Computational complexity

SDPs can be solved efficiently [e.g. [BV] Algorithms chapters]

Beyond SDPs? Which optimization problems can be solved efficiently?

Two popular answers: simplicity is associated with

ⓐ convexity

ⓑ local optimization.

Generally makes sense, but not quite :)

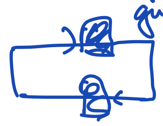
"Easy" non-convex cases:

- $\min \ x^T Q x + b^T x + c$ when $Q_i \not\succeq 0 \longleftarrow$ S-lemma

  s.t. $x^T Q_2 x + b_2^T x + c_2$

- $\min\limits_{x^T x = 1} x^T B x \quad B \in Sym(n) \longleftarrow \lambda_{min}(B)$

- $\min\limits_{\substack{X \\ \operatorname{rk} X = k}} \|X - B\|_F \longleftarrow$ SVD gives the best low-rank approximation

-  given A, B, find K. $\quad x_{k+1} = \underline{A} x_k$

$$\rho(A+BK) < 1$$

$$\Updownarrow$$

$$P \succ 0$$

$$(A+B\underline{K})^T \underline{P} (A+B\underline{K}) \prec P$$

$\hookrightarrow$ to SDP problem after a smart variable change (LMIs)

"Hard" convex problems also exist:

For any optimization problem

any f, any Ω

$$\begin{bmatrix} \min\limits_{x\in\Omega} f(x) \end{bmatrix} \longleftrightarrow \begin{bmatrix} \min\limits_{\alpha, x} \alpha \\ x\in\Omega,\ \alpha \geq f(x) \end{bmatrix} \text{ epigraph form}$$

$$\longleftrightarrow \begin{bmatrix} \min\limits_{\alpha, x} \alpha \\ (x,\alpha) \in \operatorname{conv}\{x\in\Omega,\ \alpha \geq f(x)\} \end{bmatrix}$$

Any concrete examples? NP-hard problems:

- Checking nonnegativity of a polynomial of degree 4.
- Checking whether 0 is a local min of a polynomial of degree 4.
- Minimizing a quadratic polynomial subject to polytope (simplex) constraints
- Checking if a symmetric matrix M $x^TMx \geq 0$ for any $x \geq 0$

(compare with checking PSD property that is "easy", enough to check on the spectrum)

We will formalize and prove the hardness of these examples.

First, what is a hard problem?

we consider decision problems (yes/no question)

Example: "find maximum size of an independent set in a graph" (search problem)

"is there an independent set of the size $\geq k$" (decision problem)

Class $\mathcal{P}$ contains decision problems so that $\underbrace{\exists \text{ polynomial algorithm}}$ that gives correct yes/no answer at any input

its running time $\leq C \cdot p(n)$,

where C = constant, same for any input

n = number of bits in the input

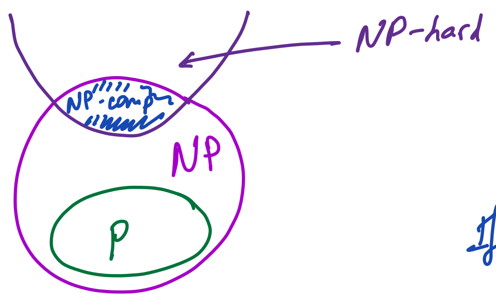
p = polynomial function

Class NP contains decision problems so that with any instance (= particular input) with the answer "yes" there exists a certificate that can be checked in polynomial time.

$P \subset NP$ (if algorithm exists, it can serve as a certificate)
Typical graph problems (like independent set) usually have a certificate "optimal subset itself"

Def NP-hard problems are so that any NP-hard problem can be reduced to them, i.e.
if we find a polynomial time algorithm for them $\Rightarrow$ we find it for any NP-problem.

NP-complete problems = NP-hard and in NP



If P=NP, all these classes (and others) will coincide

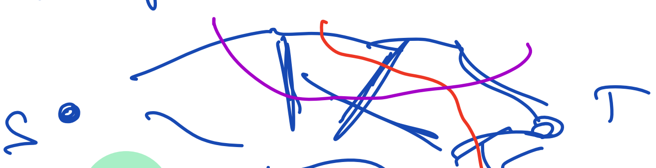
① How to check if a problem is easy (in P)?
② How to check if a problem is hard (NP-hard)?

---

① · Give an algorithm :) Might be hard (LP problems)

· Reductions : $A \longrightarrow B$
A is hard $\Rightarrow$ B must be hard. (NP hard)
A is easy $\Leftarrow$ B is easy (P)

Recall from the 1st class:



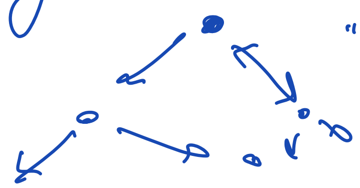
MINCUT $\rightarrow$ MIN-CUT between S-T $\longrightarrow$ MAX FLOW $\longrightarrow$ LP
(red cut is helpful, not purples)

reduction happens by solving $\sim n^2$ problems
(varying S and T vertices)

② Checking that a problem is NP-hard almost always is done via reductions

. Cook-Levin → constructed one instance of an NP-hard problem

. Karp → created first ~ 20 examples of problems reducing to it



"a tree of reductions"

Some examples:

SAT A Boolean formula is in conjunctive normal form

$x_1, \dots x_n$ $x_i = 0, 1$ $x_1 \vee x_2$ or

$x_1 \wedge x_2$ and

$\bar{x}_1$ negation

$x_1 \vee x_2 =$

x_1	x_2	$x_1 \vee x_2$	$x_1 \wedge x_2$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

example

Input: $\varphi = \underbrace{(\overset{1}{x_1} \vee \overset{1}{x_2} \vee \overset{0}{x_3})}_{\text{clause 1}} \wedge \underbrace{(\overset{1}{x_1} \vee \overset{0}{\bar{x}_2})}_{\text{clause 2}} \wedge \underbrace{(\overset{0}{x_2} \vee \overset{1}{\bar{x}_3})}_{\text{clause 3}} \wedge \underbrace{(\overset{0}{\bar{x}_1} \vee \overset{0}{\bar{x}_2} \vee \overset{1}{\bar{x}_3})}_{\text{clause 4}}$

Question: are there assignments of 1 and 0 to x_i's that satisfy φ ($\varphi = 1$)?

$x_1 \wedge \bar{x}_1$ - not satisfiable

• 3SAT Same as above, but each clause must contain 3 variables

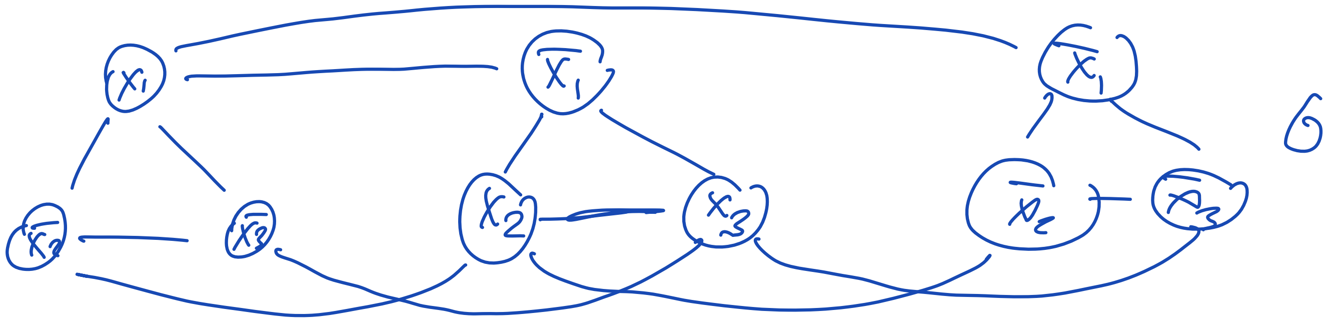
$\underbrace{\text{SAT} \to \text{3SAT}}_{S}$, then 3SAT is NP-hard!

reduction happens via restructuring all the clauses to have 3 variables in each

Exercise: (repeat variables ...)

- 1 in 3 SAT → STABLE SET
  - size of the independence set
- 1 in 3 SAT only allows one 1 in each clause

Example: $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$



φ is satisfiable in 1 in 3 SAT sense $\iff \alpha(G) \geq k$

STABLE SET → integer optim

$$\begin{cases} \sum x_i \geq k \\ x_i x_j = 0 \quad i \overset{E}{\leftrightarrow} j \\ x_i \in \{0,1\} \end{cases}$$

$x_i \in \{0,1\} \to x_i(1-x_i) = 0$

→ Quadratic feasibility is NP-hard

$$\begin{cases} \sum x_i - k \geq 0 \\ x_i x_j = 0 \\ x_i(1-x_i) = 0 \end{cases}$$

POLYPOS 4 : Input: a degree 4 polynomial with rational coeff's

$p(x) \geq 0 \quad x \in \mathbb{R}^n$

Note: degrees 1–3 result in polynomial algorithms.

In degree 4:

$\underline{p(x) := \sum q_i^2(x)} \iff \underline{q_i(x) = 0 \text{ is feasible}}$

each q_i is quadratic

Quadratic feasibility