LECTURE #13

CS 170

Spring 2021

Last time:

More examples of problems solved via dynamic programming (DP):

- Knapsack with repetition
- chain matrix multiplication
- all-pairs shortest paths
- traveling salesperson problem

Please practice DP problems on your own.

Today:

Linear programming: expressing and solving linear optimization problems.

Note: dynamic programming $\neq$ linear programming
[recursion + memoization] [high-dim optimization problem]

- An optimization problem has the following form:
  - n variables $x_1, \dots, x_n \in \mathbb{R}$
  - objective function $f(x_1, \dots, x_n) \in \mathbb{R}$
  - m constraints $C_1, \dots, C_m$ $\quad C_i(x_1, \dots, x_n) \in \{\text{true}, \text{false}\}$

  The goal is:

$$\boxed{\begin{aligned} &\max \; f(x_1, \dots, x_n) \\ &\text{s.t.} \quad \forall i \in [m] \quad C_i(x_1, \dots, x_n) = \text{true} \end{aligned}}$$

- A Linear Programming problem is s.t.

① f is a linear function $f(x_1, \dots, x_n) = a_1 x_1 + \dots + a_n x_n$ for $a_1, \dots, a_n \in \mathbb{R}$

② each C_i is a linear constraint (inequality or equality)

e.g. $C_1(x_1, x_2, x_3, x_4) \quad 5x_1 + x_2 \leq 0$

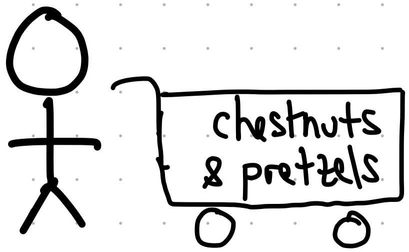
$C_2(x_1, x_2, x_3, x_4) \quad x_1 \geq 0$

$C_3(x_1, x_2, x_3, x_4) \quad x_3 + 2x_4 = 1 \quad (\Leftrightarrow x_3 + 2x_4 \geq 1, \; x_3 + 2x_4 \leq 1)$

Note: "strict" inequalities such as $x_2 < 3$ are NOT allowed

[else it could be that there is no optimal solution even when the set of all solutions is bounded, because the set of possible solutions would not be topologically closed]

Example:



	profit	max daily demand	total space in cart
chestnut box	2	50	100
pretzel	4	80	

The goal is to maximize profit.

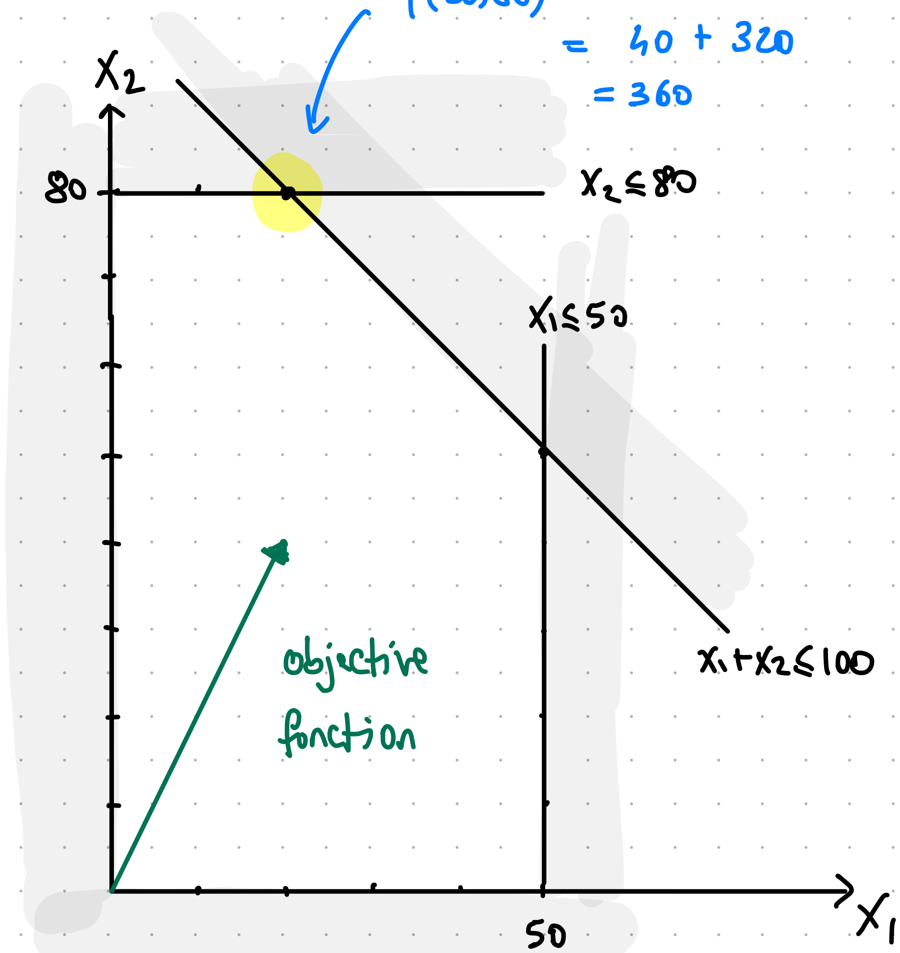
We map the problem to an LP instance:

- variables: $x_1 = \#$ chestnut boxes, $x_2 = \#$ pretzels
  $(n=2)$
- objective function: $f(x_1,x_2) = 2x_1 + 4x_2$
- constraints: $x_1 \geq 0 \quad x_2 \geq 0$
  $(m=5)$
  $x_1 + x_2 \leq 100$
  $x_1 \leq 50$
  $x_2 \leq 80$

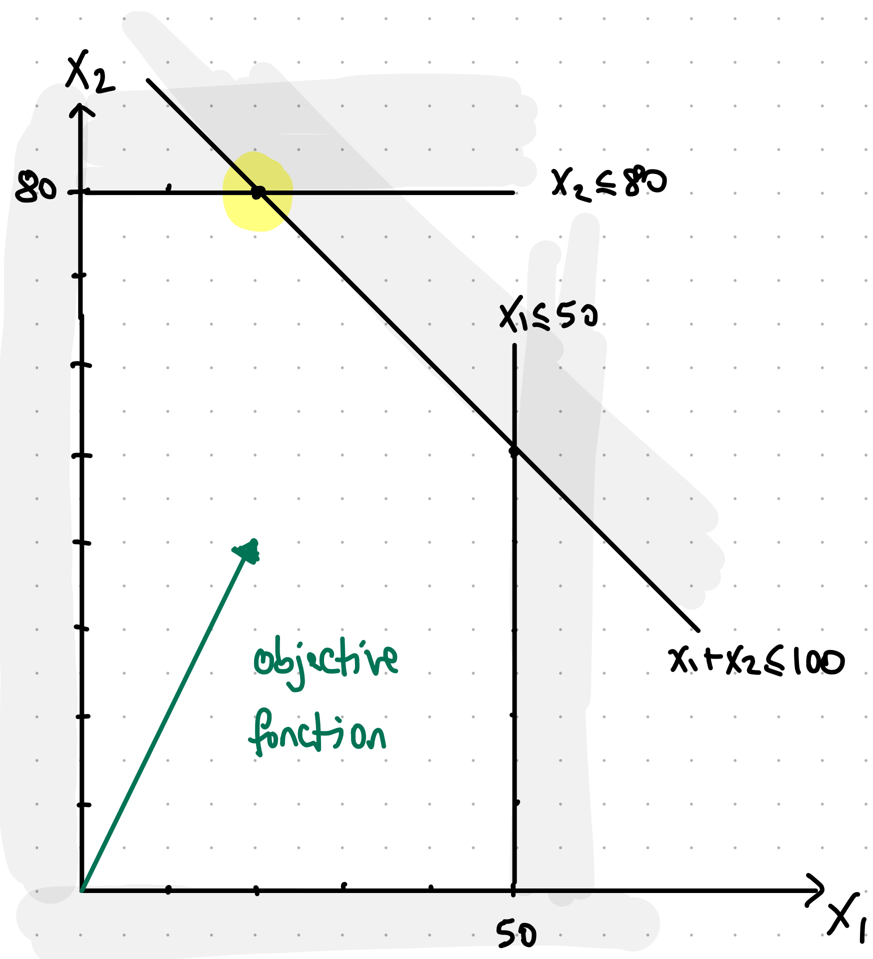
How to solve? Draw it!

$(x_1,x_2) = (20,80)$

$$f(20,80) = 2\cdot 20 + 4\cdot 80 = 40 + 320 = 360$$

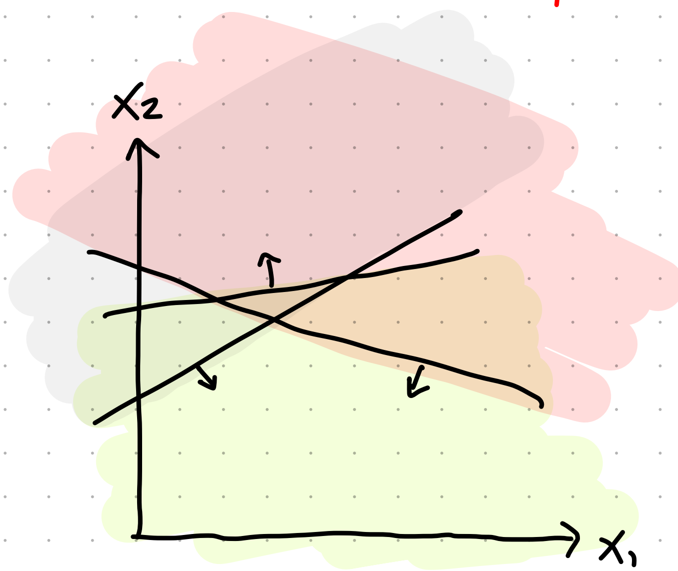


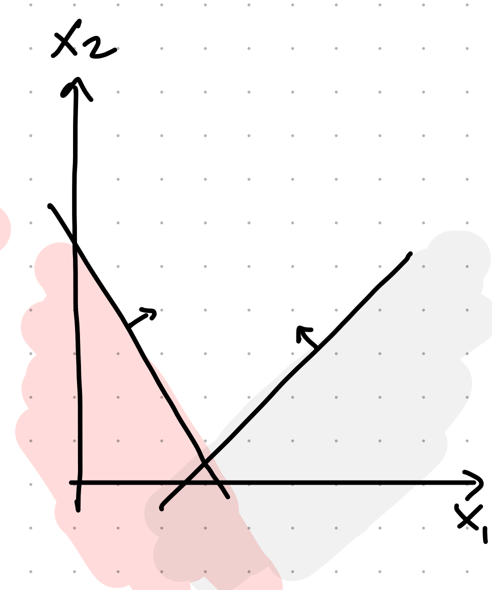
Geometric Observations



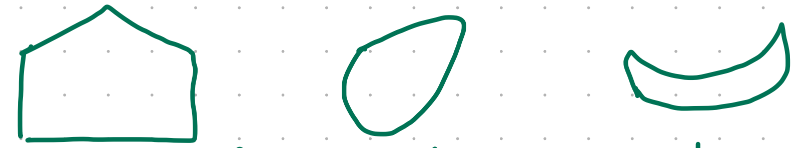
- inequality ↔ half-space
- feasible solutions ↔ intersection of all half-spaces
- objective function ↔ direction
- optimal solution ↔ follow direction in feasible solutions until no longer possible

The set of feasible solutions can also be empty or unbounded.





CONVEXITY



convex convex not convex

claim: set of feasible solutions to a LP is convex:

if $\vec{x}$ and $\vec{y}$ are feasible then $\forall \lambda \in [0,1]$ so is $\vec{z} := \lambda\vec{x} + (1-\lambda)\vec{y}$

Ex: if $x_1, x_2 \geq 0$ & $y_1, y_2 \geq 0$ then $\lambda \in [0,1]$ $\lambda x_1 + (1-\lambda) y_1 \geq 0$, $\lambda x_2 + (1-\lambda) y_2 \geq 0$

$x_1 + x_1 \leq 100$ $\quad$ $y_1 + y_2 \leq 100$ $\quad$ $(\lambda x_1 + (1-\lambda) y_1) + (\lambda x_2 + (1-\lambda) y_2)$

$x_1 \leq 50$ $\quad$ $y_1 \leq 50$ $\quad$ $= \lambda \cdot (x_1 + x_2) + (1-\lambda)(y_1 + y_2)$

$x_2 \leq 80$ $\quad$ $y_2 \leq 80$ $\quad$ $\leq \lambda \cdot 100 + (1-\lambda) \cdot 100 = 100$ $\quad$ et cetera

proof of claim: Follows from linearity of constraints.

$$\left.\begin{array}{l} \vec{a}^T \vec{x} \leq b \\ \vec{a}^T \vec{y} \leq b \\ \lambda \in [0,1] \end{array}\right\} \vec{a}^T (\lambda \vec{x} + (1-\lambda)\vec{y}) = \lambda \vec{a}^T \vec{x} + (1-\lambda)\vec{a}^T \vec{y} \leq \lambda b + (1-\lambda) b = b$$

∎

Hence one of the following holds:

1. no feasible solutions (e.g. $x_1 \geq 1$, $x_1 \leq -3$)
2. no optimum in unbounded space (e.g. max $x_1 + x_2$ s.t. $x_1, x_2 \geq 0$)
3. optimum is a vertex of the polytope ⊛

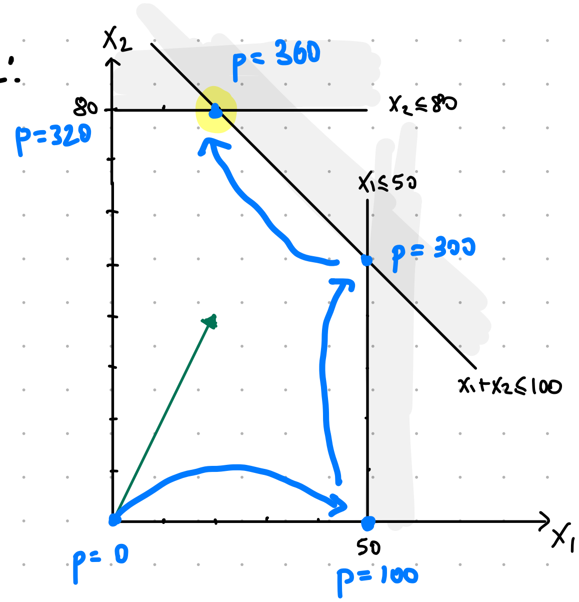
⊛
- if in the interior then can improve
- if on an edge then can move along edge [either improves or the same]

Solving Linear Programs

greedy strategy

Dantzig 1947: simplex method
1. start at any vertex of the polytope
2. move to any adjacent vertex w/ better value
3. if there is no such vertex, output current vertex

For the max-profit example:



Correctness follows from convexity.

Intuitively the greedy strategy works because there are no local maxima.

Say we are at a vertex v and all of v's neighbors have the same or worse value.

Consider the profit line passing through v.

The rest of the feasible region is below this line

So v must be optimal.

Efficiency of the simplex method

In practice the simplex method has excellent efficiency.

Not all these vertices may be feasible, but it could be this many.

But what can we prove?

We know that

$$\# \text{ vertices in polytope} \leq \binom{m}{n}$$

because a vertex consists of n constraints at equality.

Good news: the simplex method terminates in finite time

Bad news: the above bound is huge (exponential in input size)

There are many variations of the simplex method on how to choose next vertex, and it is an open problem if there is a variation making it run in polynomial time.

But there are other algorithms that run in polynomial time:

ellipsoid method, interior point method

Another example (it's important to practice mapping a problem to LP)

A company making ski helmets wants to minimize total costs.

- monthly demand $d_1, \dots, d_{12}$
- 50 employees each costing 4k/month, producing 20 helmets/month
- overtime is 1.5x more expensive, and can be max 25% more
- hiring is \$300/worker, firing is \$400/worker
- surplus storage is \$10/helmet/month (0 at start and at end)

We can write an LP for this:

- **variables:**

$w_0 = 50$, w_i = # workers in month i

x_i = # helmets made in month i

o_i = # helmets made in month i via overtime

h_i, f_i = # workers hired/fired at start of month i

$s_0 = 0$, s_i = # helmets stored at end of month i

- **constraints**

$x_i = 20 \cdot w_i + o_i$

$w_i = w_{i-1} + h_i - f_i$

$s_i = s_{i-1} + x_i - d_i$

$o_i \leq \frac{1}{4} \cdot (20 \cdot w_i)$

- **objective function:** $\min \; 4000 \sum_i w_i + 300 \sum_i h_i + 400 \sum_i f_i + 10 \sum_i s_i + \frac{3}{2} \cdot \frac{4000}{20} \sum_i o_i$

This is hard to solve by hand, but very efficient on a computer.

A caveat: fractions

We may need to round so the cost will increase correspondingly.

In the example we have large numbers so rounding is not a big issue.
But in general it is a delicate matter

Alternative:

integer linear programming = only integer solutions to a LP

But this problem is NP-complete as we will see later in the course.