Linear Programming

Find $x \in \mathbb{R}^d$ to maximize $c^T x$ (for given $c \in \mathbb{R}^d$)
subject to the system of linear inequalities (constraints):

$$Ax \leq b \iff \left.\begin{array}{c} a_1^T \cancel{x} \leq b_1 \\ \vdots \\ a_m^T x \leq b_m \end{array}\right\} m \text{ ineq.'s}$$

$a_i \in \mathbb{R}^{d \times 1}$ ← i^{th} row of A

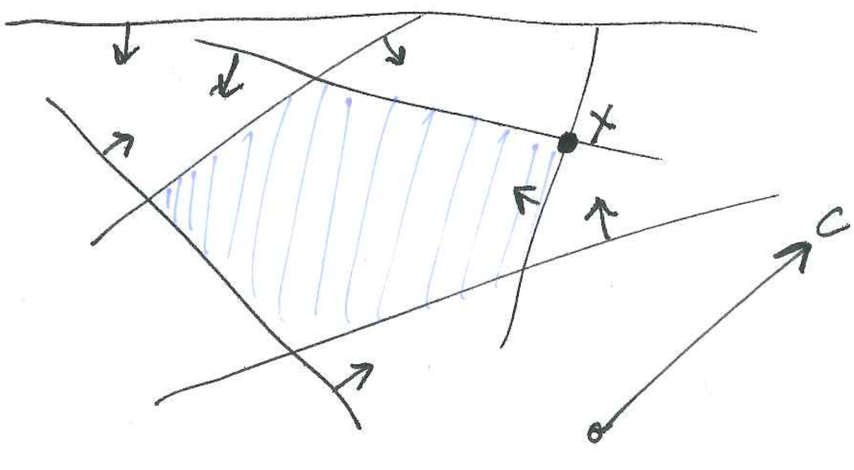
where A is $m \times d$ and b is $m \times 1$.

This has a geometric interpretation:

m constraints ⇝ m halfspaces

d variables ⇝ d dimensions

objective c ⇝ direction in $\mathbb{R}^d$



Some definitions

$$A = \begin{bmatrix} - & a_1 & - \\ & \vdots & \\ - & a_m & - \end{bmatrix}$$

i^{th} constraint is $a_i \cdot x \leq b_i$

Say x satisfies a_i if $a_i x \leq b_i$.

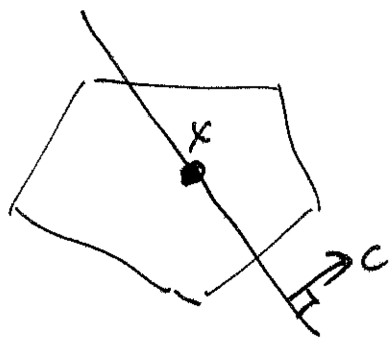
Say A is feasible if there exists x that satisfies $a_1, \ldots, a_m$.

Say x is a feasible solution if $Ax \leq b$.

Say a_i is tight if $a_i \cdot x = b_i$

Fact: If there is a finite optimal solution, then there is a solution at a vertex of the polyhedron.

pf First, observe that the solution must be on the boundary.



If x is the optimal solution then for all other feasible points y $c^T x \geq c^T y$. So the feasible solutions are in the halfspace $H = \{y : c^T(x-y) \geq 0\}$.

So, by definition, the intersection of the line bounding H and the polyhedron of feasible solutions is a face. So, it is a vertex x or it is an edge containing x.

Now we check that if x is on an edge $\overline{ab}$ then a and b are also optimal solutions.

Write x as $x = ta + (1-t)b$

WLOG $c^T a \leq c^T b$

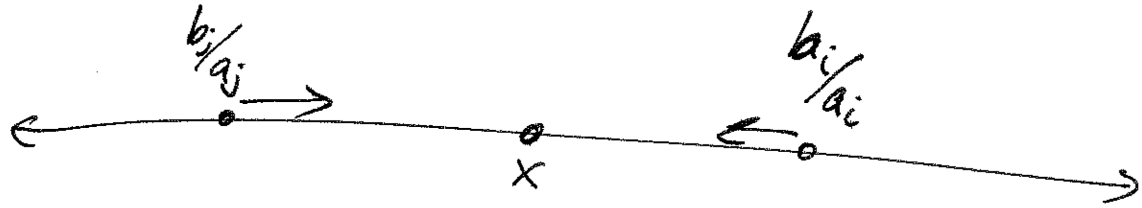
So $c^T x = c^T(ta + (1-t)b) = tc^T a + (1-t)c^T b \leq c^T b$

So b is at least as good a solution. It is strictly better if $c^T a \neq c^T b$, a contradiction.

Let's do linear programming in 1D

$$A = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \quad x \in \mathbb{R}$$

$$Ax \le b \iff \begin{matrix} a_1 x \le b_1 \\ \vdots \\ a_m x \le b_m \end{matrix}$$



2 cases:

(1) $a_i > 0$: In this case $x \le b_i/a_i$

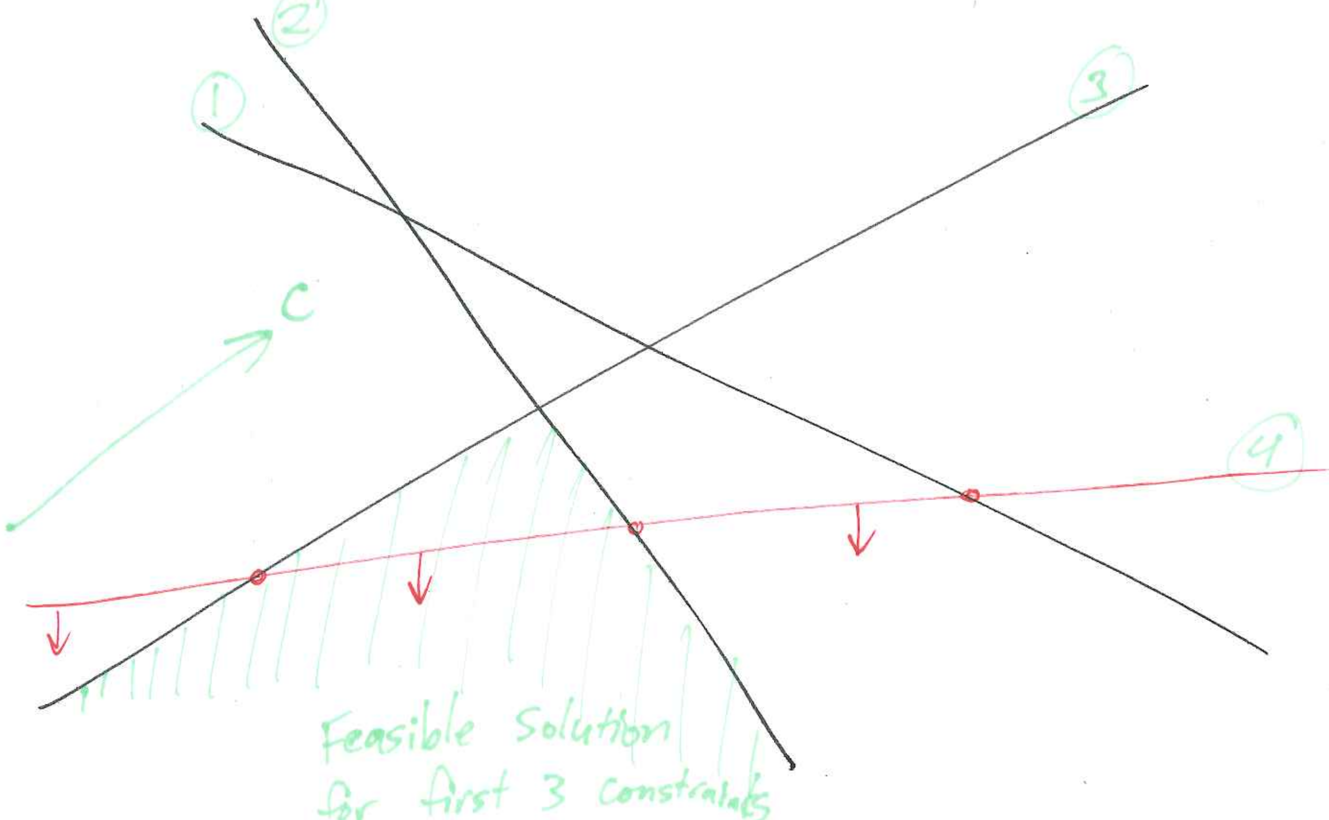
(2) $a_j < 0$: In this case $x \ge b_j/a_j$

If some negative a_j is s.t. $b_j/a_j \ge \frac{b_i}{a_i}$ for a positive a_i, then there is no solution.

Otherwise, the optimal solution is the min b_i/a_i among those for which $a_i > 0$.

Finding min in a list takes $O(n)$ time.

A 1D LP living in 2D



If the 4th constraint is tight, then the solution to the 2D LP is the solution to the 1D LP we get by restricting our attention to this one line.

The incremental algorithm

Add the constraints one at a time.
Let $x^{(i)}$ be the solution after i constraints have been added.

2 cases when adding constraint a_{i+1}.

(1) a_{i+1} is satisfied by $x^{(i)}$
↳ do nothing. Set $x^{(i+1)} := x^{(i)}$.

(2) a_{i+1} is not satisfied. (it will be tight)
Solve the 1D LP along a_{i+1}.

Analysis: Could be $O(n^2)$.
If we call 1DLP every time we get

$$\text{time} = \sum_{i=1}^{n} O(i) = O(n^2).$$

Let's Randomize (permute the constraints)

By a simple backwards analysis, we see that the probability the $(i+1)$st constraint is tight is only $\frac{2}{i+1}$. This is because ~~at~~ at most 2 constraints are tight and could cause us to run 1D LP. Each constraint has an equal probability of being the most recently added constraint.

$$\text{Expected Time} = O(n) + \sum_{i=1}^{n} \left(\frac{2}{i}\right) O(i) = O(n)$$

So, we can do 2D LP in $O(n)$ time.

In theory, this extends to any number of dimensions but the big-O hides constants that are large and depend heavily on the dimension.