CS 170
Efficient Algorithms and Intractable Problems

Lecture 15
Linear Programming

Nika Haghtalab and John Wright

EECS, UC Berkeley

Announcements

Homeworks:

- HW7 is due later this evening (Tues evening).
- HW8 will be released later today, and due the following Wed

→ Why more time than usual?! Because it's more interesting (aka challenging!) ☺

→ Don't wait until the last minute (HWP on Friday and Monday only)

Midterm 1 regrade requests are open and will close next Monday!

→ Submit a regrade request in case we made a grading error.

Our Journey so far

We have learned o much this semester already!

- Divide and Conquer
- Graph Algorithms
- Greedy Algorithms
- Dynamic Programming

This and the next few lectures:
Linear Programming, Applications, Implications:

Today:

- Definitions
- Use cases
- Some Algorithms

Linear Programs

An Example

A bakery produces donuts and cakes.

Each have some profit and require some ingredients.

Item	Profit	Flour (200 total)	Sugar (300 total)	Milk (500 total)
	$5	2	2	7
	$25	5	9	12

How many units of each should the bakery produce to maximize its profit?

An Example

A bakery produces donuts and cakes.

Each have some profit and require some ingredients.

Item	Profit	Flour (200 total)	Sugar (300 total)	Milk (500 total)
	\$5	2	2	7
	\$25	5	9	12

Decision variables

\# donuts: x and \# cakes: y

My objective

$\max\ 5x + 25y$

Constraints

$x \geq 0, y \geq 0$

$2x + 5y \leq 200$

$2x + 9y \leq 300$

$7x + 12y \leq 500$

How many units of each should the bakery produce to maximize its profit?

Constrained Optimization

Many algorithm design problems we studied so far:

1) Solution has to meet some constraints:

→MST: set of edges connecting all vertices of a graph

→Longest increasing subseq: is a subsequence and monotonically increasing.

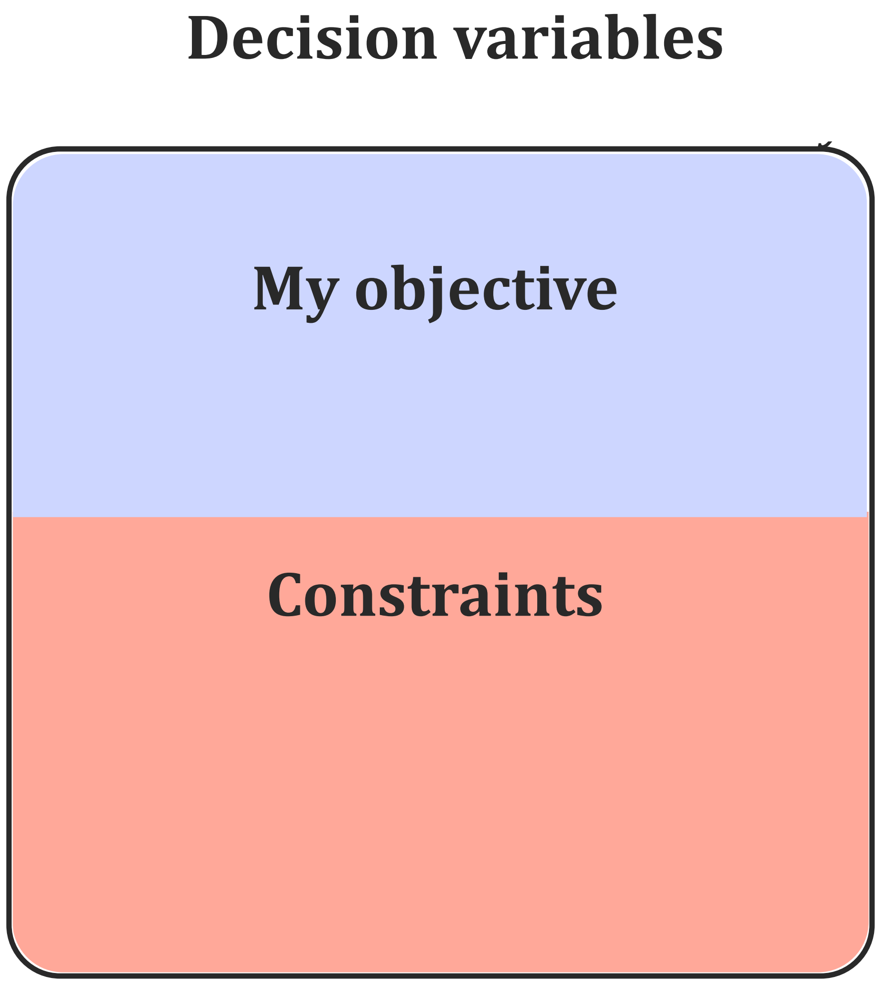
→Interval scheduling: Intervals don't conflict

2) Subject to the above, we optimize an objective:

→MST: Sum of edge weights

→Longest increasing subseq: size of subsequence

→Interval scheduling: Number of intervals



Linear Programming (LP)

A type of constrained optimization, where

- The objective function is a minimization (or a maximization) of **linear function of the decision variables.**
- Every constraints is **a linear function of the decision variables.**

The bakery problem is a linear program!

Decision variables

donuts: x and # cakes: y

My objective

$\max\ 5x + 25y$

Constraints

$x \geq 0, y \geq 0$

$2x + 5y \leq 200$

$2x + 9y \leq 300$

$7x + 12y \leq 500$

Linear Programming (LP)

A type of constrained optimization, where

- The objective function is a minimization (or a maximization) of **linear function of the decision variables.**
- Every constraints is **a linear function of the decision variables.**

The bakery problem is a linear program!

Algorithms for linear programs:

- Goal: an algorithm that solves any LP efficiently.
- So, if you can write a problem as an LP, you can solve it efficiently.

Decision variables

donuts: x and # cakes: y

My objective

$\max\ 5x + 25y$

Constraints

$x \geq 0, y \geq 0$

$2x + 5y \leq 200$

$2x + 9y \leq 300$

$7x + 12y \leq 500$

Integer or Fractional Decision Variables?

For linear Programs:

- Decision parameters are real values (belong to $\mathbb{R}$)

CANNOT have integer constraints in LPs.

Integer Linear Programs (ILP):

- A variant of linear programs, where decision variables can be integer.
- It is not an LP!

There are no algorithms that can solve any ILP in polynomial time.

But there are algorithms that can solve any LP in polynomial time!

Decision variables

\# donuts: x and \# cakes: y

My objective

$$\max\ 5x + 25y$$

Constraints

$$x \geq 0, y \geq 0$$

$$2x + 5y \leq 200$$

$$2x + 9y \leq 300$$

$$7x + 12y \leq 500$$

~~x and y are integers~~

LP Standard Form

n

Definition: A linear program is in standard form if you write is as follows

n variables $x_1, \ldots x_n$

$$
\begin{aligned}
\max \quad & c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\
\text{Subject to} \quad & a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n \le b_1 \\
& \qquad\quad\;\; + a_{22} x_2 + \cdots + a_{2n} x_n \le b_2 \\
& \vdots \\
& a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n \le b_m \\
\text{and} \quad & x_1 \ge 0, \ldots, x_n \ge 0
\end{aligned}
$$

m constraints

Non-negative variables in $\mathbb{R}$

$$\max \boldsymbol{c}^\top \mathbf{x}$$

$$\text{Subject to } \mathbf{A}\mathbf{x} \le \mathbf{b}$$

$$\text{and } \mathbf{x} \ge \mathbf{0}$$

A: m×n matrix

$\boldsymbol{c}, \mathbf{x}$: n-dim column vectors

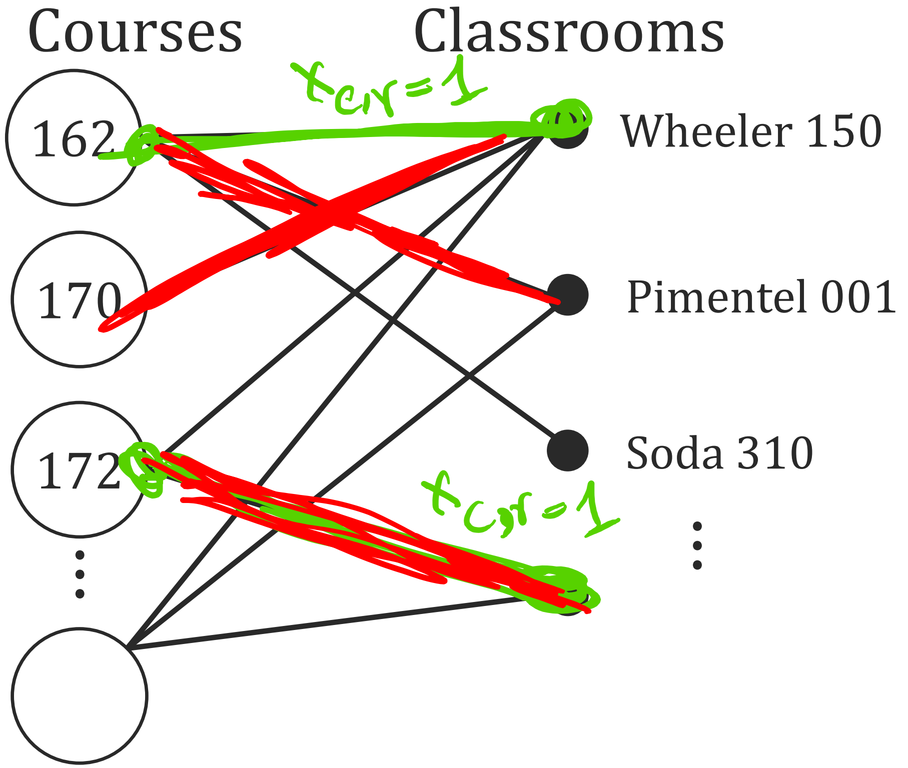
$\mathbf{b}$: m-dim column vectors

Variable: $\{ \; x_{c,r} = 0,1$ $\quad 1 \Rightarrow$ course c assigned to room r

Another Example: Classroom Allocation

$0 \Rightarrow$ course c not assigned for r

We have courses and classrooms in which they could fit. Each classroom can fit one course at most. Given Graph $G = (V, E)$, where course c can be assigned to classroom r iff $(c, r) \in E$.



Decision variable: $\boxed{x_{c,r}}$ for each c course, r room, $(c,r) \in E$.

Constraints: $0 \le x_{c,r} \cancel{\in \{0,1\}} \le 1 \quad \forall c,r$

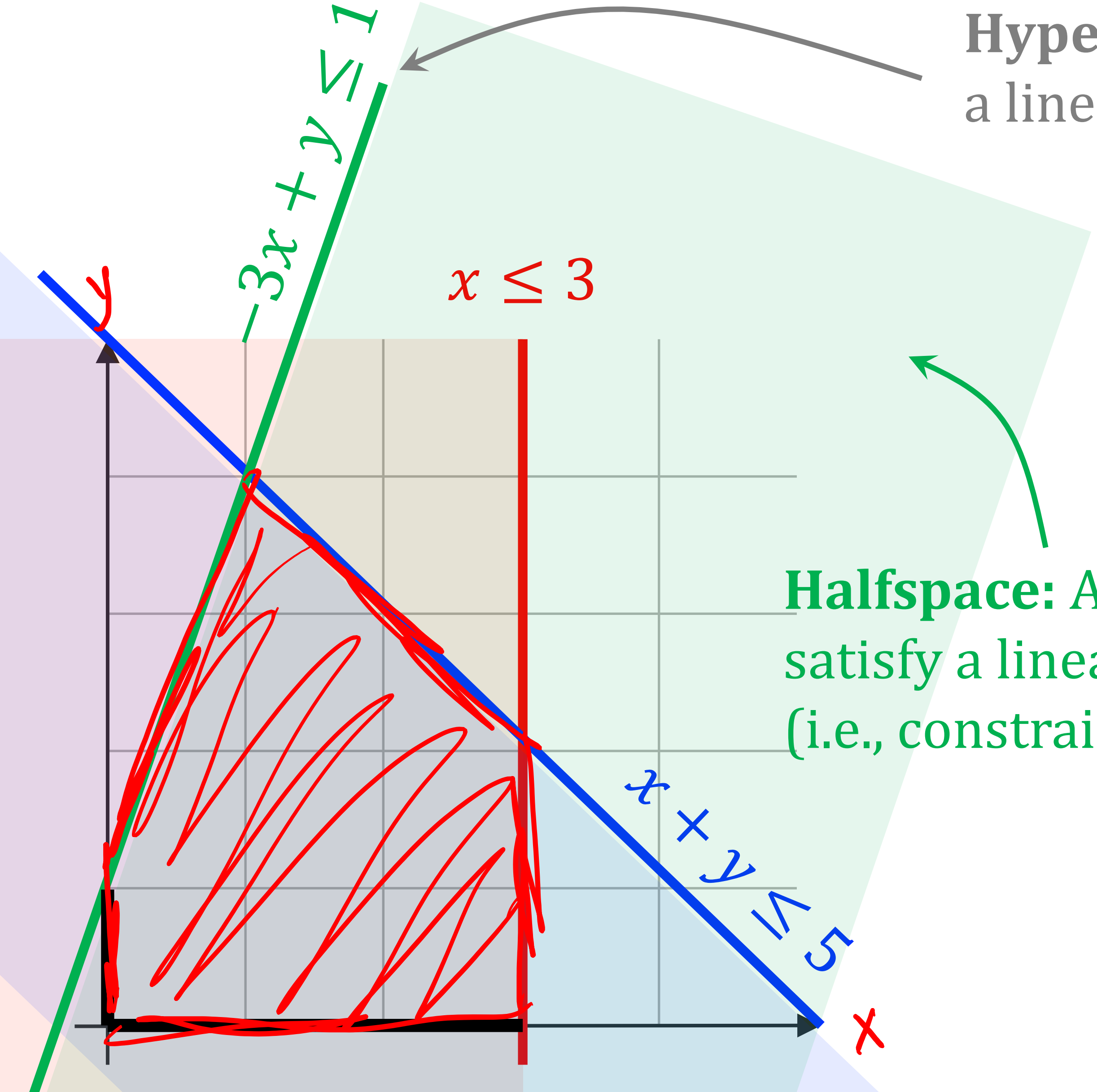
for all rooms r: $\sum_{c:(c,r)\in E} x_{c,r} \le 1$

for all courses c: $\sum_{r:(c,r)\in E} x_{c,r} \le 1$

objective: $\max \sum_{(c,r)\in E} x_{c,r}$

Goal: Maximize the number of courses allocated a classroom.

Geometric View of LPs



Hyperplane: A set of all points that satisfy a linear equality constraint, $\mathbf{a}^\top \mathbf{x} = b$.

Halfspace: A set of all points that satisfy a linear inequality constraint (i.e., constraint $\mathbf{a}^\top \mathbf{x} \leq b$)

My objective

$\max\ x + 2y$

Constraints

$x \leq 3$

$x + y \leq 5$

$-3x + y \leq 1$

$x \geq 0, y \geq 0$

Feasible Region

$\mathbb{R}^n$ space.

Feasible Region: The set of all points that satisfy **all** constraints.

My objective

max $x + 2y$

Constraints

$x \leq 3$

$x + y \leq 5$

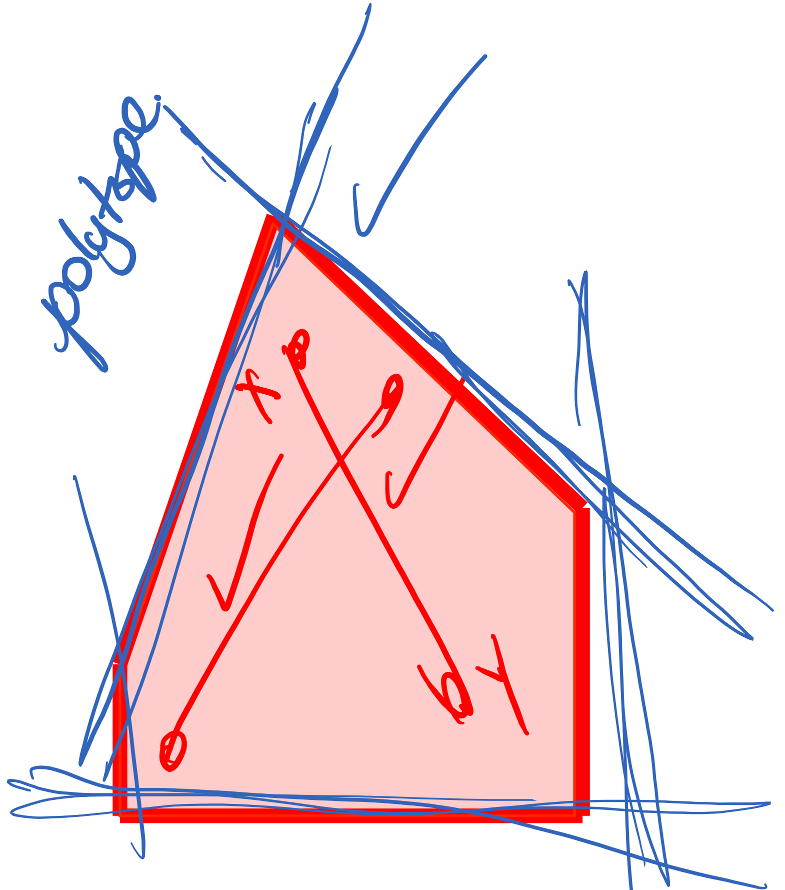
$-3x + y \leq 1$

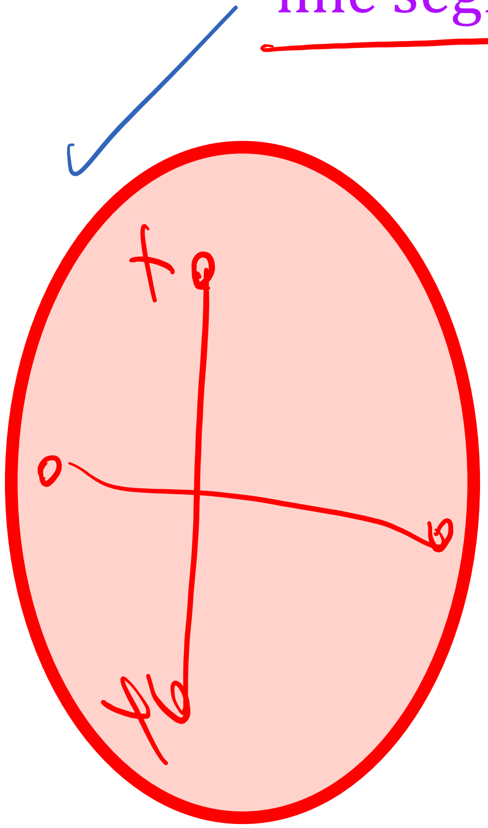
$x \geq 0, y \geq 0$

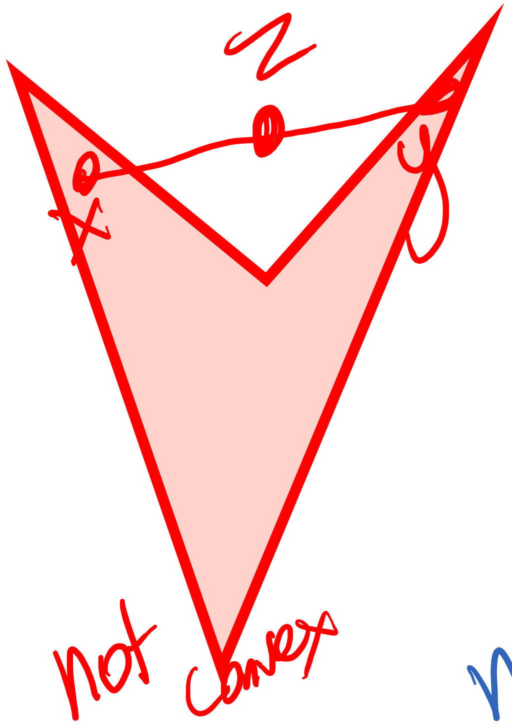
Feasible Region and Convexity

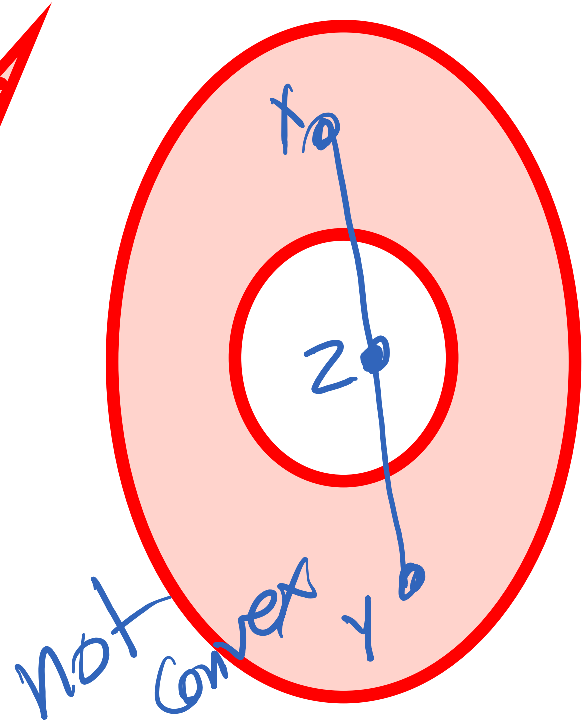
Definition: A set of points $S \subseteq \mathbb{R}^n$ is convex, if for any two points $x, y \in S$, all points $z = \alpha x + (1-\alpha) y$ for $\alpha \in [0,1]$ also satisfy $z \in S$

line segment between x and y.







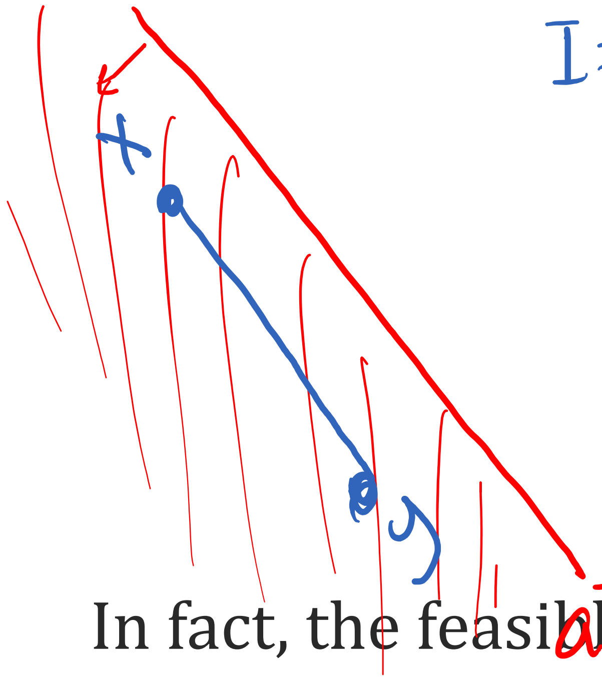


Feasible Region and Convexity

Show this is convex.

Claim: In any linear program, the feasible region is a convex set.

Sketch: feasible solution $\vec{x}$ $\Longleftrightarrow$ $\boxed{\vec{x} \text{ meets every constraint } a_i^T x \le b_i \text{ for } i \in [m].}$



If x, y meet constraint i: $\boxed{\begin{matrix} a_i^T x \le b_i \\ a_i^T y \le b_i \end{matrix}}$ $\Longrightarrow$ any $z = \alpha x + (1-\alpha) y$, $\alpha \in [0,1]$ also meets the constraint

$$a_i^T \cdot z = \underbrace{a_i^T x}_{\le b_i} \alpha + \underbrace{a_i^T y}_{\le b_i} (1-\alpha) \le b_i$$

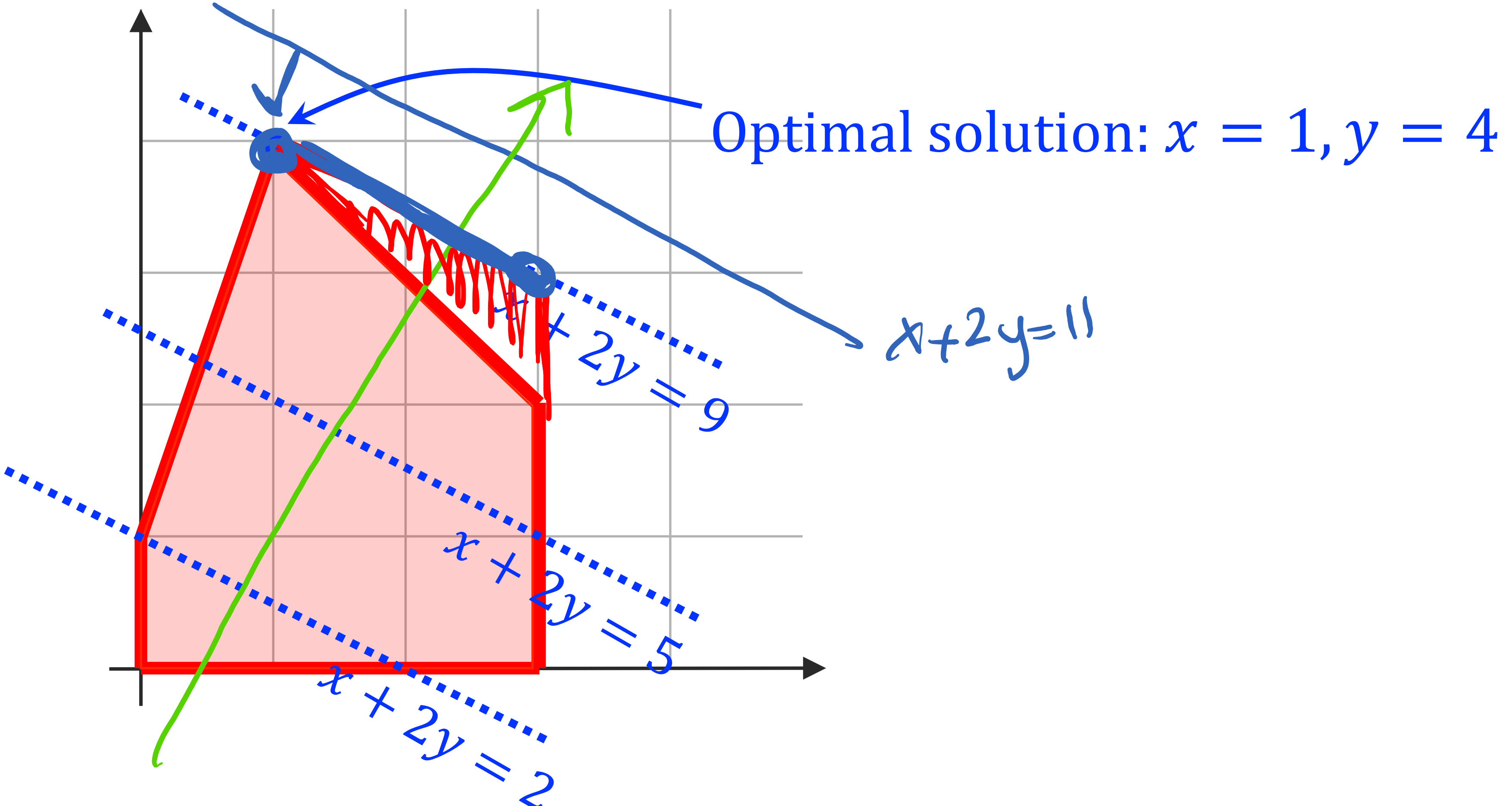
In fact, the feasible region is a "convex polytope": Intersection of halfsapces.

$a_i^T x \le b_i$

Geometric view of the optimal solution

Consider the level-sets of the objective function (All points that lead to the same objective value)

→ $x + 2y = c$ for all c.



My objective

$\max\ x + 2y$

Constraints

$x \leq 3$

$x + y \leq 5$

$-3x + y \leq 1$

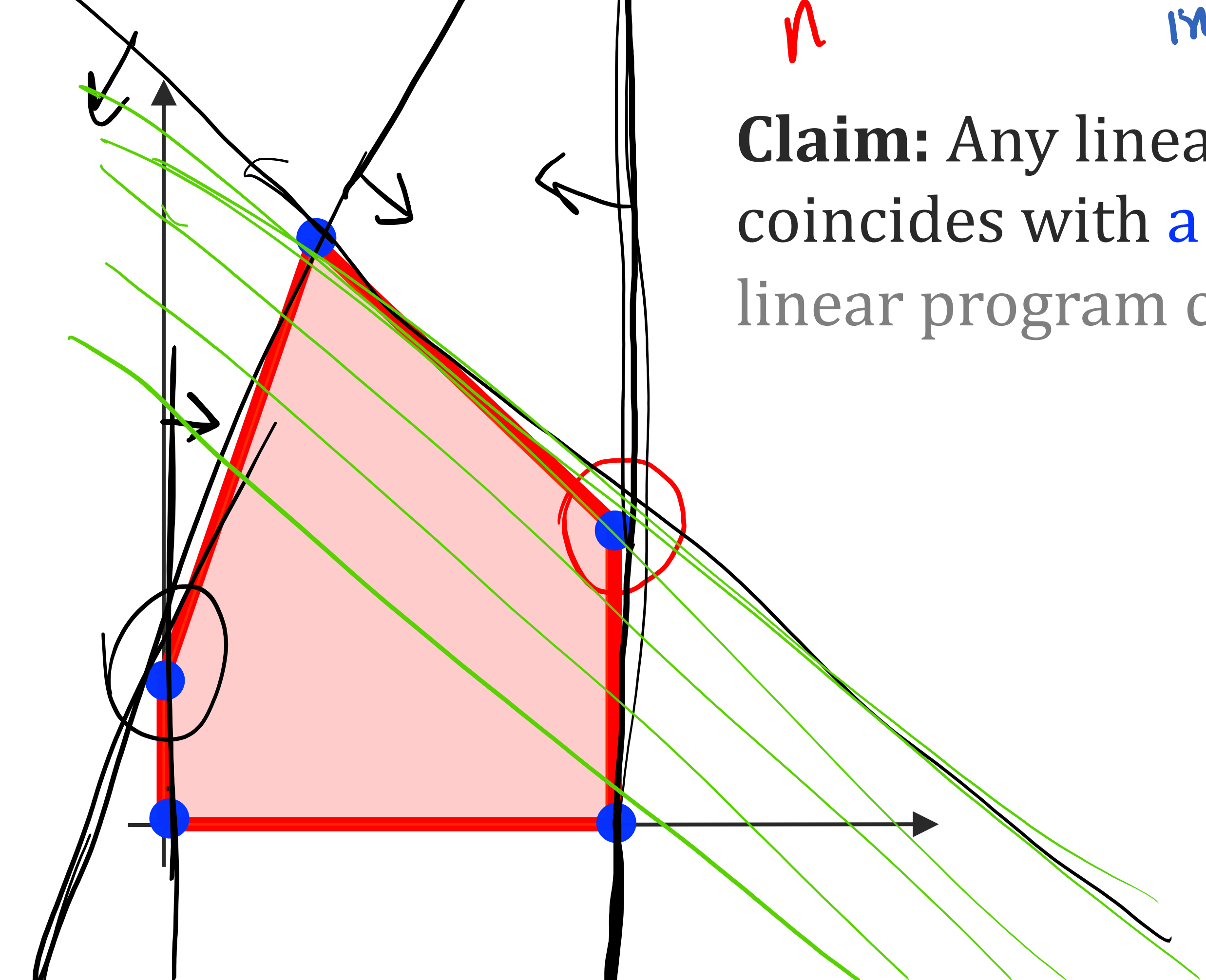
$x \geq 0, y \geq 0$

not a vertex

Optimal solutions and extreme points / vertex -

Definition: An vertex (or extreme point) of the feasible region is a point that falls at the intersection of ~~d~~ n hyperplanes, each corresponding to a constraint.

in $\mathbb{R}^n$



Claim: Any linear program has an optimal solution that coincides with a vertex of the feasible region (or else, the linear program can achieve unbounded value).

One Algorithm for finding all vertices

An algorithm for finding all vertices of the feasible region:

Given m constraints
For each subset of n constraints:

1. Solve for x^* at their intersection
   → Linear system: replace $\leq$ with $=$ and use Gaussian elimination.
2. Check to see if x^* is in the feasible region
   → Check all other $m - n$ constraints are held.
   → If so, add x^* to the list of extreme points.

$$\max \boldsymbol{c}^\top \mathbf{x}$$

$$\text{Subject to } \mathrm{A}\mathbf{x} \leq \mathbf{b}$$

$$\text{and } \mathbf{x} \geq \mathbf{0}$$

A: m$\times n$ matrix
$\boldsymbol{c}$, $\mathbf{x}$: n-dim column vectors
$\mathbf{b}$: m-dim column vectors

LP Algorithm 1

Go through all vertices and find the one with highest payoff $\boldsymbol{c}^\top \mathbf{x}$.

Discuss

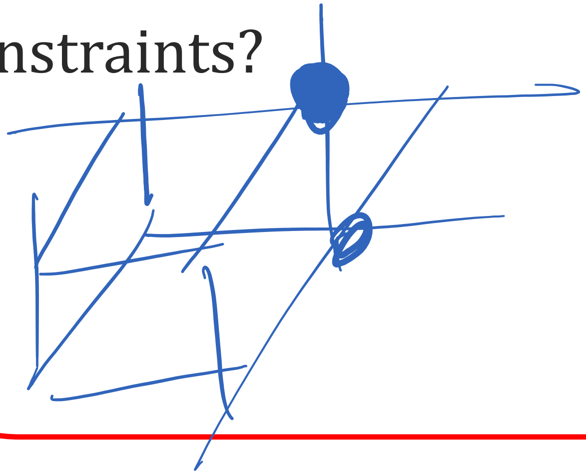
Why would such an algorithm be very slow?

How large can the # of vertices be, given m constraints and n variables?

vertex is intersection of n of the constraints.

$\leq \binom{m}{n} \approx m^n$

Give a construction of a feasible region with 2^n vertices using $m = O(n)$ of constraints?

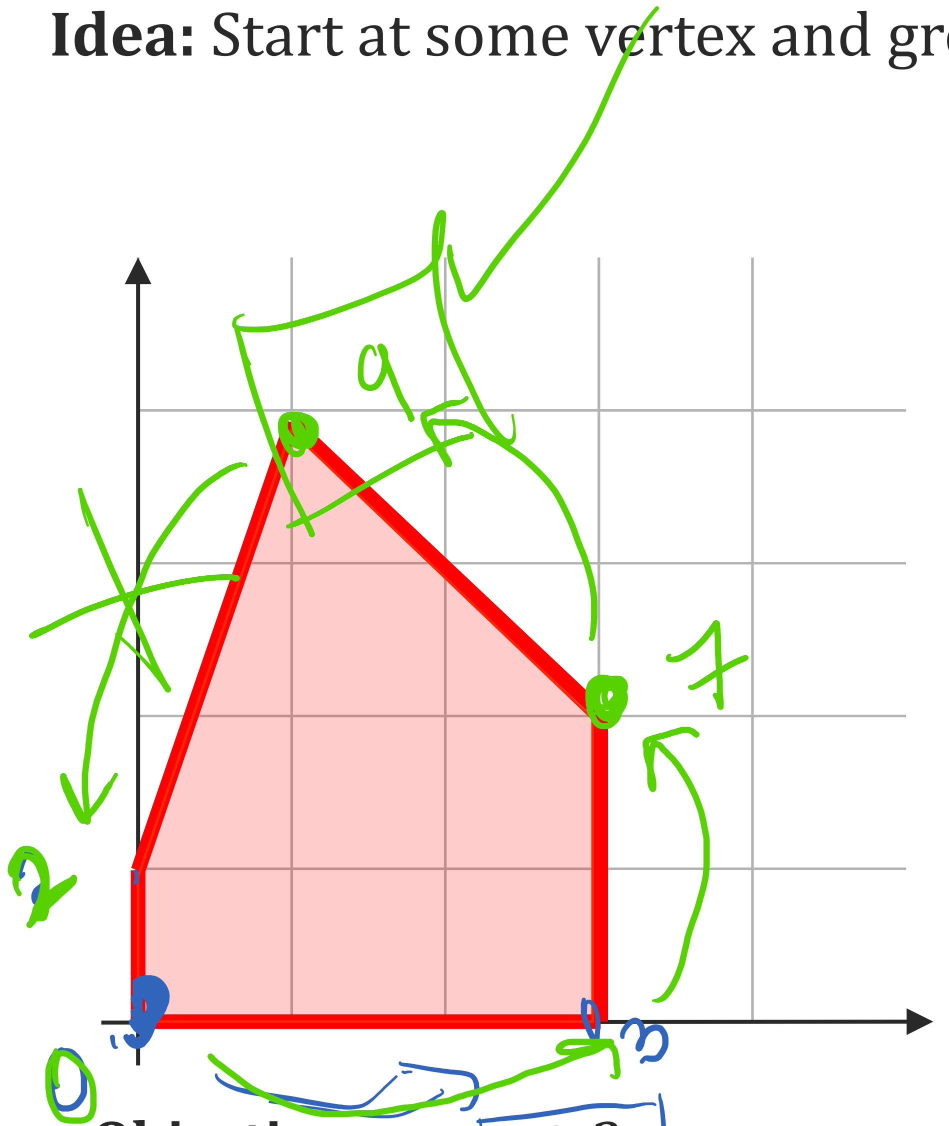


$0 \leq x_i \quad 1 \geq x_i \quad \forall i \Rightarrow 2n$ constraints.

vertices 2^n.

LP Algorithm 2: Simplex

Idea: Start at some vertex and greedily move to a better neighboring vertex.



Objective: max $\boxed{x + 2y}$

1. Start at a vertex x^*

2. Look at all "neighboring" vertices.

3. Find the neighbor y^* with the best objective value.

4. If y^* has a better value than x^*, move to y^* and repeat. Otherwise return x^*.

Simplex: Defining Neighboring Vertices

How do we define "neighboring" vertices in high dimension?

Imagine we have $n = 3$ variables and $m = 5$ constraints (call the constraints 1, 2, ..., 5).
Recall: vertices are defined as the intersection of n constraints
→ Take vertex at the intersection of constraints $\boxed{\{1,2,3\}}$
→ One way to define neighbors: Swap out one constraints only
→ Neighbors: {1,2,4}, {1,2,5}, {1,4,3}, {1,5,3}, {4,2,3}, {5,2,3}

vertex at intersection of constraint 1,2,4

Definition: Formally, two vertices are **neighbors** if they differ only in **one constraints**.

Discuss

Suppose we have n variables and m constraints. What is the largest number of neighbors any vertex can have?

#neighbors $\leq m \times n$.

Simplex: Great in Practice, Slow in the Worst-Case

Facts about simplex: Given n variables and m constraints
1. In the worst-case, Simplex still runs in time exponential in n.
2. In practice:
→ Simplex runs very fast.
→ Beats more algorithms that are provably polynomial time!
→ Impressive, given that it's 75 years old algorithm!

George Dantzing
Berkeley Math Ph.D

How to explain the performance of Simplex algorithm in practice?
- Smoothed Analysis framework (20 years ago)
- If a bit of Gaussian noise is added to the constraints, Simplex provably runs in polynomial time!
- Smoothed analysis → Useful theoretical tool for addressing runtime in practice!

Polynomial time Algorithms for LPs.

Theorem: Linear Programs can be solved in Polynomial Time, even in the worst-case.

[Khachian 1979]: Ellipsoid Method
→ Good in theory, kind of slow in practice (way slower than Simplex in practice)

[Karmarkar 1984]: Interior point method
→ Good in theory, reasonably good in practice

Advanced algorithms: we won't give a description or prove these claims in CS170.

But you can use them to prove that any problem that can be written as an LP can be solved in polynomial time!

Linear Programs and the World

A Soviet Discovery Rocks World of Mathematics

By MALCOLM W. BROWNE

A surprise discovery by an obscure Soviet mathematician has rocked the world of mathematics and computer analysis, and experts have begun exploring its practical applications.

Mathematicians describe the discovery by L.G. Khachian as a method by which computers can find guaranteed solutions to a class of very difficult problems that have hitherto been tackled on a kind of hit-or-miss basis.

Apart from its profound theoretical interest, the discovery may be applicable in weather prediction, complicated industrial processes, petroleum refining, the scheduling of workers at large factories, secret codes and many other things.

"I have been deluged with calls from virtually every department of government for an interpretation of the significance of this," a leading expert on computer methods, Dr. George B. Dantzig of Stanford University, said in an interview.

The solution of mathematical problems by computer must be broken down into a series of steps. One class of problem sometimes involves so many steps that it could take billions of years to compute.

The Russian discovery offers a way by which the number of steps in a solution can be dramatically reduced. It also offers the mathematician a way of learning quickly whether a problem has a solution or not, without having to complete the entire immense computation that may be required.

According to the American journal Sci-

Continued on Page A20, Column 3

"All the News That's Fit to Print"

The New York Times

LATE CITY EDITION

NEW YORK, WEDNESDAY, NOVEMBER 7, 1979

25 CENTS

IRAN'S CIVIL GOVERNMENT OUT; HOSTAGES FACE DEATH THREAT; OIL EXPORTS BELIEVED HALTED

Similar Transportation Bond Issues Are Passed in New York and Jersey

CARTER SAID TO ASK FOR LOYALTY CHECK OF AIDES ON POLITICS

F.A.A. Seeks Fine of $1.5 Million For 'Unairworthy' Braniff Flights

Main Iran Oil Port Reported Closed And Prices of Spot Petroleum Soar

Energy Dept. Accuses 9 Refiners Of $1.18 Billion Oil Overcharges

Interior Dept. Assailed On Missing a Deadline For Species Protection

A Soviet Discovery Rocks World of Mathematics

Linear Programs and the World

BREAKTHROUGH IN PROBLEM SOLVING

A 28-year-old mathematician at A.T.&T. Bell Laboratories has made a startling theoretical breakthrough in the solving of systems of equations that often grow too vast and complex for the most powerful computers. The discovery, which is to be formally published next month, is already circulating rapidly through the mathematical world. It has also set off a deluge of inquiries from brokerage houses, oil companies and airlines, industries with millions of dollars at stake in problems known as linear programming. These problems are fiendishly complicated systems, often with thousands of variables. They arise in a variety of commercial and government applications, ranging from allocating time on a communications satellite to routing millions of telephone calls over long distances. Linear programming is particularly useful whenever a limited, expensive resource must be spread most efficiently among competing users. And investment companies use the approach in creating portfolios with the best mix of stocks and bonds.

Wrap up

We saw an introduction of Linear Programming!

→ Definitions
→ How to formulate problems as LPs
→ Intuition and some algorithms

Takeaway: Anything that can be written as an LP can be solved in polynomial times (in the number of constraints and variables).

Next lectures: More about linear programming