# Algorithm Design and Analysis

## Linear Programming (Part I)

# Roadmap for today (and beyond)

- Definition of linear programming and examples
- Linear programs for flows
- Linear programs for two-player zero-sum games
- Overview of algorithms for linear programming
- **LP Part II** (Tuesday): Duality
- **LP Part III** (next Thursday): Simplex and integrality

# Formal definition

**Definition (Linear program):** A linear program consists of

- $n$ real-valued **variables** $x_1, x_2, \dots, x_n$
- A linear ***objective function***, e.g., minimize/maximize $2x_1 + 3x_2 + x_3$
- $m$ linear **inequalities**, e.g., $3x_1 + 4x_2 \leq 6$, or $0 \leq x_1 \leq 3$

**Goal:** Find values for $x$'s that satisfy the constraints and minimize/maximize the objective

# Example: Plant production problem

**Example Problem (Plant production):** You run 4 production plants for making cars. Each is different and requires a certain amount of **labor** per car produced, **materials** per car produced, and **pollution** caused per car produced.

- Need to produce at least 400 cars at plant 3 (contract)
- We have 3300 hours or labor available
- We have 4000 units of materials available
- We are allowed to produce 12000 units of pollution
- Goal is to maximize the number of cars produced

| | labor | materials | pollution |
|---|---|---|---|
| plant 1 | 2 | 3 | 15 |
| plant 2 | 3 | 4 | 10 |
| plant 3 | 4 | 5 | 9 |
| plant 4 | 5 | 6 | 7 |

**Variables:**

**Objective:**

**Constraints:**

**Note: We are not guaranteed to get an integer solution! The optimal solution might be to produce 213.5 cars at a plant.**

# Important notes

- Linear programs ***can not*** contain strict inequalities, e.g., $x_1 < 3$
- Linear equalities are okay, e.g., $x_1 + x_2 = 3$
- **Super Important:** Linear programs ***can not*** guarantee integer-valued solutions! Variables can be assigned fractional values.
- Objective and inequalities ***must be linear***, e.g., can't have $x_1 x_2 \leq 3$
- Sometimes you don't need an objective function. Any solution that satisfies the constraints is good. This is called the "*feasibility problem*"

# Classification of solutions

There are three possible outcomes:

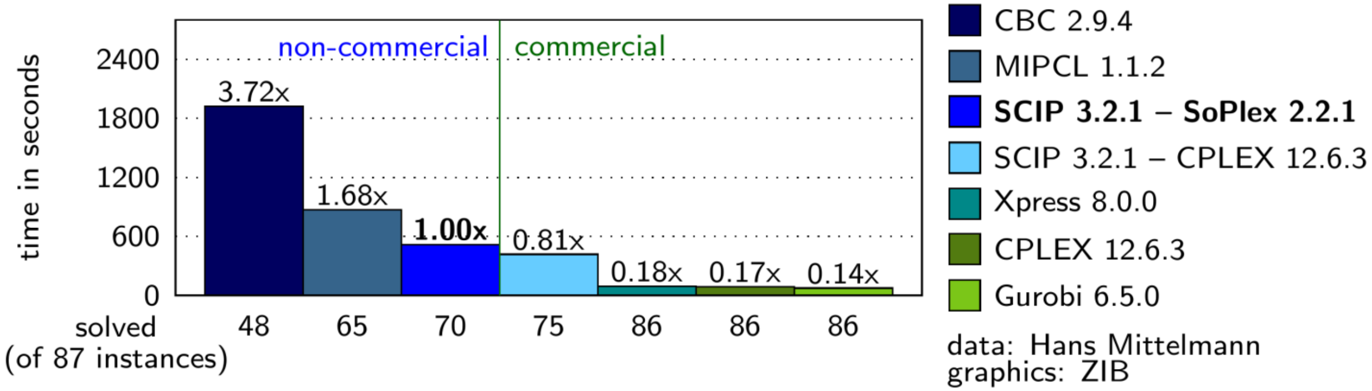
- **Infeasible**: There is no solution that satisfies the constraints
- **Feasible and bounded**: There is a solution of maximum objective value
- **Feasible and unbounded**: There are solutions of arbitrarily large value

# Linear programming in practice

- Linear programming is the basis of tremendously ***many real-world optimization problems***
- Business schools teach entire courses on linear programming, and have entire departments dedicated to optimization / OR.
- Linear programming underpins more general tools like ***integer programming***, ***constraint programming***

# Linear programming in practice

- Commercial tools for linear/integer/constraint programming cost 10s of thousands of $$$
  - Cplex, Gurobi, Xpress
- Lots of open-source tools exist, but they are substantially slower than the commercial alternatives
  - SCIP, MIPCL, Coin-OR Branch-and-Cut (CBC), ***Google OR-Tools***
- Commercial solvers are ***5x faster*** than open source…



# Example: Max flow

- How would we model the maximum flow problem as an LP?

**Variables:**

**Objective:**

**Constraints:**

# Example: Minimum-cost Max flow

- Sounds tricky, there's two objectives! (Min cost, and max flow)
- **Solution #1**: Maximize flow first

**Variables:**

**Objective:**

**Constraints:**

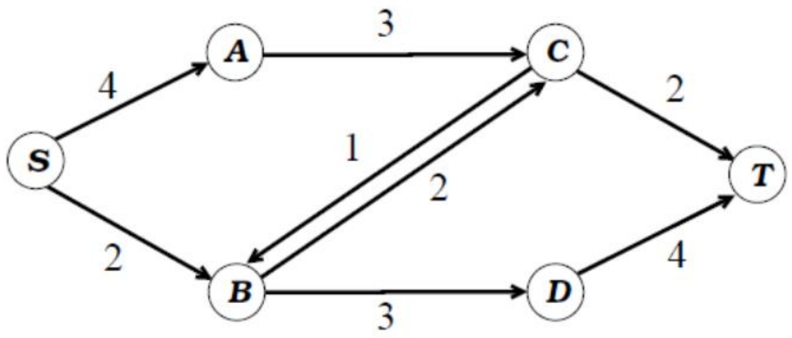
# Example: Minimum-cost Max flow

- Sounds tricky, there's two objectives! (Min cost, and max flow)
- **Solution #2**: Reduce to "minimum-cost circulation"

**Variables:**

**Objective:**

**Constraints:**



# Example: 2-player zero-sum games

**Variables:**

**Objective:**

**Constraints:**

| | column player | | |
|---|---|---|---|
| row player | 20 | -10 | 5 |
| | 5 | 10 | -10 |
| | -5 | 0 | 10 |

$$\max_{p} \min_{q} V_R(p, q)$$

# Standard form

- The same LP can be written in many ways
- It is convenient to have a "standard way" to write an LP

**Definition (Standard Form):** An LP with $n$ variables $x_1, \dots, x_n$ and $m$ constraints in **standard form** can be written with constants $c_1, \dots, c_n$, $b_1, \dots b_m$, $a_{11}, \dots, a_{mn}$

*Objective must be max, not min*

*Constraints are all $\leq$ constant*

*All variables are non-negative. (These do not count towards the $m$ number of constraints!)*

$$\begin{aligned} \textbf{maximize} \quad & c_1 x_1 + \cdots + c_n x_n \\ \textbf{subject to} \quad & a_{11} x_1 + \cdots + a_{1n} x_n \leq b_1 \\ & a_{21} x_1 + \cdots + a_{2n} x_n \leq b_2 \\ & \qquad \vdots \\ & a_{m1} x_1 + \cdots + a_{mn} x_n \leq b_m \\ & x_i \geq 0 \quad \text{for all } i \end{aligned}$$

$=$

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

# Converting to standard form

**Claim:** Every LP that is not written in standard form can be converted to an equivalent LP in standard form

- ***How to convert minimization to maximization?***
- ***How to convert a $\geq$ constraint?***
- ***How to convert an $=$ constraint?***
- ***How to convert a variable $x_i$ which could negative?***

*Objective must be max, not min*

*Constraints are all $\leq$ constant*

*All variables are non-negative.*

# The geometry of linear programming

- What does a linear program look like geometrically?

# The geometry of linear programming

- Each constraint is a ***halfspace***: It cuts $\mathbb{R}^n$ into two halves
- The intersection of all the halfspaces is the ***feasible region***
- If the feasible region is empty, the LP is ***infeasible***
- The feasible region can be unbounded
- Maximizing $c^T x$ moves a hyperplane with normal vector $c$ until it is tangent to the feasible region

# Convexity

- The feasible region $Ax \leq b,\ x \geq 0$ is a convex set
  - (If $p$ and $q$ are feasible, then so is any point on the line segment $pq$)
  - Why? A halfspace is convex, and the intersection of convex sets is convex

# Algorithms for Linear Programming

- ***The simplex algorithm***
  - The first, most famous, and very practical.  Exponential worst-case time.
- ***Ellipsoid algorithm***
  - First discovered polynomial time algorithm.  Slow in practice.
- ***Karmarkar's Algorithm (interior-point method)***
  - More efficient polynomial-time algorithm

# The Simplex Algorithm

- The feasible region is a polytope in $\mathbb{R}^n$
- Pick a starting ***vertex*** of the polytope
- Move to highest-cost neighbouring vertex
- Repeat until maximum
- Worst-case complexity is exponential.

**Fun fact: The typical simplex implementation doesn't use standard form. It uses so-called "equational form", where you write maximize $\boldsymbol{c^T x}$ subject to $\boldsymbol{Ax = b}$, where $\boldsymbol{x \geq 0}$**

**Exercise (Starting point):** How do we find a starting point for the simplex algorithm? Given an LP, can you find another LP with a trivial starting point whose solution gives a feasible point in the original LP?

# The Ellipsoid Algorithm

- Determines whether an LP is feasible (doesn't solve for optimal)
- Find an ellipse that contains the feasible region
- Check if the center is a feasible point
- If not, shrink the ellipse and repeat
- Complexity is $O(n^6\ w)$

**Exercise (Optimizing):** Given an algorithm that can determine if an LP is feasible (but does not find an optimal solution), how can we use it as a black box to find an optimal solution of an LP?

# Karmarkar’s Algorithm

- Works with feasible points but doesn’t go corner to corner
- Moves in interior of the feasible region – “interior point method”

# Take-home messages

- LPs have tremendously many applications, many theoretically interesting, and many lucrative.
- Many interesting problems we learned about earlier in the course can be expressed/solved using LPs (flows, games)