Apr 5

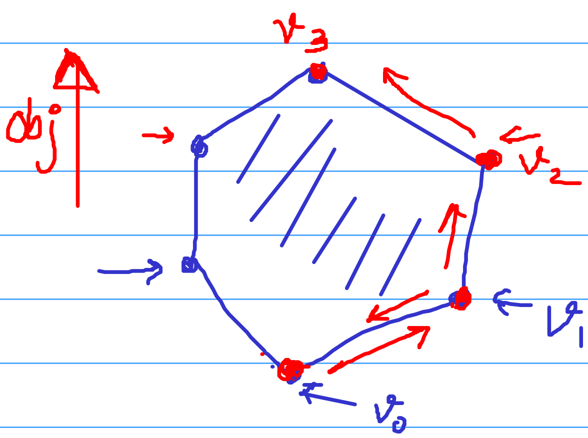
Solving Linear Programs:

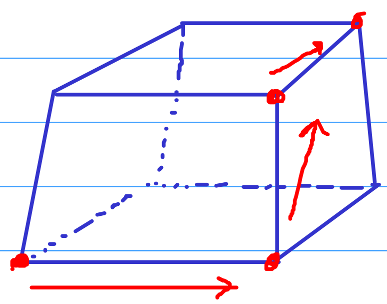
- → Simplex Algorithm (1950's) (not in P)
- → Ellipsoid Algorithm (1970's)
- → Interior point methods (1980's)
- → Gradient Descent

Convex Programming

Simplex Algorithm:

- Start with a vertex of the polyhedron (also called basic feasible solution)
- At any vertex, look at the neighboring vertices (connected by an edge). Go to one of the neighbors s.t. objective value improves.
- Stop when no improvement possible





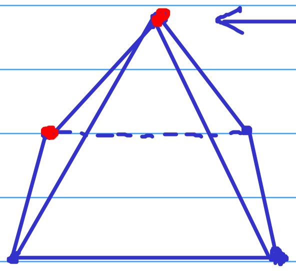
n = no. of varibles

$a_j^T x \le b_j$ for $1 \le j \le m$

Vertex: intersection of n tight constraints.

Edge: $n-1$ tight constraints.

Non-degeneracy assumption:
each vertex has exactly n tight constraints.
Important for bounding the running time of Simplex.



$\longleftarrow$ degenerate.

$n = 3$

How to walk along an edge?
Say, we are at a vertex defined by

$$\left.\begin{aligned} a_1^T x &= b_1 \\ a_2^T x &= b_2 \\ &\vdots \\ a_n^T x &= b_n \end{aligned}\right\}$$

choose one among them to relax

$$\left.\begin{aligned} a_{n+1}^T x &< b_{n+1} \\ &\vdots \\ a_m^T x &< b_m \end{aligned}\right\}$$

Feasible. One new tight constraint.

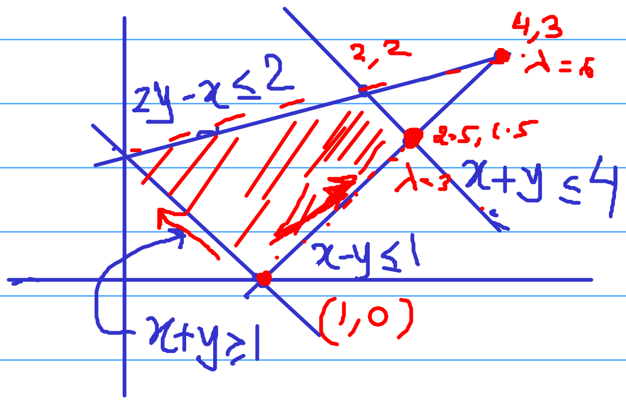
Say we choose $a_n^T x = b_n$ to be non-tight.

$$a_n^T x = b_n - \lambda \leftarrow \qquad x(\lambda)$$

choose λ s.t. all other constraints remain feasible, while one new constraint becomes tight

Each non-tight constraint might give an upper bound on λ. Choose the minimum λ.

n possible edges.
Choose one that improves objective.



Max y

	×	$x+y=1$	$\leftarrow$ $x+y = 1+\lambda$
$\rightarrow$	✓	$\rightarrow$ $x-y=1$	$x-y=1$
	×	$\rightarrow$ $2y-x<2$	$x = 1+\lambda/2$
$\rightarrow$	✓	$x+y<4$	$y = \lambda/2$

$$2y - x = \lambda - 1 - \lambda/2 = \lambda/2 - 1 \leq 2$$
$$\Rightarrow \lambda \leq 6$$
$$x+y = 1+\lambda \leq 4 \Rightarrow \lambda \leq 3$$
$$\lambda = 3$$

$$x = 2.5 \qquad y = 1.5$$

Pivot

$n+k$ tight const. $\binom{n+k}{k+1}$

$n-1$ tight $\qquad$ $k+1$ const relaxed.

Claim: If we are not at an optimal point then there must be an edge that strictly improves the objective. HW

$\Rightarrow$ guarantees finiteness of the algorithm.

Running time: no. of vertices. $\binom{m}{n}$

In practice - good.

There is no known rule for selecting edges that guarantees polynomial time.

Open question. to find such a rule.

Good in Practice.

- Finding initial feasible solution? (Simplex)

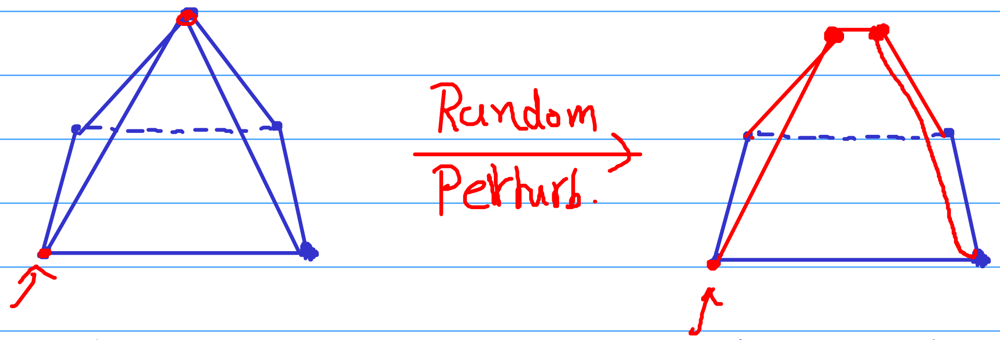
$$\left\{\begin{array}{l} Ax = b \\ x \geq 0 \end{array}\right. \quad \Bigg| \quad \left.\left\{\begin{array}{l} \min \sum z_i \\ Ax + z = b \\ x \geq 0 \\ z \geq 0 \end{array}\right.\right\} \left[\begin{array}{l} x = 0 \\ z = b \end{array}\right.$$

Claim: x is a feasible solution for LHS LP iff $(x, 0)$ is optimal for RHS LP.

$(x=0, z=b) \leftarrow$ Initial feasible for RHS LP. Find optimal using simplex.

Ensuring Non-degeneracy.

Random Perturbation $\quad a_j^T x \le b_j + \varepsilon_j$



After each constraint is randomly perturbed, each vertex will have exactly n tight constraints with high probability.

Note that small perturbation does not affect the objective value by much.

Smoothed Analysis of Simplex [2001]

(A, b)

$(A, b+\varepsilon) \Leftarrow$ polynomial time

Spielman, Teng 2001 showed that for every input (A, b) if it is randomly perturbed to $(A, b+\varepsilon)$ then with high prob simplex will run in polynomial time on $(A, b+\varepsilon)$

This is a much stronger statement than saying that simplex will take poly time on a random input.