Portland State University

W'21 CS 584/684

Algorithm Design & Analysis

Fang Song

Lecture 16

- Linear programming
- Intractability

Another formulation of max-flow problem

Recall. An $s\text{–}t$ flow is a function $f: E \to \mathbb{R}$ satisfying

- [Capacity] $\forall e \in E: 0 \le f(e) \le c(e)$
- [Conservation] $\forall v \in V \setminus \{s,t\}$: $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$

The value of a flow f is $v(f) \coloneqq \sum_{e \text{ out of } s} f(e)$

Max–Flow Problem

Real-value variables $\vec{f} = \{f_e : e \in E\}$

Maximize: $v(\vec{f})$

Subject to:

$$0 \le f_e \le c(e), \ \forall e \in E$$

$$\sum_{e \text{ into } v} f_e - \sum_{e \text{ out of } v} f_e = 0, \ \forall v \in V \setminus \{s,t\}$$

Linear constraints: no $x^2, xy, \sin(x)$, ...

Grade maximization

Input. HW from two courses (xxx & 584/684) due in one day

- Every hour you spend, you earn 1pts on xxx or 5pts on 584/684
- Your brain will explode if you work more than 12hrs on xxx or 15hrs on 5/684
- Of course, there are only 24 hrs in a day

Goal. Maximize the total pts you can earn

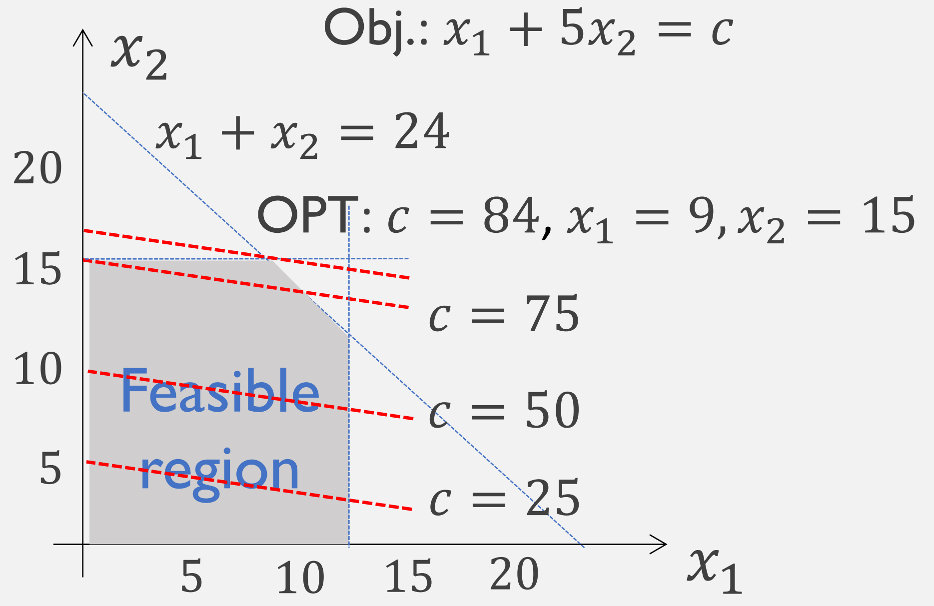
Grade—Maximization

Variables: x_1 (xxx hrs); x_2 (5/684 hrs)

Maximize: $x_1 + 5x_2$

Subject to: // linear constraints

$$0 \le x_1 \le 12$$
$$0 \le x_2 \le 15$$
$$x_1 + x_2 \le 24$$



Linear programming

Linear programming. Optimize a linear objective function subject to linear inequalities.

- Formal definition and representations
- Duality
- Algorithms: simplex, ellipsoid, interior point

Why significant?

- Design poly-time algorithms & approximation algorithms
- Wide applications: math, economics, business, transportation, energy, telecommunications, and manufacturing

Ranked among most important scientific advances of 20th century

Linear programming

- "Standard form" of an LP
  - m = # constraints, n = # decision variables. $i = 1, \dots, m, j = 1, \dots, n$
  - Input: real numbers c_j, a_{ij}, b_i
  - Output: real numbers x_j
  - Maximize linear objective function subject to linear inequalities
  - Feasible vs. optimal soln's.

Max $\sum_{j=1}^{n} c_j x_j$
Subject to: // linear constraints

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad 1 \le i \le m$$

$$x_j \ge 0 \quad 1 \le j \le n$$

$\equiv$

Max $\boldsymbol{c}^T \boldsymbol{x}$
Subject to: $A\boldsymbol{x} \le \boldsymbol{b}$
$\boldsymbol{x} \ge \boldsymbol{0}$

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

$$\boldsymbol{c} = \begin{pmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{pmatrix} \quad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \quad \boldsymbol{0} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \end{pmatrix}$$

Linear programming: variants

- "Slack form" of an LP: linear equalities

Max $\sum_{j=1}^{n} c_j x_j$

Subject to: // linear constraints

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad 1 \le i \le m$$

$$x_j \ge 0 \quad 1 \le j \le n$$

$\Rightarrow$

Max $\sum_{j=1}^{n} c_j x_j$

Subject to: // linear constraints

$$s_i = b_i - \sum_{j=1}^{n} a_{ij} x_j \quad 1 \le i \le m$$

$$\text{(slack vars) } s_i \ge 0 \quad 1 \le i \le m$$

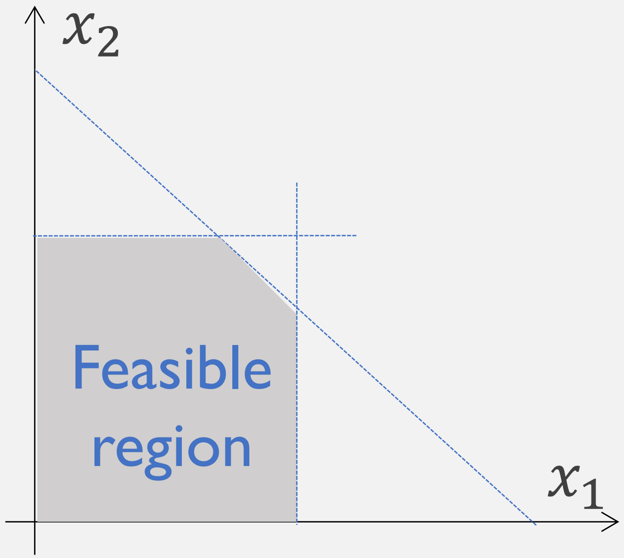
$$x_j \ge 0 \quad 1 \le j \le m$$

- Other equivalent variations
  - Minimization vs. maximization
  - Variables without nonnegativity constraints
  - $\ge$ vs. $\le$

Geometry of linear programming

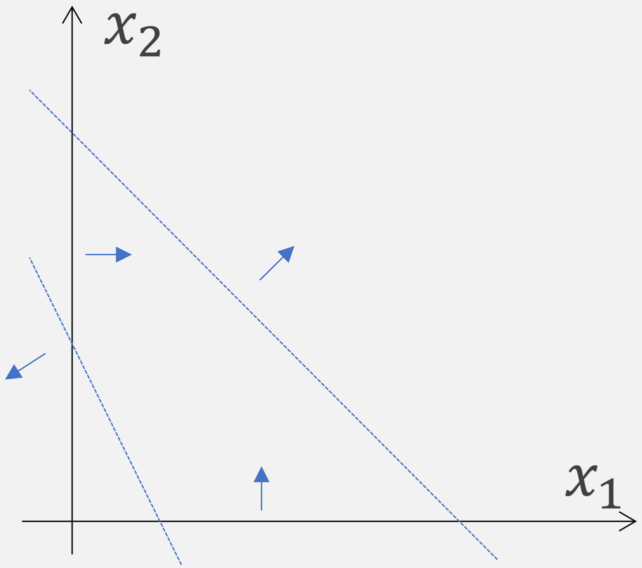
1. Feasible

Maximize: $x_1 + 5x_2$
Subject to:
$$0 \le x_1 \le 12$$
$$0 \le x_2 \le 15$$
$$x_1 + x_2 \le 24$$



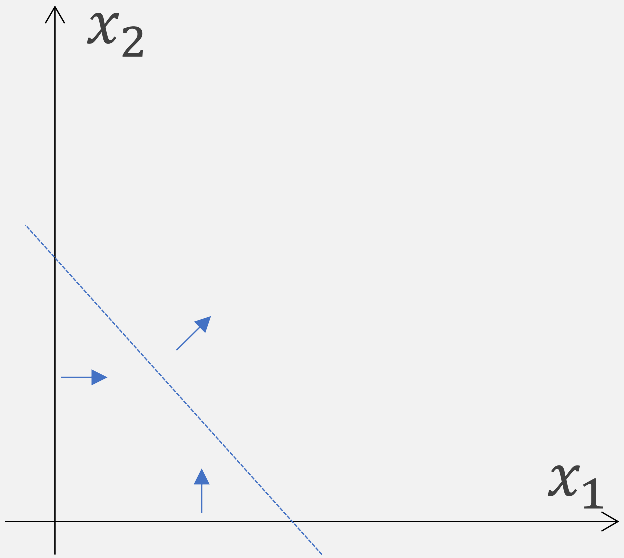
2. Infeasible

Maximize: $x_1 - x_2$
Subject to:
$$2x_1 + x_2 \le 1$$
$$x_1 + x_2 \ge 2$$
$$x_1, x_2 \ge 0$$

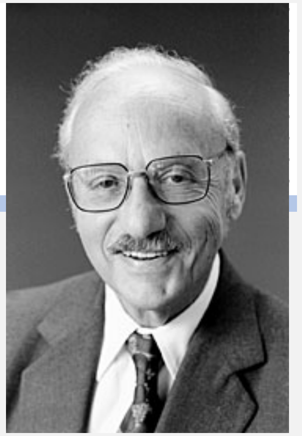


3. Unbounded

Maximize: $2x_1 + x_2$
Subject to:
$$x_1 + x_2 \ge 1$$
$$x_1, x_2 \ge 0$$

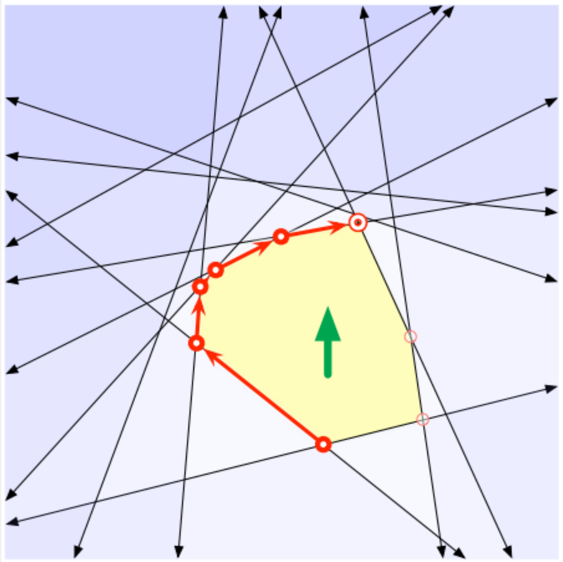


Simplex algorithm: the gist

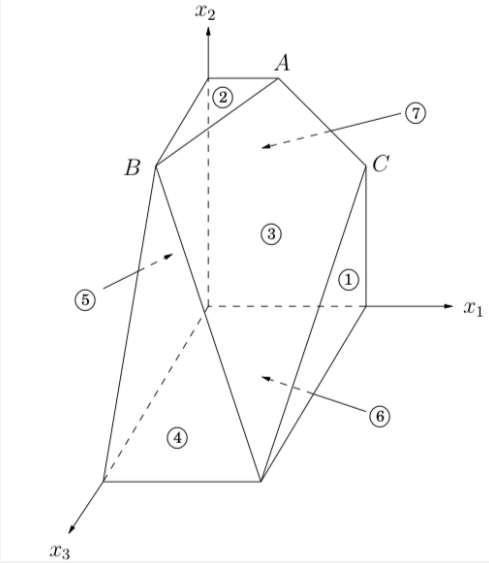


George Dantzig 1947

Let v be any vertex of the feasible region
While there is a neighbor u of v with better obj. value
    $v \leftarrow u$

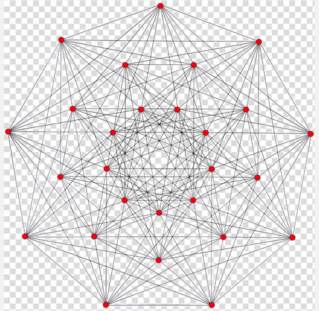


"Hill-climbing" along vertices in the polygon



3D-polyhedron defined by 7 inequalities

n variables?



- A linear eq. defines a hyperplane in R^n
- A linear ineq. defines a halfspace in R^n

- Each vertex is specified by n ineq's
- 2 vertices are neighbors if share $n-1$ defining ineq's

Simplex algorithm: the fine prints

Let v be any vertex of the feasible region
While there is a neighbor u of v with better obj. value
    $v \leftarrow u$

- How to find an initial feasible vertex?
  - Reduced to an LP and solved by simplex!
- Which neighbor to move to? (Pivot)
- Running time? [m ineq's, n variables]
  - ☹ Exponential in worst-case! $\binom{m+n}{n} \geq \left(1+\frac{m}{n}\right)^n$ verticies
  - ☺ Super fast in real world [typically terminates after at most $2(m+n)$ pivots]
- Correctness?
  - Convex polyhedron & linear objective function: local max $\equiv$ global max

Poly-time algorithms for linear programming

- Ellipsoid algorithm [Khachiyan1979]

POLYNOMIAL ALGORITHMS IN LINEAR PROGRAMMING*

L. G. KHACHIYAN

Moscow

  - A mathematical "Sputnik"
  - Not competitive in practice

- Interior point algorithm [Karmarkar1984]

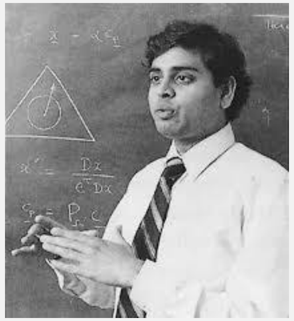
A New Polynomial-Time Algorithm for Linear Programming

N. Karmarkar

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

N.B. Commercial solvers can solve LPs with millions of variables and tens of thousands of constraints.

Leonid Khachiyan

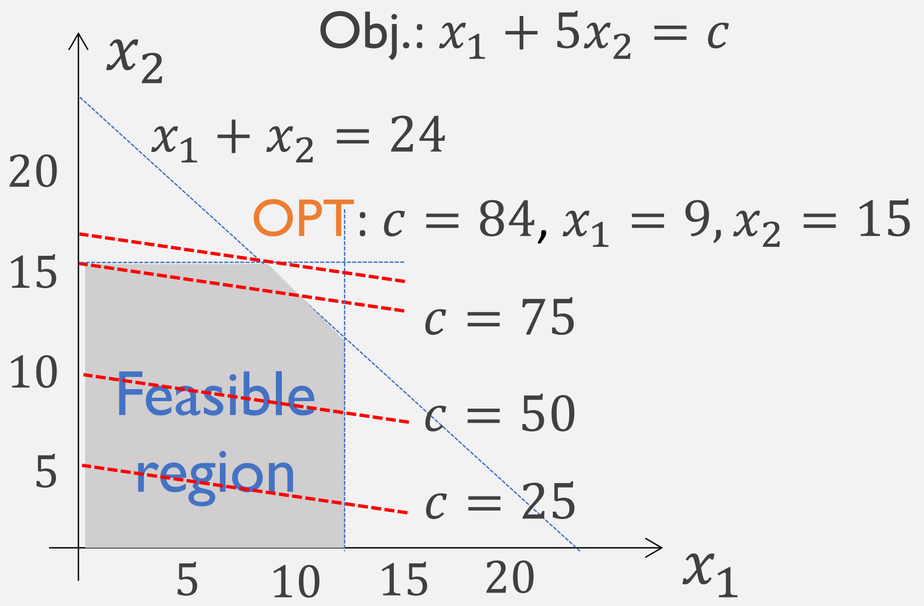


Narendra Karmarkar

How to decide optimality?

(P) Maximize: $x_1 + 5x_2$
Subject to:

$$0 \le x_1 \le 12$$
$$0 \le x_2 \le 15$$
$$x_1 + x_2 \le 24$$



Certificate: $x_1 + 5x_2 = 4 \cdot x_2 + 1 \cdot (x_1 + x_2) \le 4 \cdot 15 + 24 = 84$

How to find these (magic) multipliers?

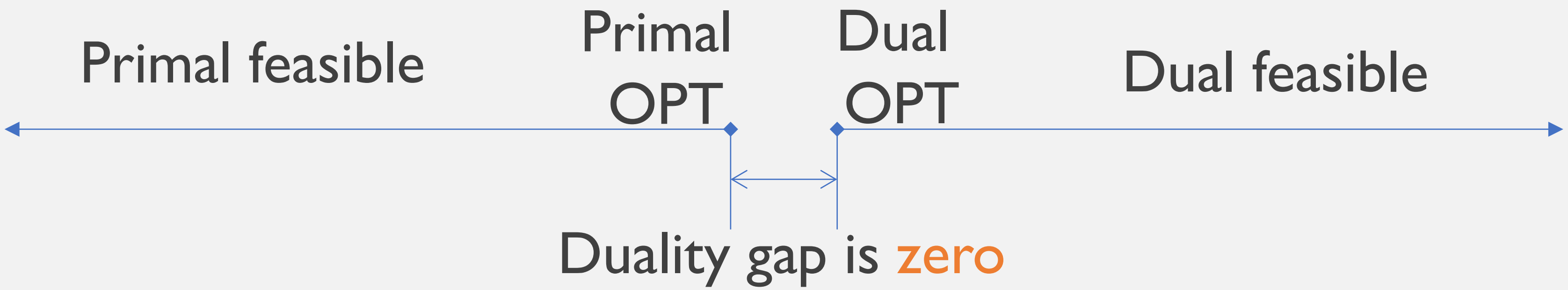
Recall: max-flow & min-cut duality

- Weak duality (certificate of optimality) $v(f) \leq cap(A, B)$

- Strong duality (max-flow min-cut theorem)

Value of max flow = capacity of min cut

Fundamental theorem of linear programming



(Primal) Max $\boldsymbol{c}^T\boldsymbol{x}$
Subject to:
$$A\boldsymbol{x} \le \boldsymbol{b}$$
$$\boldsymbol{x} \ge \boldsymbol{0}$$

(Dual) Min $\boldsymbol{y}^T\boldsymbol{b}$
Subject to:
$$\boldsymbol{y}^T A \ge \boldsymbol{c}^T$$
$$\boldsymbol{y} \ge \boldsymbol{0}$$

- Weak duality. If $\boldsymbol{x}$ is a feasible solution for a linear program $\sqcap$, and $\boldsymbol{y}$ is a feasible solution for its dual $\sqcup$, then $\boldsymbol{c}^T\boldsymbol{x} \le \boldsymbol{y}^T A\boldsymbol{x} \le \boldsymbol{y}^T\boldsymbol{b}$.

- Strong duality. $\sqcap$ has an optimal solution and $\boldsymbol{x}^*$ if and only if its dual $\sqcup$ has an optimal solution $\boldsymbol{y}^*$ such that $\boldsymbol{c}^T\boldsymbol{x} = \boldsymbol{y}^T A\boldsymbol{x} = \boldsymbol{y}^T\boldsymbol{b}$.

Duality example

(P) Maximize: $x_1 + 5x_2$
Subject to:
$$0 \leq x_1 \leq 12$$
$$0 \leq x_2 \leq 15$$
$$x_1 + x_2 \leq 24$$

(D) Minimize: $12y_1 + 15y_2 + 24y_3$
Subject to:
$$y_1 + y_3 \geq 1$$
$$y_2 + y_3 \geq 5$$
$$y_1, y_2, y_3 \geq 0$$

Max $= 84, x_1 = 9, x_2 = 15$

Min $= 84, y_1 = 0, y_2 = 4, y_3 = 1$

(magic) multipliers

Exercise: Multicommodity flow

- A flow network with multiple flows (commodities)
  - $c(e)$: capacity on each edge
  - $K_i = (s_i, t_i, d_i)$: source, sink, and demand of commodity i. $i = 1, \dots, \ell$
- Goal. Decide if it is possible to accommodate all commodities

Max/min: 0

Subject to:

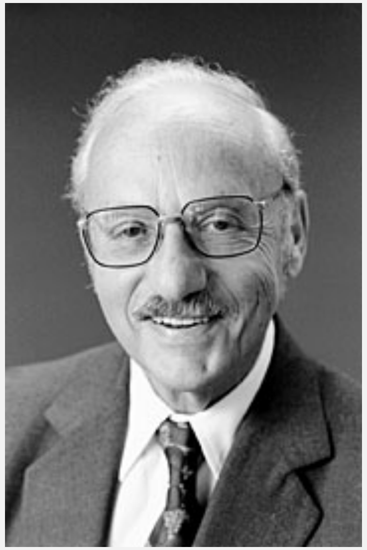
$$f_{ie} \geq 0, \qquad \forall e \in E$$

$$\sum_{i=1}^{\ell} f_{ie} \leq c(e), \qquad \forall e \in E$$

$$\sum\nolimits_{e \text{ into } v} f_{ie} - \sum\nolimits_{e \text{ out of } v} f_{ie} = 0, \qquad \forall v \in V \setminus \{s, t\}$$

$$\sum\nolimits_{e \text{ out of } s_i} f_{ie} - \sum\nolimits_{e \text{ into } s_i} f_{ie} = d_i, i = 1, \dots, \ell$$

A dialogue between Dantzig & von Neumann



George Dantzig

John von Neumann

Let me show you my exciting finding: simplex algorithm for LP
… … … [next 30 mins]

Get to the point, please!

OK! Em... To be concise … [next 3 mins]

Ah, that!

[next 60 mins]
…. (convexity)… (fixed point) … (2-player game) …
so, there is duality which'd follow by my min-max theorem …

For any matrix A, $\min_x \max_y xAy = \max_y \min_x xAy$.

A refection on the algorithmic journey

- So far: algorithm design triumph
  - Divide-and-conquer
  - Greedy
  - Dynamic programming
  - Linear programming (duality)
  - Local search
  - Randomization
  - …

Examples

- $O(n \log n)$ Merge sort
- $O(n \log n)$ interval scheduling
- $O(n^2)$ edit distance
- $O(n^3)$ bipartite matching

- New goal: understand what is hard to compute

Computational intractability

- **Computability:** can you solve it, in principle?

Halting problem is uncomputable [Given program code, will this program terminate or loop indefinitely?]

Church-Turing Thesis. A function can be computed in any *reasonable* model of computation iff. it is computable by a **Turing machine**.

- **Complexity:** can you solve it, under resource constraints?

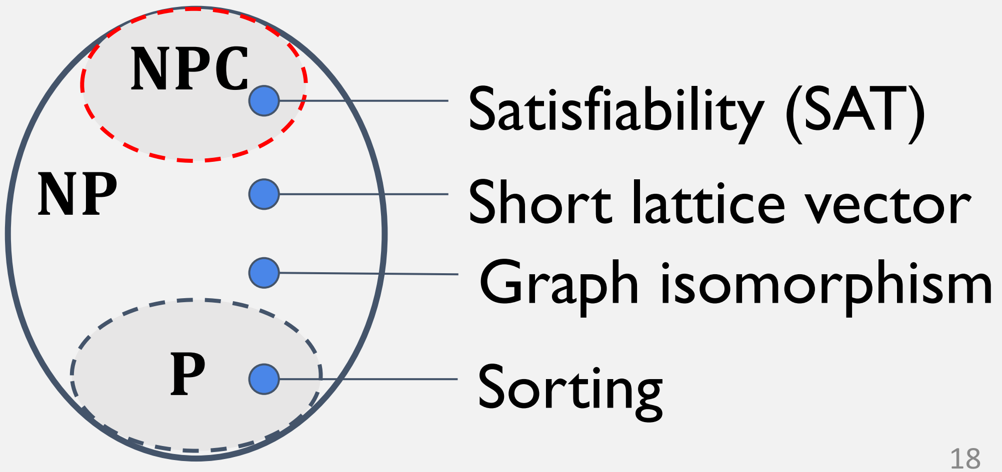
Extended Church-Turing Thesis. A function can be computed efficiently in any *reasonable* model of computation iff. it is efficiently computable by a **Turing machine**.

Disprove ECT???

Quantum supremacy using a programmable superconducting processor

Central ideas in complexity

- Poly-time as "feasible"
  - Most natural problems either are easy (e.g., n^3) or no poly-time alg. known
- Reduction : relating hardness ($A \leq B \Rightarrow A$ no harder than B)
- Classify problems by "hardness"
  - $\mathbf{P}$ = {problems that are easy to answer}
  - $\mathbf{NP}$ = {problems that are easy to verify given hint} [lots of examples, stay tuned!]
  - **Complete** problems: "hardest" in a class



What'd be considered "feasible"?

Q. Which problems will we be able to solve in practice?

A. Those with poly-time algorithms. [von Neumann1953,Godel1956, Cobham1964,Edmonds1965,Rabin1966]

YES	Probably No
Shortest path	Longest path
Matching	3D-matching
Min cut	Max cut
2-SAT	3-SAT
Planar 4-color	Planar 3-color
Bipartite vertex cover	Vertex cover
Primality	Factoring

Classify problems

Desiderata. Classify problems as those that can be solved in polynomial-time and those that cannot.

Roughly: C program on machine with infinite memory

- Provably require exponential time.
  - Given a Turing machine, does it HALT in at most k steps?
  - Given a board position in an $n \times n$ generalization of chess, can black win?

☹Frustrating news: Huge number of fundamental problems have defied classification for decades.

- We will show: these problems are "computationally equivalent" and appear to be different manifestations of one hard problem.

Tool: polynomial-time reduction

Desiderata'. Suppose we can solve Y in poly-time. What else could we solve in polynomial time?

- Reduction. Problem X polynomial reduces to Problem Y if arbitrary instance of X can be solved using:
  - Polynomial number of standard computation steps
  - & polynomial number of calls to oracle that solves A

Notation. $X \leq_{P,Cook} Y$ (or $X \leq_P Y$)

! Mind your direction, don't confuse $X \leq_{\mathrm{P}} Y$ with $Y \leq_{\mathrm{P}} \mathrm{X}$

N.B. We pay for time to write down instances to oracle

$\Rightarrow$ instances of Y must be of polynomial size.

What polynomial-time reductions buy us

- Design algorithms. If $X \leq_P Y$ and Y can be solved in poly-time, then X can also be solved in polynomial time.

- Establish intractability. If $X \leq_P Y$ and X cannot be solved in poly-time, then Y cannot be solved in polynomial time.

- Establish equivalence. If $X \leq_P Y$ and $X \leq_P Y$, then $X \equiv_P Y$.

Bottomline. Reductions classify problems acc. to relative difficulty

Quiz

- Which of the following poly-time reductions are known?
  A. $\text{FIND}-\text{MAX}-\text{FLOW} \leq_P \text{FIND}-\text{MIN}-\text{CUT}$
  B. $\text{FIND}-\text{MIN}-\text{CUT} \leq_P \text{FIND}-\text{MAX}-\text{FLOW}$
  C. Both A and B
  D. Neither A nor B

VALUES VS. ACTUAL FLOW/CUT