Intractability

Introduction

> A problem of intractable if it can't be solved in polynomial time.

- Church-Turing thesis
  - Turing machines can compute any function that can be computed by a physically harnessable process of the natural world
  - No need to seek more powerful machines or languages
  - Enables rigorous study of comutation
- Which problems can we solve in practice?
  - Those with poly-time algorithms

Search problems

Four fundamental problems

LSOLVE. Given a system of linear equations, find a solution.

$$\begin{aligned} 0x_0 + 1x_1 + 1x_2 &= 4 \\ 2x_0 + 4x_1 - 2x_2 &= 2 \\ 0x_0 + 3x_1 + 15x_2 &= 36 \end{aligned} \qquad \begin{aligned} x_0 &= -1 \\ x_1 &= 2 \\ x_2 &= 2 \end{aligned}$$

variables are real numbers

LP. Given a system of linear inequalities, find a solution.

$$\begin{aligned} 48x_0 + 16x_1 + 119x_2 &\le 88 \\ 5x_0 + 4x_1 + 35x_2 &\ge 13 \\ 15x_0 + 4x_1 + 20x_2 &\ge 23 \\ x_0,\ x_1,\ x_2 &\ge 0 \end{aligned} \qquad \begin{aligned} x_0 &= 1 \\ x_1 &= 1 \\ x_2 &= \tfrac{1}{5} \end{aligned}$$

variables are real numbers

ILP. Given a system of linear inequalities, find a 0-1 solution.

$$\begin{aligned} x_1 + x_2 &\ge 1 \\ x_0 + x_2 &\ge 1 \\ x_0 + x_1 + x_2 &\le 2 \end{aligned} \qquad \begin{aligned} x_0 &= 0 \\ x_1 &= 1 \\ x_2 &= 1 \end{aligned}$$

variables are 0 or 1

SAT. Given a system of boolean equations, find a binary solution.

$$\begin{aligned} (x'_1 \text{ or } x'_2) \text{ and } (x_0 \text{ or } x_2) &= true \\ (x_0 \text{ or } x_1) \text{ and } (x_1 \text{ or } x'_2) &= false \\ (x_0 \text{ or } x_2) \text{ and } (x'_0) &= true \end{aligned} \qquad \begin{aligned} x_0 &= false \\ x_1 &= false \\ x_2 &= true \end{aligned}$$

variables are "true" or "false"

LSOLVE. Given a system of linear equations, find a solution.
LP. Given a system of linear inequalities, find a solution.
ILP. Given a system of linear inequalities, find a 0-1 solution.
SAT. Given a system of boolean equations, find a binary solution.

Q. Which of these problems have poly-time algorithms?

- LSOLVE. Yes. Gaussian elimination solves N-by-N system in N^3 time.
- LP. Yes. Ellipsoid algorithm is poly-time. ← but was open problem for decades
- ILP, SAT. No poly-time algorithm known or believed to exist!

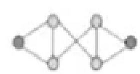
but we still don't know for sure

P vs. NP

- NP is the class of all search problems

problem	description	poly-time algorithm	instance I	solution S
LSOLVE (A, b)	Find a vector x that satisfies $Ax = b$	Gaussian elimination	$0x_0 + 1x_1 + 1x_2 = 4$ $2x_0 + 4x_1 - 2x_2 = 2$ $0x_0 + 3x_1 + 15x_2 = 36$	$x_0 = -1$ $x_1 = 2$ $x_2 = 2$
LP (A, b)	Find a vector x that satisfies $Ax \le b$	ellipsoid	$48x_0 + 16x_1 + 119x_2 \le 88$ $5x_0 + 4x_1 + 35x_2 \ge 13$ $15x_0 + 4x_1 + 20x_2 \ge 23$ $x_0, x_1, x_2 \ge 0$	$x_0 = 1$ $x_1 = 1$ $x_2 = 1/5$
ILP (A, b)	Find a binary vector x that satisfies $Ax \le b$	???	$x_1 + x_2 \ge 1$ $x_0 + x_2 \ge 1$ $x_0 + x_1 + x_2 \le 2$	$x_0 = 0$ $x_1 = 1$ $x_2 = 1$
SAT (Φ, b)	Find a boolean vector x that satisfies $\Phi(x) = b$	???	$(x'_1$ or $x'_2)$ and $(x_0$ or $x_2)$ = true $(x_0$ or $x_1)$ and $(x_1$ or $x'_2)$ = false $(x_0$ or $x_2)$ and (x'_0) = true	x_0 = false x_1 = false x_2 = true
FACTOR (x)	Find a nontrivial factor of the integer x	???	147573952589676412927	193707721

- P is the class of search problems solvable in poly-time

problem	description	poly-time algorithm	instance I	solution S
LSOLVE (A, b)	Find a vector x that satisfies $Ax = b$	Gaussian elimination (Edmonds 1967)	$0x_0 + 1x_1 + 1x_2 = 4$ $2x_0 + 4x_1 - 2x_2 = 2$ $0x_0 + 3x_1 + 15x_2 = 36$	$x_0 = -1$ $x_1 = 2$ $x_2 = 2$
LP (A, b)	Find a vector x that satisfies $Ax \le b$	ellipsoid (Khachiyan 1979)	$48x_0 + 16x_1 + 119x_2 \le 88$ $5x_0 + 4x_1 + 35x_2 \ge 13$ $15x_0 + 4x_1 + 20x_2 \ge 23$ $x_0, x_1, x_2 \ge 0$	$x_0 = 1$ $x_1 = 1$ $x_2 = 1/5$
SORT (a)	Find a permutation that puts array a in order	mergesort (von Neumann 1945)	2.3 8.5 1.2 9.1 2.2 0.3	5 2 4 0 1 3
STCONN (G, s, t)	Find a path in a graph G from s to t	depth-first search (Theseus)		

Nondeterminism

- nondeterminism machine can guess the desired solution

Classifying Problems

- Problem X ploy-time reduces to problem Y if X can be solved with:
  - Polynomial number of standard computational steps
  - Polynomial number of calls to Y
- Consequence
  - If we can poly-time reduce SAT to problem Y, then we conclude that Y is (probably) intractable

SAT reduces to ILP

SAT. Given a system of boolean equations, find a solution.

$$\begin{aligned} x'_1 \text{ or } x_2 \text{ or } x_3 &= \text{true} \\ x_1 \text{ or } x'_2 \text{ or } x_3 &= \text{true} \\ x'_1 \text{ or } x'_2 \text{ or } x'_3 &= \text{true} \\ x'_1 \text{ or } x'_2 \text{ or } x_4 &= \text{true} \end{aligned}$$

← can reduce any SAT problem to this form

ILP. Given a system of linear inequalities, find a binary solution.

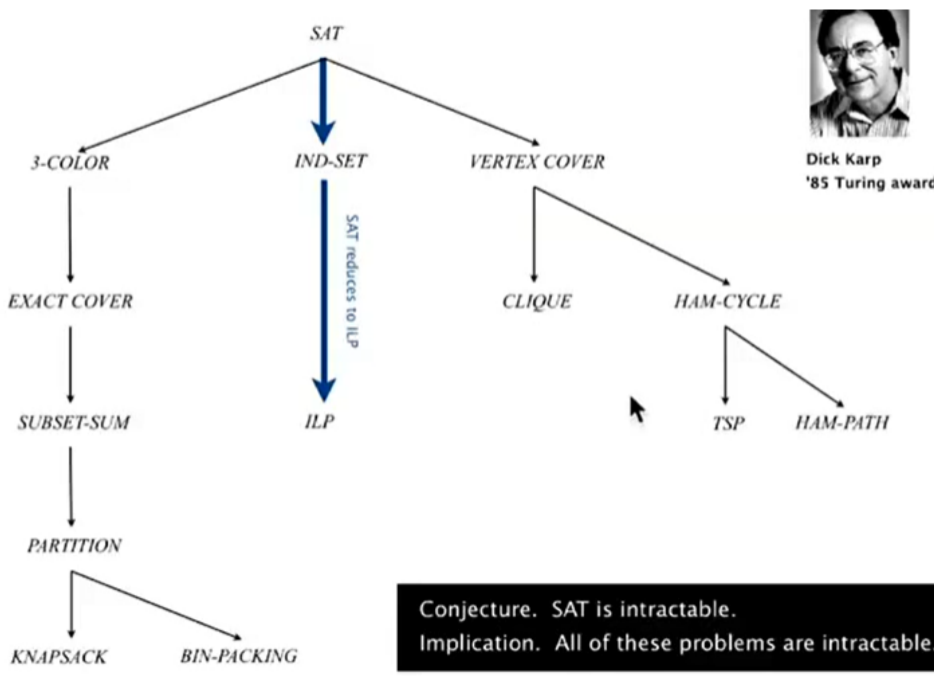
$C_i = 1$ iff equation i is satisfied

$$\begin{array}{ll} C_1 \ge 1 - x_1 & C_3 \ge 1 - x_1 \\ C_1 \ge x_2 & C_3 \ge 1 - x_2 \\ C_1 \ge x_3 & C_3 \ge 1 - x_3 \\ C_1 \le (1 - x_1) + x_2 + x_3 & C_3 \le 3 - x_1 - x_2 - x_3 \\ & \\ C_2 \ge x_1 & C_4 \ge 1 - x_1 \\ C_2 \ge 1 - x_2 & C_4 \ge 1 - x_2 \\ C_2 \ge x_3 & C_4 \ge x_4 \\ C_2 \le 1 + x_1 - x_2 + x_3 & C_4 \le 2 - x_1 - x_2 + x_4 \end{array}$$

$\Phi = 1$ iff all equations are satisfied

$$\begin{aligned} \Phi &\le C_1 \\ \Phi &\le C_2 \\ \Phi &\le C_3 \\ \Phi &\le C_4 \\ \Phi &\ge C_1 + C_2 + C_3 + C_4 - 3 \end{aligned}$$

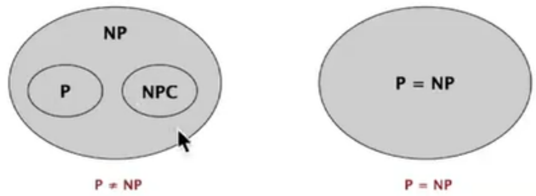
More reductions to SAT



Conjecture. SAT is intractable.
Implication. All of these problems are intractable.

NP-completeness

- An NP problem is NP-complete if all problems in NP poly-time reduce to it
- SAT is NP-complete
- Implication
  - Poly-time algorithm for SAT iff P=NP
  - No poly-time algorithm for some NP problem $\Rightarrow$ none for SAT



Summary

- P---class of search problems solvable in poly-time
- NP---class of all search problems, some of which seem wickedly hard
- NP-complete---hardest problems in NP
- Intractable---problem with no poly-time algorithm
- Use a theory guide
  - A poly-time algorithm for an NP-complete problem would be a stunning breakthrough (a proof that P=NP)
  - You will confront NP-complete problems in your career
  - Safe to assume that P≠NP and that such problems are intractable
  - Identify these situations and proceed accordingly

Coping with Intractability

- FACTOR
  - In NP, but not known (or believed) to be in P or NP-complete
- Relax one of desired features
  - Solve arbitrary instances of the problem

    Special cases may be tractable.
    - Ex: Linear time algorithm for 2-SAT. ← at most two literals per equation
    - Ex: Linear time algorithm for Horn-SAT. ← at most one un-negated literal per equation

  - Solve the problem to optimality

    Develop a heuristic, and hope it produces a good solution.
    - No guarantees on quality of solution.
    - Ex. TSP assignment heuristics.
    - Ex. Metropolis algorithm, simulating annealing, genetic algorithms.

    Approximation algorithm. Find solution of provably good quality.
    - Ex. MAX-3SAT: provably satisfy 87.5% as many clauses as possible.

    but if you can guarantee to satisfy 87.51% as many clauses as possible in poly-time, then P = NP !

  - Solve the problem in poly-time

Complexity theory deals with worst case behavior.

- Instance(s) you want to solve may be "easy."
- Chaff solves real-world SAT instances with ~ 10K variable.