Complexity & Algorithms, Spring '26

Linear Programming

Reminders:

- HW due Friday
- Next HW → up soon
- Next reading - 2 parts
- I'm out Friday, back Monday

Motivating example

n foods, m nutrients

Let a_{ij} = amount of nutrient i in food j

r_i = requirement of nutrient i $\in \{1 \dots m\}$

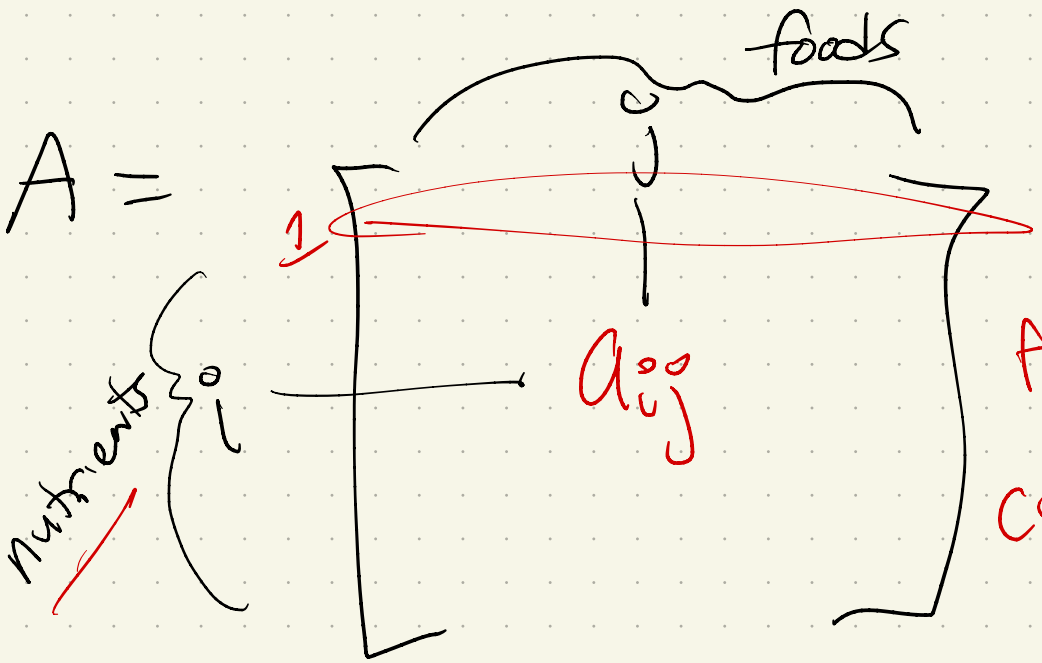
x_j = amount of food j purchased

c_j = cost of food j, $j \in \{1, \dots, n\}$

Goal: Buy food so you satisfy nutrients, while minimizing cost

What do we control? food I buy, how much

Can view as matrix →

$A =$ 

$\rightarrow \vec{r} = (r_1, r_2, \dots r_m)$

And $\rightarrow \vec{x} = (x_1, \dots, x_n)$

costs: $\vec{c} = (c_1, \dots, c_n)$

So: minimize cost $c_1x_1 + c_2x_2 + \cdots + c_nx_n$

$$= \sum_{i=1}^{n} c_i x_i$$

s.t. satisfy nutrition:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots a_{1n} \geq r_1$$

$$A\vec{x} \geq \vec{r}$$

$$\vec{x} \geq \vec{0}$$

$x_i \geq 0$

In general, get systems like this:

$$\text{maximize} \sum_{j=1}^{d} c_j x_j$$

$$\text{subject to} \sum_{j=1}^{d} a_{ij} x_j \leq b_i \quad \text{for each } i = 1 \,..\, p$$

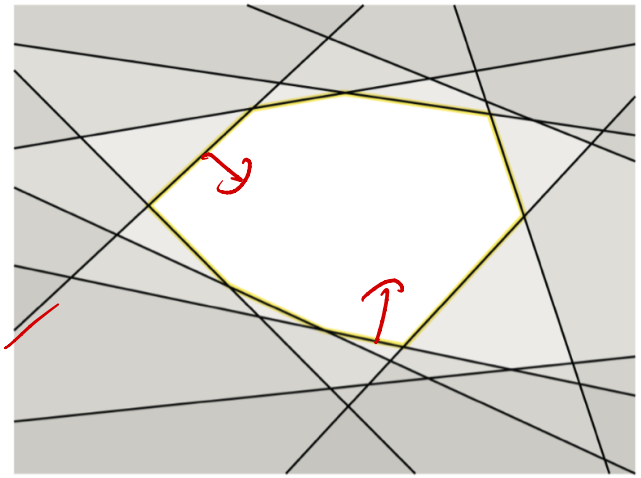
$$\sum_{j=1}^{d} a_{ij} x_j = b_i \quad \text{for each } i = p+1 \,..\, p+q$$

$$\sum_{j=1}^{d} a_{ij} x_j \geq b_i \quad \text{for each } i = p+q+1 \,..\, n$$

linear eqns

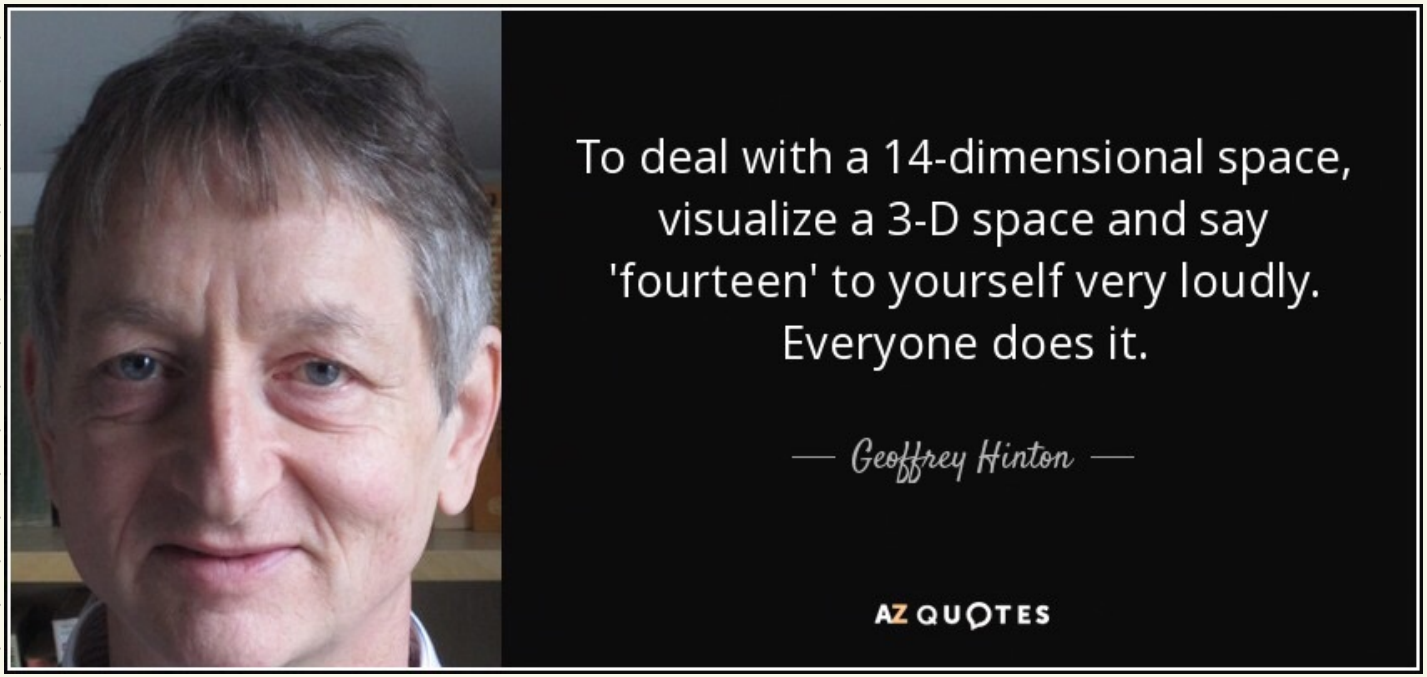
Geometric picture:

$\mathbb{R}^2$



A two-dimensional polyhedron (white) defined by 10 linear inequalities.

To answer about higher dimensions:
A moment of honesty!



I'll draw 2d & 3d pictures, but most of this works fine in $\mathbb{R}^d$!

History

Dates back to 1800's where studied by Fourier.

By 1940's: serious study, due to business/optimization demand

- Not known to be NP-Hard
  (Karp actually listed it as key open question in original paper on NP-Hardness)

Canonical form:

Avoid having both $\leq$ and $\geq$.

Why? Simplifies algorithms.

So get something more like our first example:

$$\text{maximize} \sum_{j=1}^{d} c_j x_j$$

$$\text{subject to} \sum_{j=1}^{d} a_{ij} x_j \leq b_i \quad \text{for each } i = 1..n$$

$$x_j \geq 0 \quad \text{for each } j = 1..d$$

Note: all c_j, b_i others can be + or −

Or, given a vector $\vec{c}$, matrix A + vector $\vec{b}$:

$$\max \vec{c}^T \vec{x}$$

$$\text{s.t. } A\vec{x} \leq \vec{b}$$

$$\vec{x} \geq \vec{0}$$

Anything can be put into canonical form:

The reduction:

① Avoid $\geq$:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq b$$

$$\times(-1) \iff -a_1x_1 - a_2x_2 \cdots \leq -b$$

② Avoid $=$:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

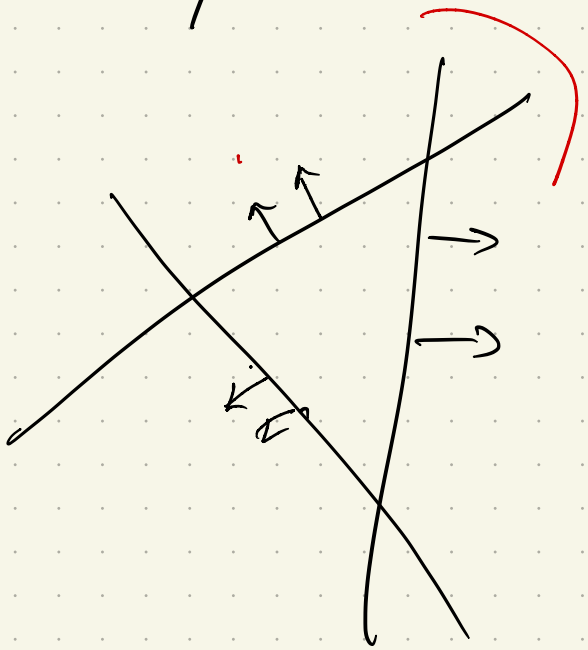
$$\Downarrow \times 2$$

③ $\min \ c^T\vec{x} \rightarrow \max \ -c^T\vec{x}$

$$\left.\begin{array}{l} \leq b \\ \geq b \end{array}\right\} 2eqn$$

How could these not have a solution?

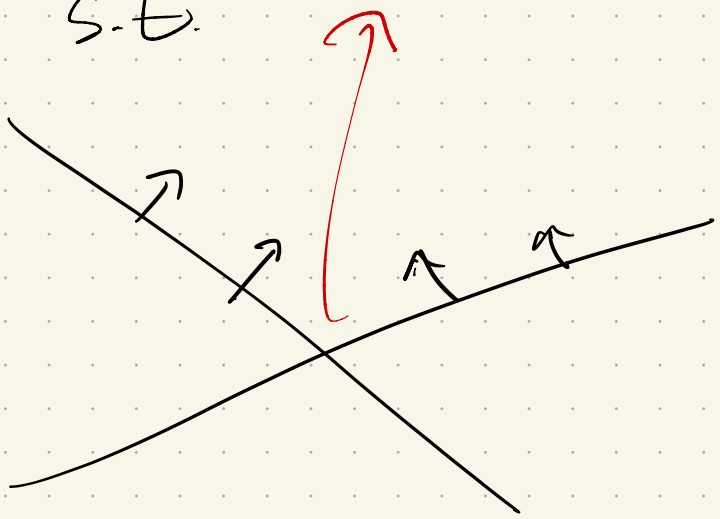
2 ways:



or

maximize $x_1 + x_2$

s.t.

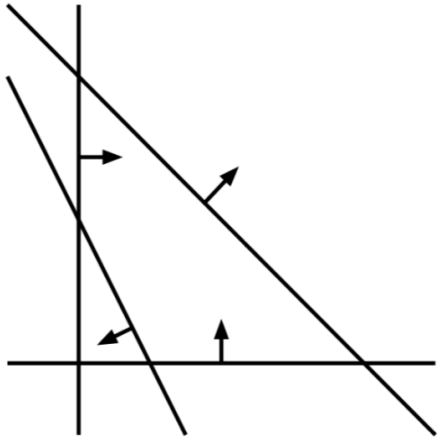


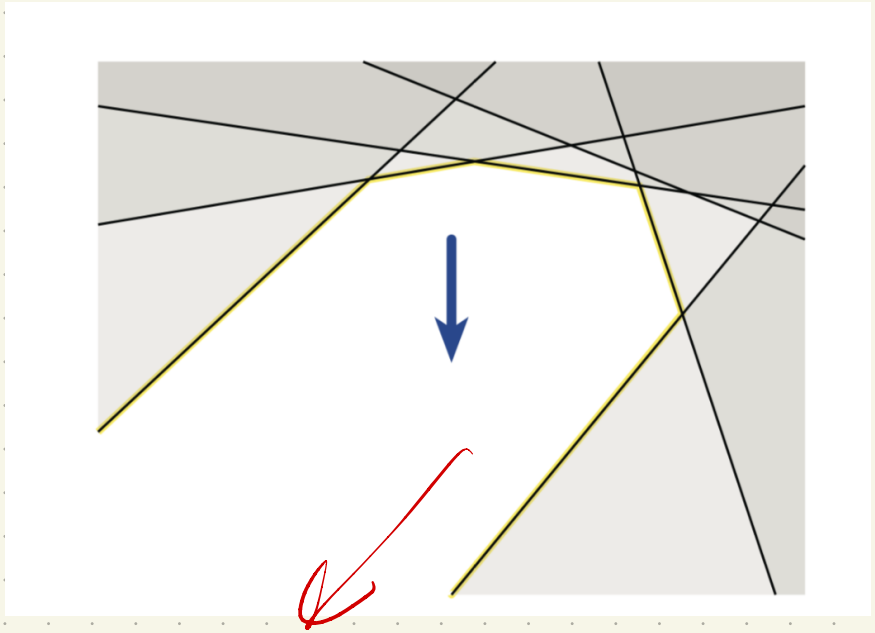
→ infeasible; some set of constraints mean no $\vec{x}$ exists

unconstrained ↓ bounding box

Better pictures (still 2d):

$$\begin{aligned} \text{maximize } & x - y \\ \text{subject to } & 2x + y \leq 1 \\ & x + y \geq 2 \\ & x, y \geq 0 \end{aligned}$$





Note:

① Multiplying by -1 turns any maximization problem into a minimization one:

why?

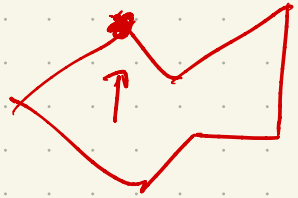
② Can turn inequalities into equalities via "slack" variables: (not canonical form)

$$\sum_{i=0}^{n} a_i x_i \leq b \Rightarrow \sum_{i=0}^{n} a_i x_i + s = b, \quad s \geq 0$$

Q: Why slack? vs "leeway"

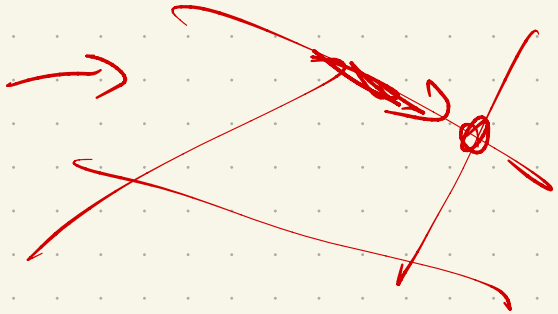
Aside: Why linear?

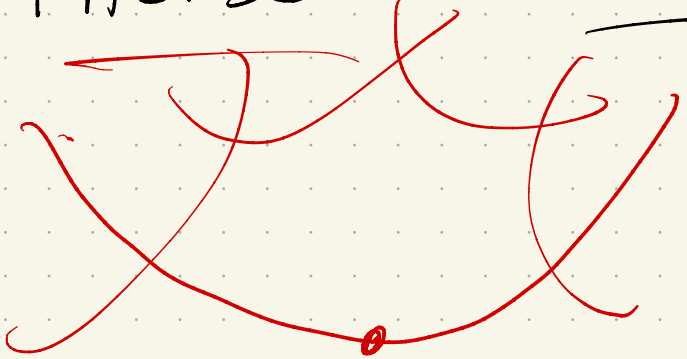
- Convexity: local optimum issues



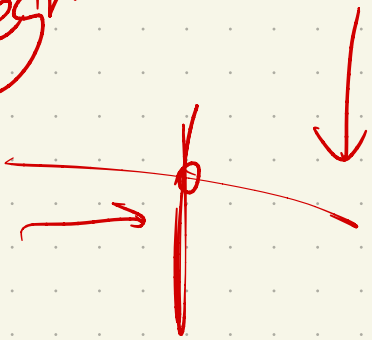
get stuck

- Linear intersections are optimum



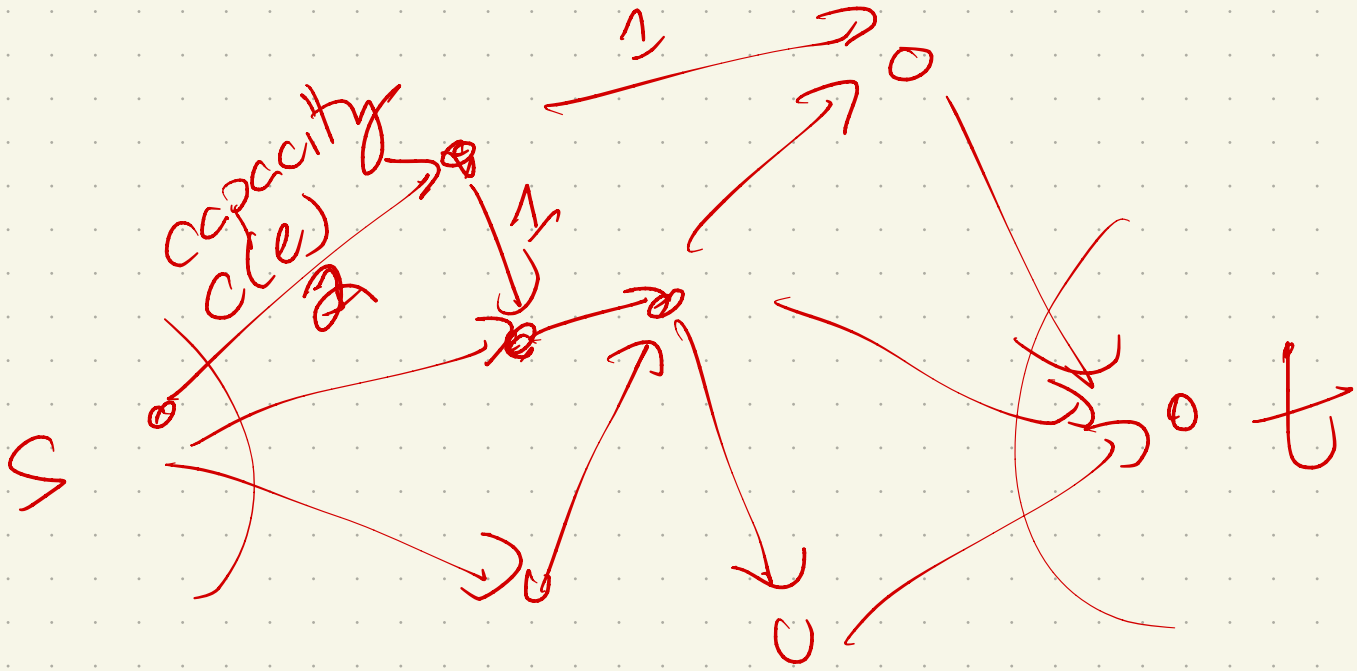


1 "segment"



- Complexity:

even quadratic eqns are NP-Hard



flow: assign of $f(e) \le c(e)$

$$\max \sum_{e \text{ out of } s} f(e)$$

s.t. $f_{in}(v) = f_{out}(v)$ $\forall$ vertices $v \neq s, t$.

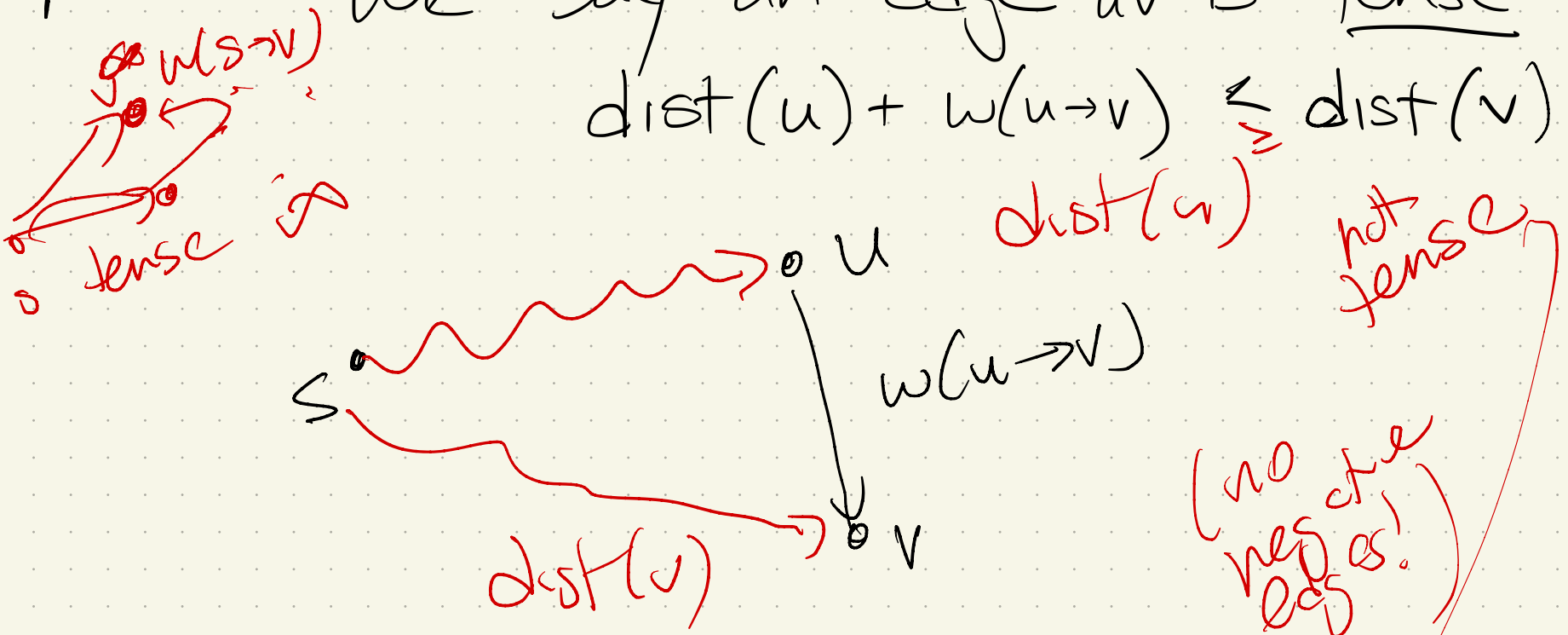
s.t. $0 \le f(e) \le c(e)$

Modeling shortest paths

Use "tenseness" of an edge.

Recall: We say an edge $\overrightarrow{uv}$ is <u>tense</u> if

$$dist(u) + w(u \to v) < dist(v)$$

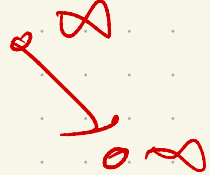


so e not tense $\Rightarrow$

$$dist(v) - dist(u) \leq w(u \to v)$$

So: LP
$dist(s) = 0$
$\forall e,\ dist(v) - dist(u) \le w(u \to v)$

$\forall d(v) = \infty$



← no edge tense

What the LP sets: the dist values!

Remember intial setup in Dijkstra:

$\forall v,\ dist(v) = \infty$

increase barely until have just tense edges

Note: $\forall v, d(v) = 0$ is actually feasible!

But is not distances

maximization

Duality: A simple example

I run a chocolate store, with 2 kinds:

- cheap → sell for $1 profit
- expensive → sell for $6 profit

I can sell up to 400 a day. $x_1 + x_2 \leq 400$

My supplier can only give me 200 of the cheap ones, & 300 of the fancy ones ($x_1 \leq 200$, $x_2 \leq 300$)

(& they don't stay fresh)

Goal: maximize profit: $1 \cdot x_1 + 6 \cdot x_2$

Result: An LP to maximize profit.

LP: max $x_1 + 6x_2$

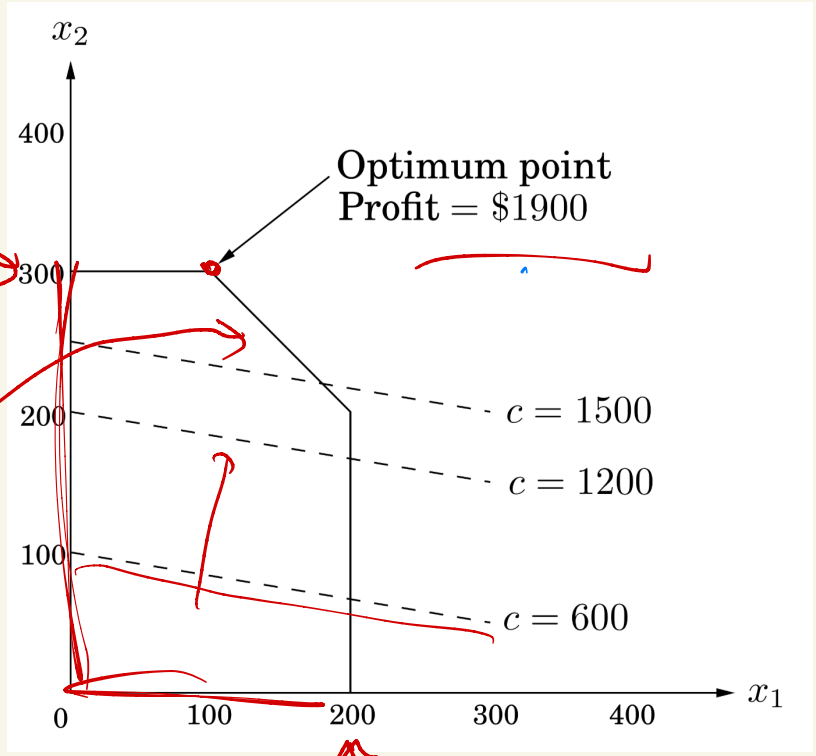
s.t.

$x_1 \leq 200$

$x_2 \leq 300$

$x_1 + x_2 \leq 400$

$x_1, x_2 \geq 0$



Can we check that this is best?

$$\max \; x_1 + 6x_2$$

s.t.

$$x_1 \le 200$$
$$x_2 \le 300$$
$$x_1 + x_2 \le 400$$
$$x_1, x_2 \ge 0$$

① ← equation 1

② ← eqn

Play w/ inequalities:

① + 6·②:

$$x_1 \le 200$$
$$6x_2 \le 1800$$

↳ $x_1 + 6x_2 \le 2000$

Interesting!

These 2 inequalities tell us that we couldn't ever beat $2000.

But recall soln was $1900.

Can we get a better combo?

$$\max \quad x_1 + 6x_2$$

s.t.

$$x_1 \le 200 \quad (1) \times 0$$
$$x_2 \le 300 \quad (2) \times 5$$
$$x_1 + x_2 \le 400 \quad (3) \times 1$$
$$x_1, x_2 \ge 0$$

Play: $0 \cdot (1) + 5 \cdot (2) + 1 \cdot (3)$

$$5x_2 \le 1500$$
$$x_1 + x_2 \le 400$$

$$\Rightarrow x_1 + 6x_2 \le 1900$$

These multipliers, (0, 5, 1), are a certificate of optimality.

↳ No valid solution can ever beat $1900

But

How do we find these magic values??

In this, we had three "≤" inequalities

↳ so goal is to find the right 3 multipliers: y_1, y_2, and y_3

Let's try to rewrite...

Multiplier		Inequality	
y_1	$\times$	$x_1 \leq 206$	eq 1
y_2	$\times$	$x_2 \leq 300$	eq 2
y_3	$\times$	$x_1 + x_2 \leq 400$	eq 3

Result

$y_1 \cdot x_1 \leq 200 y_1$

$y_2 x_2 \leq 300 y_2$

$y_3 x_1 + y_3 x_2 \leq 400 y_3$

Rewrite: Make left side look like the original max/min goal, so right will be an upper bound

So here:

$$(y_1 + y_3)x_1 + (y_2 + y_3)x_2 \leq 200y_1 + 300y_2 + 400y_3$$

Means:

$$1 \cdot x_1 + 6 \cdot x_2 \leq 200y_1 + 300y_2 + 400y_3$$

if:

$$\begin{cases} y_1, y_2, y_3 \geq 0 \\ y_1 + y_3 \geq 1 \\ y_2 + y_3 \geq 6 \end{cases}$$

] b/c if negative, inequalities flip!

} b/c original eqn.

Any y_i's would give an upper bound!

We want the best one

↳ ie minimize another LP!

Duality recap

$$\text{s.t.} \quad \max \; x_1 + 6x_2$$
$$x_1 \le 200$$
$$x_2 \le 300$$
$$x_1 + x_2 \le 400$$
$$x_1, x_2 \ge 0$$

dual $\Longleftrightarrow$

$$\min \; 200y_1 + 300y_2 + 400y_3$$
$$\text{s.t.}$$
$$y_1 + y_3 \ge 1$$
$$y_2 + y_3 \ge 6$$
$$y_1, y_2, y_3 \ge 0$$

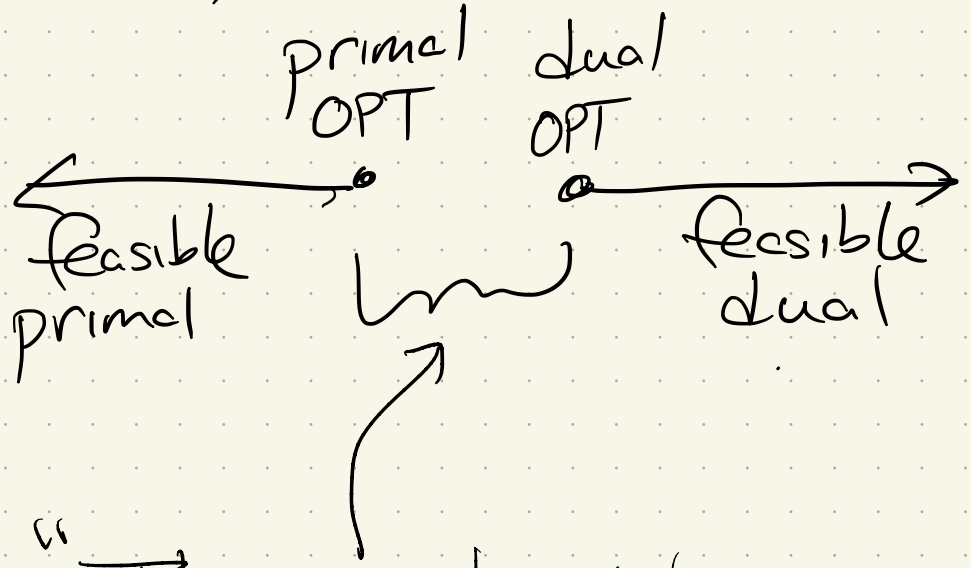
Any solution to bottom is upper bound to top LP.

$\Rightarrow$ If we can find primal/duals that are equal, both are OPT

Here, 1900: primal $(x_1, x_2) = (100, 300)$

Dual: $(y_1, y_2, y_3) = (0, 5, 1)$ ←

Works for any LP:



Fundamental Theorem:

gap = 0 for LPs

In general:

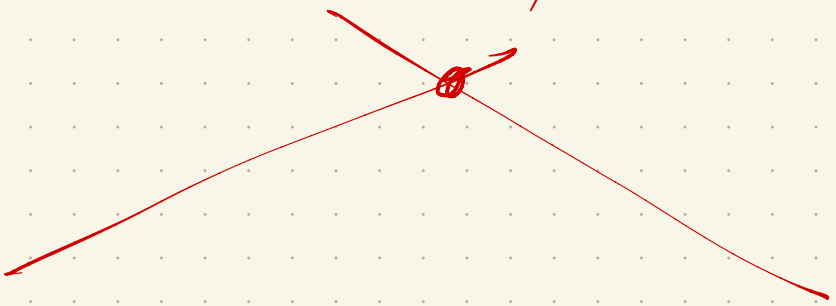
Primal LP	Dual LP
$\max \; c^T x$	$\min \; y^T b$
s.t.	s.t.
$A\vec{x} \leq \vec{b}$	$y^T A \geq c^T$
$\vec{x} \geq 0$	$y \geq 0$
$\vec{c}, \vec{x}, \vec{b}$ vectors	$\vec{c}, \vec{b}, \vec{y}$, vectors

Limits of LP:

Many things are <u>not</u> LPs!

Ex: Integer solutions

LPs allow fractions/decimals



ILP is NP-Hard

↳ many solvers

Ex: Quadratic or more complex constraints

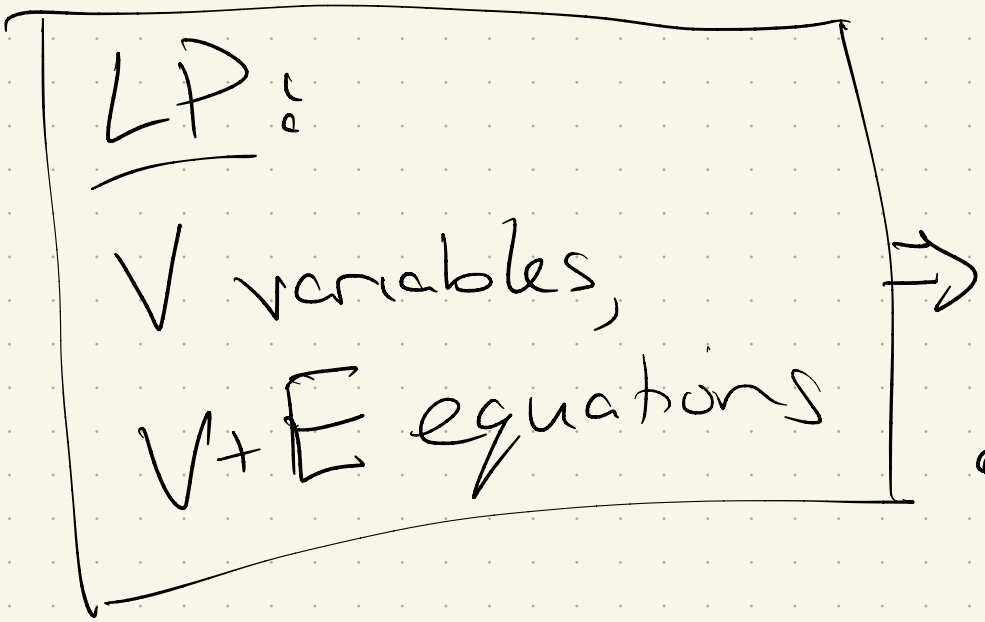
↳ distances in $\mathbb{R}^2$

$d(x,y) = \sqrt{(x_2-x_2)^2+(y_1-x_1)^2}$ X

Often other approaches could give a better runtime!

Ex: Shortest paths

Dijkstra: $E \log V$



⇒

- LP w/ simplex alg. is exponential
- with "better" algorithm matrix multiply time $E^{\omega} = E^{2.7\ldots}$

The algorithm: Simplex (Dantzig 1947)

Assumes canonical form:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{d} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{d} a_{ij} x_j \le b_i \quad \text{for each } i = 1..n \\ & x_j \ge 0 \quad \text{for each } j = 1..d \end{aligned}$$

So:

- no min
- only $\le$
- $x \ge 0$ for all variables
- fast in practice, but exponential in worst case

Klee-Minty, 1973: some feasible polytopes have $O(n^{\lfloor d/2 \rfloor})$ vertices

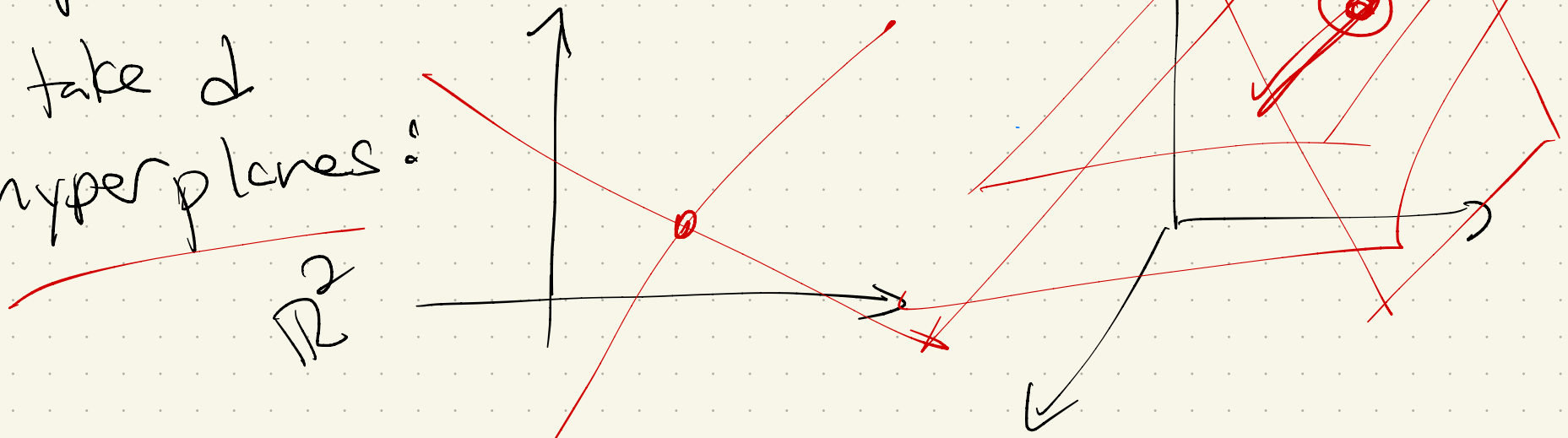
Simplex

Take any vertex v in feasible region
While some neighbor v' is better
$v \leftarrow v'$

(Details lacking!)

Step 1: find a vertex

take d hyperplanes:



Any d hyperplanes give a vertex.

Loop through m-d others!

Either feasible for each

Or Not

↳ use this new hyperplane

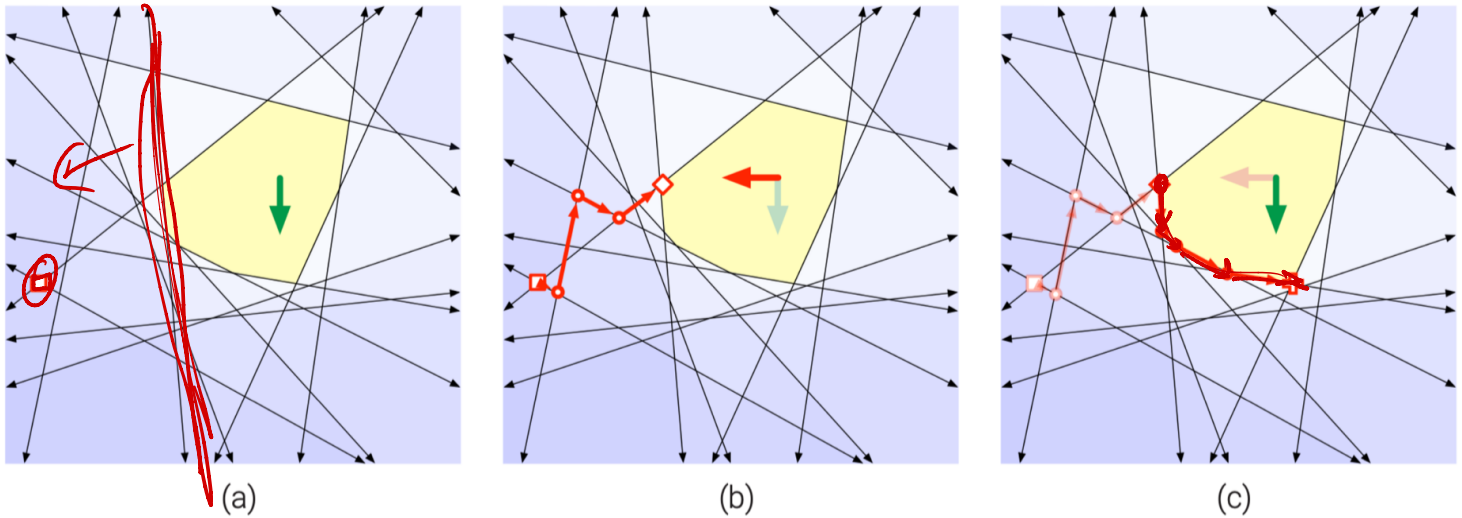
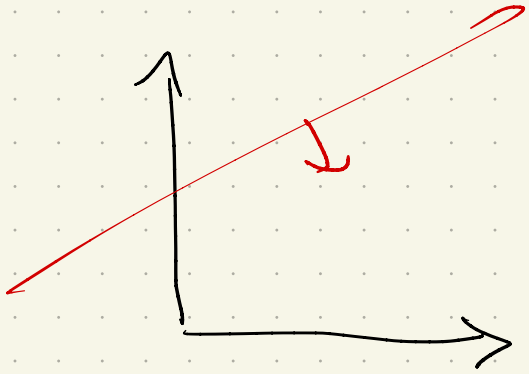


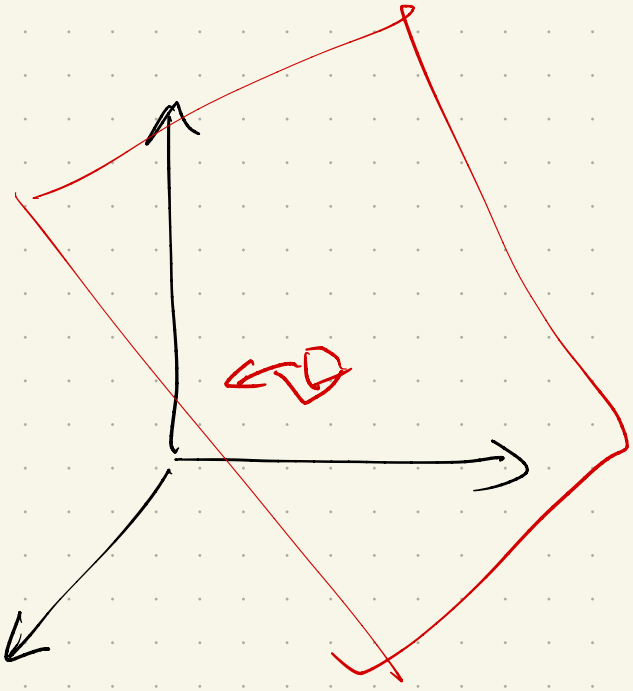
Figure I.2. Primal simplex with dual initialization: (a) Choose any basis. (b) Rotate the objective to make the basis locally optimal, and pivot "up" to a feasible basis. (c) Pivot down to the optimum basis for the original objective.

Let's go over that more carefully:

Each LP equality or inequality describes a hyperplane in $\mathbb{R}^d$.

2d: $ax+by \leq c$





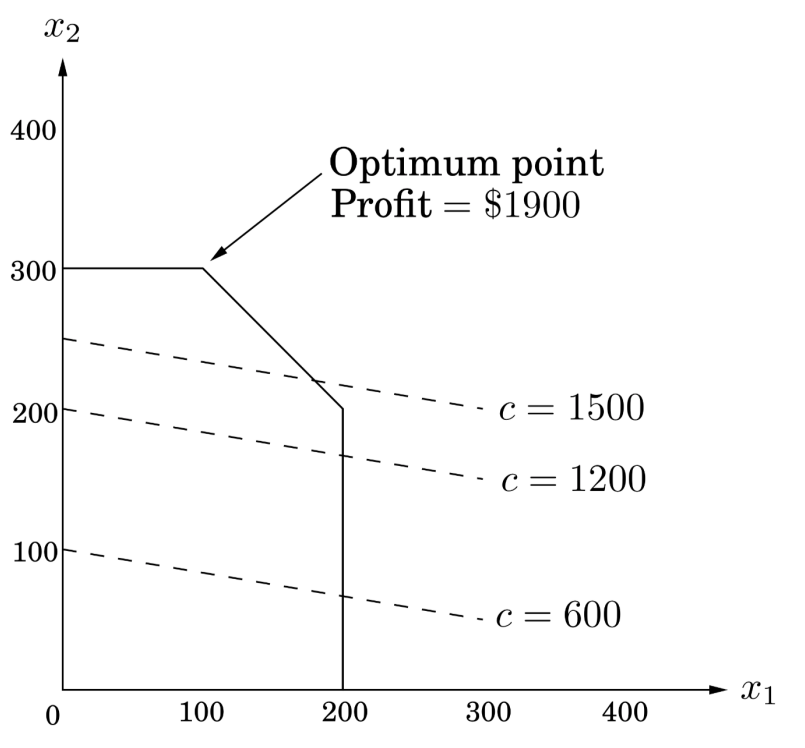
3d: $ax+by+cz \leq d$

$\mathbb{R}^d$: $a_1x_1 + a_2x_2 + \cdots + a_dx_d \leq c$

Vertices:

These happen when $> d$ hyperplanes meet in $\mathbb{R}^d$.

In $\mathbb{R}^2$ a

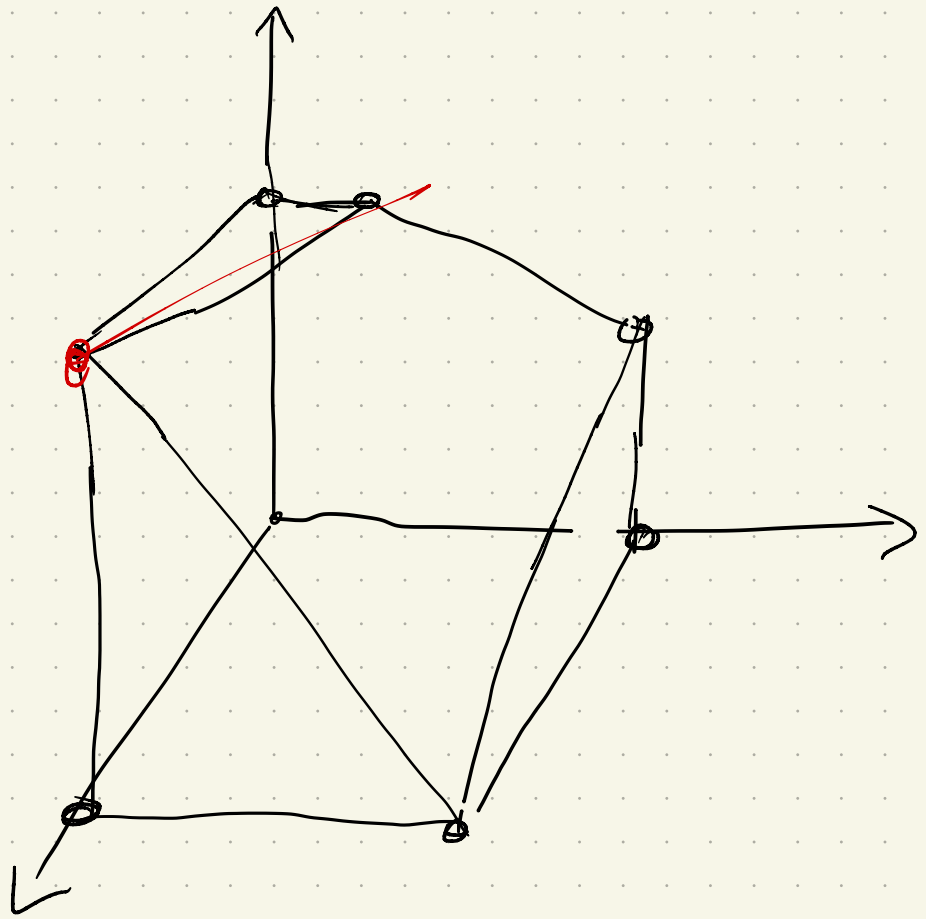


Note:
Not every pair will meet!

In $\mathbb{R}^3$

Any 2 intersect in a line.

Any 3 intersect in a vertex (assuming not parallel)



In $\mathbb{R}^d$: d equations, d variables $\Rightarrow$ one point

Dfn: Pick a subset of inequalities

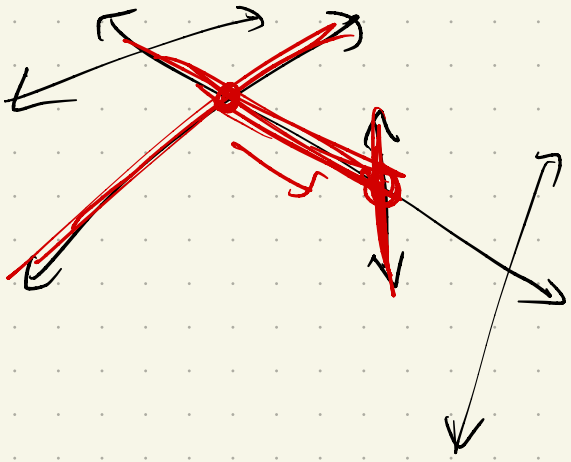
If there is a unique point that satisfies all with equality, & it is feasible

↳ this is a vertex of the solution

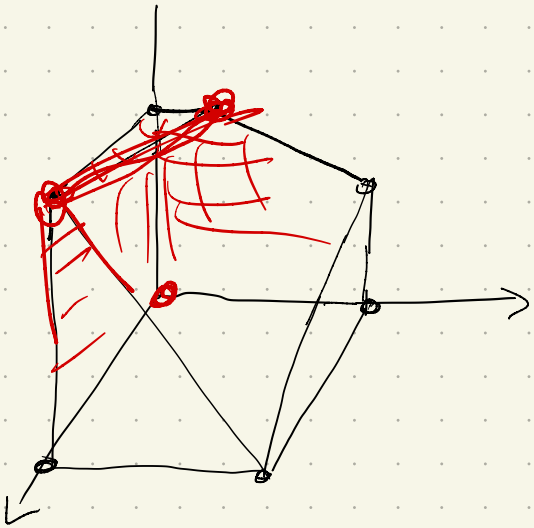
move to nbr

Since each vertex is specified by d equations, we call any two that share d-1 of them neighbors:

2d:



3d:



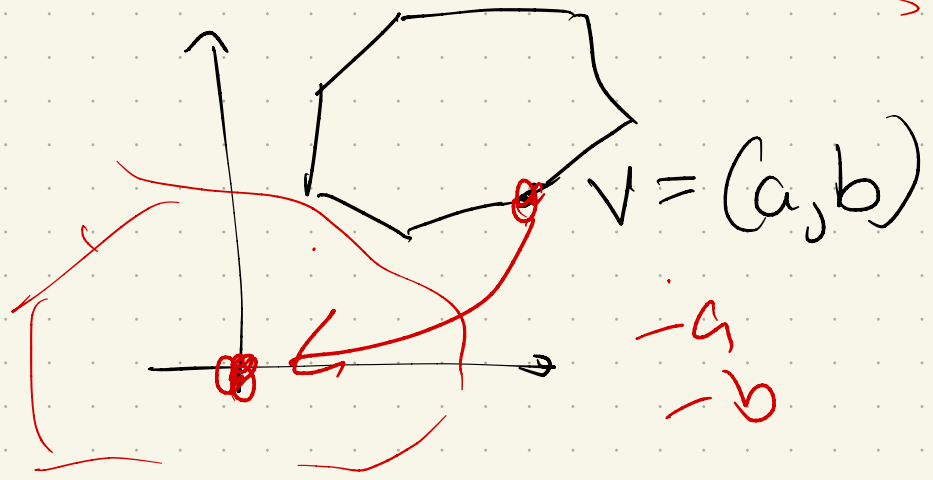
Simplex algorithm:

In each stage, 2 tasks:

1. Check if current vertex is optimal
2. If not choose a neighbor that gives a better score under the objective function

Both are easy if $v = \vec{0}$ (the origin) (see next slide)

If not at $\vec{0}$: translate



LP: $\max c^T x$

s.t $A\vec{x} \leq \vec{b}$

$x_i \geq 0 \quad \forall i$

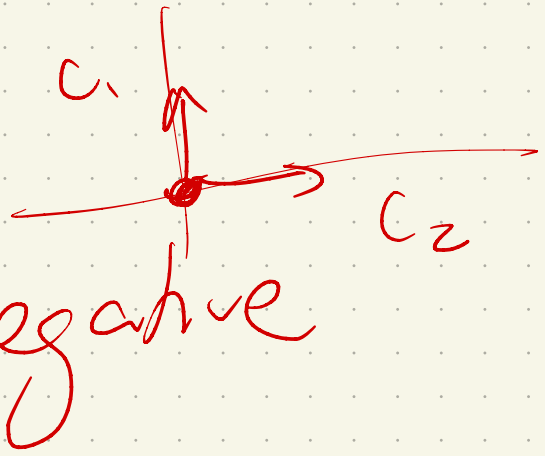
Note: $\vec{x} \in \mathbb{R}^d$, so $x = (x_1, \ldots, x_d)$

Now, $\vec{0}$ is always a vertex — why?

→ intersection of these hyperplanes

Optimal only if

all c_i's are negative



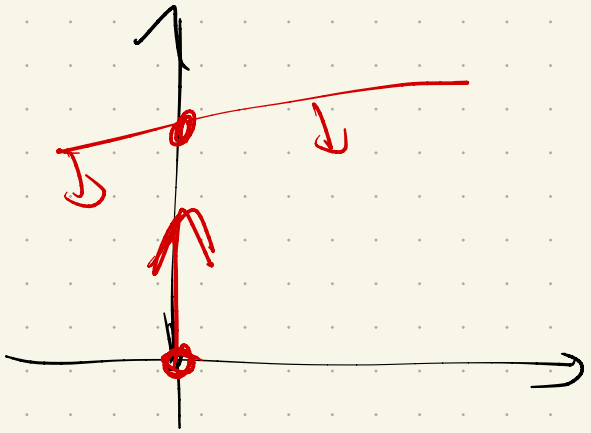
Conversely:

If <u>any</u> $c_i > 0$, we can increase the obj. function $\underline{c^T \vec{x}}$

How? increase x_i; $c_i x_i > 0$

So: pick one & increase!

How much?



either c_1 or c_2 is positive

↳ when do we get stuck?

when we hit a line

Runtime:

Consider a vertex $\vec{v} \in \mathbb{R}^d$ with m inequalities & d variables

How many possible neighbors?

each removes 1 eqn & replaces with another

$\Rightarrow \leq d \cdot (m-d)$

Checking if it is a neighbor & is feasible: basically dot product & Guassian elimination.

So: each iteration is

$$\leq d \cdot (m-d) \cdot [\text{time for G.E}]$$

$$[\text{time for G.E}] = n^{\omega}$$

Can improve slightly:

- just need one $c_i > 0$ + rescaling to $\vec{0}$ is easy.

So can get to $O(m \cdot n)$

How many iterations?

{m+d inequalities
any d give a vertex} $\Longrightarrow$ $\binom{m+d}{d}$

$\leq (m+d)^d$

So exponential in worst case.

(Remember, for a while people thought this might be NP-Hard)

Klee & Minty gave examples in the 70's that actually take exponential time.

Can we avoid this by choosing the "best" neighbor in our update?

No ideal way!

Many proposed, but for almost every one, there is some input polyhedron that needs an exponential number of pivots.

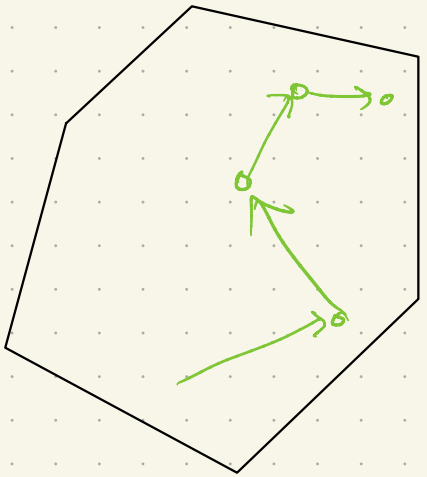
Ellipsoid algorithm, Khachiyan 1979

- (weakly) polynomial time
  - ↪ dependent of precision

- high level idea: compute smaller & smaller ellipses which contain solution

Interior point Methods, Karmarkar 1984

- Move through polytope's interior
  - Still weakly polynomial
    - but - practical!



More recent:

- Matrix Multiply time (in 2019!)

RESEARCH-ARTICLE

Solving linear programs in the current matrix multiplication time

Authors: Michael B. Cohen, Yin Tat Lee, Zhao Song | Authors Info & Claims

STOC 2019: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing • Pages 938 - 942
https://doi.org/10.1145/3313276.3316303

Published: 23 June 2019 Publication History

Check for updates

Abstract

This paper shows how to solve linear programs of the form $\min_{Ax=b,x\geq 0} c^\top x$ with n variables in time $O^*((n^\omega+n^{2.5-\alpha/2}+n^{2+1/6})\log(n/\delta))$ where ω is the exponent of matrix multiplication, α is the dual exponent of matrix multiplication, and δ is the relative accuracy. For the current value of $\omega\sim 2.37$ and $\alpha\sim 0.31$, our algorithm takes $O^*(n^\omega \log(n/\delta))$ time. When $\omega = 2$, our algorithm takes $O^*(n^{2+1/6}\log(n/\delta))$ time.

Our algorithm utilizes several new concepts that we believe may be of independent interest:

(1) We define a stochastic central path method.

(2) We show how to maintain a projection matrix

$$\sqrt{W}A^\top(AWA^\top)^{-1}A\sqrt{W}$$

in sub-quadratic time under ℓ_2 multiplicative changes in the diagonal matrix W.

This is a large & active area of study: