Geometry of LPs and the Simplex Algorithm

Recall: A polyhedral set is defined as the intersection of a finite number of halfspaces; $P = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$

Definitions:

Vertex: $x \in P$ s.t. $x - y \in P$ and $x + y \in P \Rightarrow y = 0$ (ie not a convex combination of two other points)

Intuitively, these are the corners of the polyhedron. So you cannot move forward and backward along any direction.

Basic feasible solution: $x \in P$ s.t. at least n linearly independent constraints are tight.

Lemma: Vertices and basic feasible solutions are equivalent.

Proof: $\Rightarrow$ Consider subspace defined by $\leq n-1$ equality constraints. Subspace has dim. ≥ 1. Therefore, we can move in both directions on some line.

$\Leftarrow$ Suppose you could move in both directions along some line going through the basic feasible solution. Let the difference vector be y and $-y$. For every tight constraint a_i, $a_i \cdot y$ must be 0. But the a_i's form a full rank matrix; thus $y = 0$.

Theorem: Any bounded LP in standard form has an optimum at a basic feasible solution.

Proof: Suppose not; we will show that the number of tight constraints can be increased without changing the objective. Consider the subspace given by the tight constraints — this has dimension ≥ 1. Thus $\exists$ vector d s.t. $a_i^T \cdot d = 0 \ \forall a_i \in T$, T being the set of tight constraints. For sufficiently small ε, $x + \varepsilon d$ are feasible. Also, $c^T(x \pm \varepsilon d) = c^T x \pm \varepsilon c^T d$.

Since x is optimal, $c^T x \pm \varepsilon c^T d = c^T x \Rightarrow c^T d = 0$. In one of the directions, x_i will change for some x_i. We keep moving until we hit a constraint.

The Simplex Algorithm (Dantzig 1947)

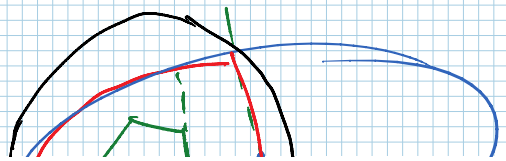
Class of algorithms based on moving from vertex to vertex until you reach an optimal solution. Specific algorithm depends on the "pivoting rule". The idea is to switch a constraint from the tight set with one from the non-tight set such that the objective improves or stays the same. It can be shown that this is always possible. There is no known pivoting rule for which the simplex algorithm is polytime. However, it has been quite successful in practice.

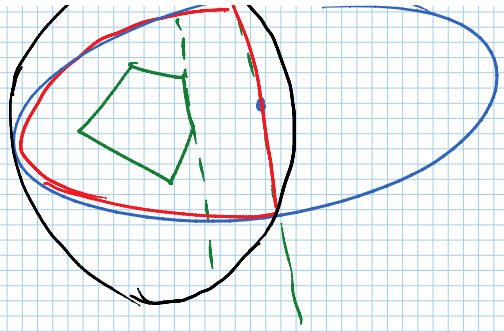
Ellipsoid Algorithm and Separation Oracles (Khachiyan '79)

- weakly polynomial algorithm
- only feasibility (recall that optimization and feasibility are essentially equivalent)

Idea:- Maintain ellipsoid containing polyhedron P.

- check if center of ellipsoid is in P
- if yes, done
- if not, find separating hyperplane (parallel to violated constraint through the center) and cut ellipsoid in two
- enclose half-ellipsoid containing P in minimal ellipsoid containing it.





- Initial ellipsoid: Ball centered at 0 with large enough radius.
- Can show that $\frac{Vol(E_{k+1})}{Vol(E_k)} < \exp^{-\frac{O(1)}{n}}$

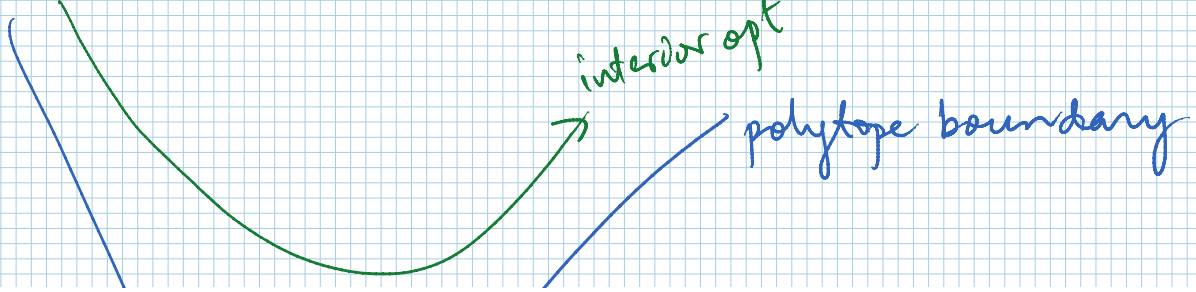
  $\Rightarrow$ # of iterations $= O\left(n \log \frac{Vol(E_0)}{Vol(P)}\right)$. As mentioned earlier,
- Can show that $Vol(E_0) \leq 2^{O(L)}$ and $Vol(P) \geq 2^{-O(L)}$

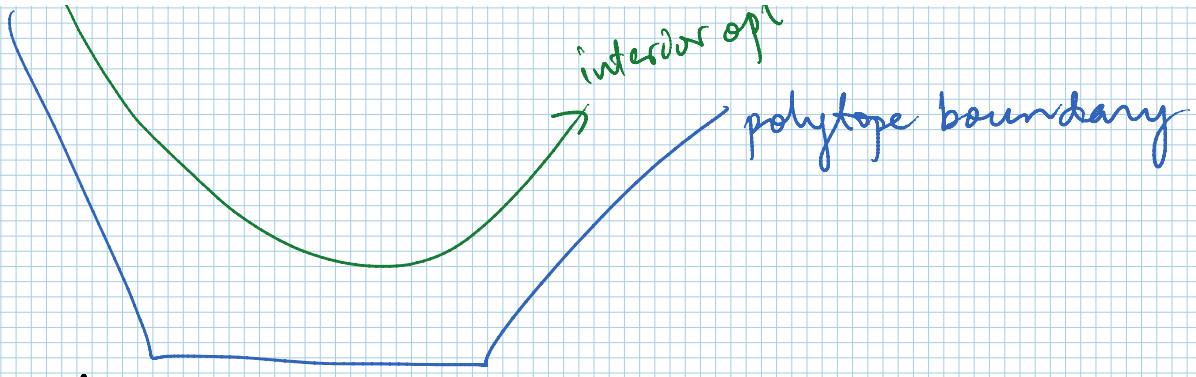
  where $L = m + n + \log det_{max} + \log b_{max} + \log c_{max}$,

  where $det_{max} = \max\{det(A') : A' \text{ is a square submatrix of } A\}$
- Running time for an iteration dominated by the time to find the bounding ellipsoid, which is polynomial.

Interior point algorithms

- More practical
- Always stay inside feasible space
- Use a potential function to measure duality gap and distance from boundary (includes a barrier function to move boundary unattractive)
- When close to the boundary, map polyhedron to new space (scaling) where you are not close to the boundary
- Avoid turning corners in simplex algorithm





Separation Oracles

Note that the ellipsoid algorithm does not require the LP to have polynomial number of constraints, but only requires a polytime SEPARATION ORACLE which tells us either that a point is feasible or gives a violated constraint.

Maxflow:

$$\max \sum_{p \in P(s,t)} f_p$$

$$\sum_{p : e \in p} f_p \le u_e \quad \forall p \in P(s,t)$$

$$f_p \ge 0$$

Exponential number of variables!
Polynomial number of constraints.

$$\min \sum u_e l_e$$

$$\sum_{e \in p} l_e \ge 1 \quad \forall p \in P(s,t)$$

$$l_e \ge 0$$

Polynomial number of variables
Exponential # of constraints.

Separation oracle: Shortest path!!

Global Min-cut

Find smallest cut in the graph.

UNDIRECTED

$$\min \sum u_e x_e$$

$$\text{s.t.} \sum_{e \in T} x_e \ge 1 \quad \forall T \in \mathcal{T}$$

$$\max \sum_{T \in \mathcal{T}} y_T$$

$$\sum_{T : e \in T} y_T \le u_e$$

s.t. $\sum_{e \in T} x_e \geq 1 \quad \forall T \in \mathcal{T}$ (set of spanning trees)

$x_e \geq 0$

$\sum_{T : e \in T} y_T \leq u_e$

$y_T \geq 0$

These are called tree packings

- What is an optimal tree packing for an n-vertex cycle?

Theorem (Nash-Williams, Tutte)

The maximum number of edge-disjoint spanning trees of an undirected graph is at least $\frac{1}{2}$ the size of the minimum cut.

DIRECTED GRAPH

$$\min \sum_{a \in A} u_a x_a$$

s.t. $\sum_{a \in T} x_a \geq 1 \quad \forall T \in \mathcal{T}_s$ (arborescences rooted at s)

$x_a \geq 0$

$$\max \sum_{T \in \mathcal{T}_s} y_T$$

s.t $\sum_{T \in \mathcal{T}_s} y_T \leq u_a$

$y_T \geq 0$

Theorem (Edmonds): The maximum number of s-arborences in a graph is exactly equal to the minimum cut with s on the source side of the cut.

Q: What happens for a cycle?