2007.

[Sho72] Naum Z. Shor. Utilization of the operation of space dilatation in the minimization of convex functions. *Cybernetics and Systems Analysis*, 6(1):7–15, 1972.

[Van01] Robert J. Vanderbei. *Linear programming: Foundations and Extensions.* Kluwer Academic Publishers, Boston, MA, 2001.

[YN76] David B. Yudin and Arkadii S. Nemirovski. Evaluation of the information complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody*, 12:128–142, 1976.

These lectures notes are based in part on Chapter 22 of [Van01], Chapter 8 of [BT97] and on scribe notes for the ellipsoid algorithm from a class taught by Santosh Vempala. The ellipsoid method was developed in the works of Shor [Sho72] and Yudin and Nemirovskii [YN76]. The analysis showing that it can be implemented in polynomial time is due to Khachiyan [Kha80]. The algorithms in Section 4.2 are due to Levin [Lev65] and to Bertsimas and Vempala [BV04].