Lecture 26

Hardness of Approximation

Foundations of Probabilistic Proofs
Alessandro Chiesa

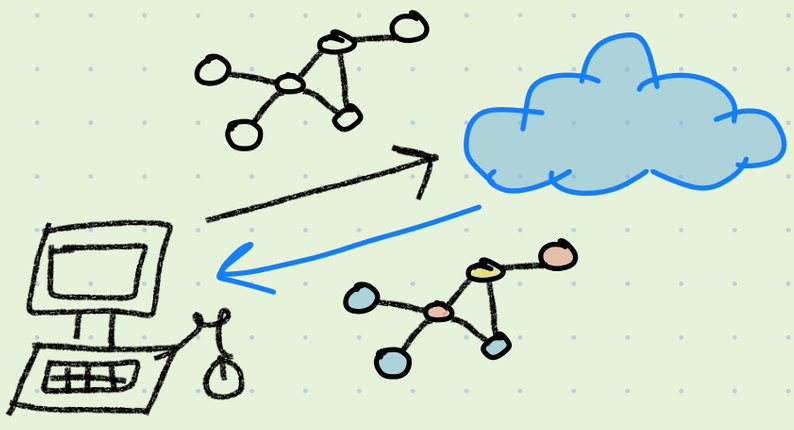
Applications of PCPs

Two main directions:

SEEN BEFORE

Computational Integrity

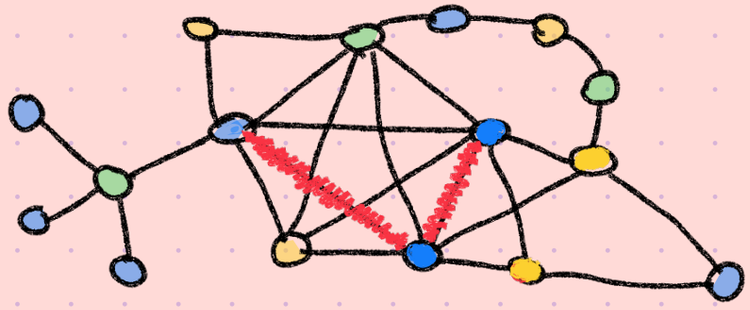
How to verify computations faster than they can be run?



TODAY

Hardness of Approximation

Which problems remain hard even if we only seek an approximate solution?



Coping with NP-Hardness

Numerous fundamental optimization problems are NP-hard.

E.g. 3SAT, GRAPH COLORING, TRAVELING SALESMAN, KNAPSACK, CLIQUE, VERTEX COVER, ...

→ If P≠NP then none of these problems can be solved in polynomial time.

Q: How to cope with NP-hardness?

① correct but slow algorithms

Solve the problem exactly via an algorithm whose (super-polynomial) running time is not bad on small instances. Ex: $t(n) = 1.1^n$ ($1.1^{200} < 200$ million $\ll 2^{200}$)

② fast but incorrect algorithms (aka APPROXIMATION ALGORITHMS)

Solve the problem approximately via a polynomial-time algorithm.

The quality of the approximation must be guaranteed for every input.

Another direction to cope with NP-hardness is to forego worst-case input guarantees:

ⓘ efficient algorithms that solve "natural" inputs (a major challenge is to model "natural")

ⓘⓘ efficient algorithms that work well in practice (aka heuristics)

Approximation Algorithms

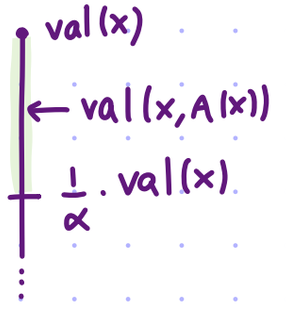
A **maximization problem** is a pair $\mathcal{O} = (sol, val)$ where:

- $sol(x)$ is the (possibly empty) set of valid solutions for the instance $x \in \{0,1\}^*$
- $val(x,w)$ is the value of the solution $w \in sol(x)$ for x

We define $val(x) := \max\{ val(x,w) : w \in sol(x) \}$.

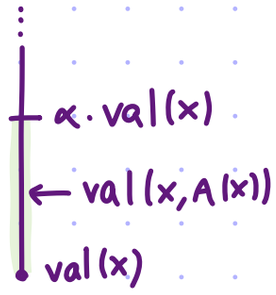
An algorithm A has **approximation ratio $\alpha \geq 1$** for $\mathcal{O} = (sol, val)$ if

$$\forall x \quad val(x, A(x)) \geq \frac{1}{\alpha} \cdot val(x) .$$



The **minimization** case is similar except that:

- $val(x) := \min\{ val(x,w) : w \in sol(x) \}$.
- $\alpha \geq 1$ must satisfy $\forall x \quad val(x, A(x)) \leq \alpha \cdot val(x)$.



Example: Max3SAT
- Instances x are 3CNF formulas.
- If x has n variables then $sol(x) = \{0,1\}^n$.
- $val(x,w)$ = fraction of clauses in x satisfied by $w \in \{0,1\}^n$

Basic Goal: design approximation algorithms for NP-hard problems with small α

Approximation Landscape

Researchers have designed approximation algorithms since the 1970s.

Examples:

- KNAPSACK — $\alpha = 1+\varepsilon$ in time $\text{poly}(n, \frac{1}{\varepsilon})$ (FPTAS based on dynamic programming)
- VERTEXCOVER — $\alpha = 2$ (vertices in a maximal matching)
- SETCOVER — $\alpha = O(\log n)$ (pick set covering most points & repeat)
- INDEPENDENTSET — $\alpha = \text{MaxDegree}+1 = O(n)$ (remove min-degree vertex and its neighbors & repeat)

(duals!)

- TRAVELINGSALESMAN — none

Results suggest that problems behave very differently wrt approximation factors.
(Even if the problems are reducible to one another via polynomial-time reductions.)

Q: Why different approximation factors?

NP reductions preserve satisfiability but not necessarily approximability.
They independently transform subgadgets with differing blowups in size.

Hardness of Approximation

Q: Why these specific approximation factors?

We can aim to explain them by showing that we cannot do better:

GOAL: prove hardness of approximation results

We wish to prove statements of the form

"If problem $\mathcal{O}$ is α-approximable in polynomial time then $P = NP$".

This generalizes the classical "If problem $\mathcal{O}$ is solvable in polynomial time then $P = NP$".

(solvable = 1-approximable)

Challenge: NP reductions don't have "gaps"

computation of a non-deterministic TM (sensitive to small changes) — Cook-Levin Theorem → 3SAT — reduction → problem of interest

(local reductions)

→ Strong hardness of approximation requires rejecting computations to be far from accepting ones.

→ PCPs play a role in hardness of approximation

Constraint Satisfaction Problems

A constraint satisfaction problem (CSPs) generalizes the notion of 3SAT.

def: A (Σ,ℓ,q)-constraint is a pair $C=(S,f)$ where $S\in\binom{[\ell]}{q}$ and $f:\Sigma^S\to\{0,1\}$.
An assignment $a\in\Sigma^\ell$ satisfies $C=(S,f)$ if $f(a(S))=1$.

def: A (Σ,ℓ,q,m)-CSP is a list $\phi=(C_1,\dots,C_m)$ where each $C_j=(S_j,f_j)$ is a (Σ,ℓ,q)-constraint.
The value of ϕ on assignment $a\in\Sigma^\ell$ is $val(\phi,a):=\frac{1}{m}\cdot\sum_{j=1}^m f_j(a(S_j))$.
The value of ϕ is $val(\phi):=\max_{a\in\Sigma^\ell} val(\phi,a)$, and ϕ is satisfiable if $val(\phi)=1$.

We associate two problems to CSPs:

- MaxCSP$[\Sigma,\ell,q,m]$ is the search problem that asks to find $a\in\Sigma^\ell$ that maximizes $val(\phi,a)$
- GapCSP$[\varepsilon_c,\varepsilon_s,\Sigma,\ell,q,m]$ is the decision problem that asks to distinguish if $val(\phi)\geq 1-\varepsilon_c$ or $val(\phi)\leq\varepsilon_s$

claim: GapCSP$[\varepsilon_c,\varepsilon_s,\Sigma,\ell,q,m]$ is NP-hard → approximating MaxCSP$[\Sigma,\ell,q,m]$ with $\alpha<\frac{1-\varepsilon_c}{\varepsilon_s}$ is NP-hard

proof: If $val(\phi)\geq 1-\varepsilon_c$ then the approximation is at least $\frac{1}{\alpha}\cdot val(\phi)\geq\frac{1}{\alpha}\cdot(1-\varepsilon_c)>\frac{\varepsilon_s}{1-\varepsilon_c}\cdot(1-\varepsilon_c)=\varepsilon_s$.
If $val(\phi)\leq\varepsilon_s$ then the approximation is at most $val(\phi)\leq\varepsilon_s$. ■

To show hardness of approximation it suffices to show hardness of gap problems.

CSP vs PCP

There is an equivalence between CSPs and (non-adaptive) PCPs.

claim: L reduces to $\text{GapCSP}[\varepsilon_c, \varepsilon_s, \Sigma, \ell, q, m] \to L \in \text{PCP}[\varepsilon_c, \varepsilon_s, \Sigma, \ell, q, r=\log m]$

proof: $V_{PCP}^{a \in \Sigma^\ell}(x)$: 1. Reduce x to the CSP $\phi = (C_1, \ldots, C_m)$.

2. Sample $j \in [m]$.

3. Check that $f_j(a(S_j)) = 1$.

The PCP verifier makes q queries and uses $r = \log m$ random bits.

Completeness and soundness follow from the fact that $\forall a \in \Sigma^\ell \quad \Pr[V_{PCP}^a(x) = 1] = \text{val}(\phi, a)$. ■

claim: $L \in \text{PCP}[\varepsilon_c, \varepsilon_s, \Sigma, \ell, q, r] \to L$ reduces to $\text{GapCSP}[\varepsilon_c, \varepsilon_s, \Sigma, \ell, q, m = 2^r]$ in time $\text{poly}(2^r, n)$

proof: Let $V = (Q, D)$ be the PCP verifier for L.

Map the instance x to the CSP instance $\phi = (C_\rho)_{\rho \in \{0,1\}^r}$ where

$$C_\rho = (S_\rho, f_\rho) \text{ with } S_\rho = Q(x, \rho) \text{ and } f_\rho(a(S_\rho)) = D(x, \rho, a(S_\rho)).$$ ■

Dictionary:

PCP verifier	↔ CSP instance	proof length	↔ number of variables
PCP string	↔ assignment	query complexity	↔ arity of constraints
completeness/soundness	↔ YES/NO thresholds	randomness complexity	↔ log of number of constraints

Inapproximability of Max3SAT [1/3]

We learned that $NP \subseteq PCP[\varepsilon_c, \varepsilon_s, \Sigma, \ell, q, r=O(\log n)] \to GapCSP[\varepsilon_c, \varepsilon_s, \Sigma, \ell, q, m=poly(n)]$ is NP-hard.

So the PCP Theorem tells us that $\exists\, \varepsilon_s, q$ s.t. $GapCSP[\varepsilon_c=0, \varepsilon_s, \Sigma=\{0,1\}, \ell=poly(n), q, m=poly(n)]$ is NP-hard.

What can we say about the inapproximability of Max3SAT ($q=3$ and constraints are 3CNF clauses)?

<u>theorem</u>: $\exists\, \varepsilon_s \in (0,1)$ s.t. deciding if a 3SAT ϕ has $val(\phi)=1$ or $val(\phi) \leq \varepsilon_s$ is NP-hard

Follows from the above result and the next lemma (map each constraint to a 3CNF):

<u>lemma</u>: $GapCSP[\varepsilon_c, \varepsilon_s, \Sigma=\{0,1\}, \ell, q, m]$ reduces to $Gap3SAT[\varepsilon_c, \varepsilon_s' = 1-\frac{1-\varepsilon_s}{q\cdot 2^q}, \Sigma=\{0,1\}, \ell' = \ell + m2^q q, q'=3, m' = m2^q q]$

<u>proof</u>: Let $\phi = (C_1, \ldots, C_m)$ be a $(\{0,1\}, \ell, q, m)$-CSP.

Express each C_j as $\bigwedge_{k=1}^{2^q} \phi_{j,k}$ (use the evaluation table)

where each $\phi_{j,k}$ is the OR of $\leq q$ literals over the assignment's variables $x_1, \ldots, x_\ell$.

Then express each $\phi_{j,k}$ as a 3CNF using $\leq q$ clauses and $\leq q$ auxiliary variables.

Overall we have $\ell' \leq \ell + m \cdot 2^q \cdot q$ variables and $m' \leq m \cdot 2^q \cdot q$ constraints.

If $val(\phi, a) \geq 1-\varepsilon_c$ then can extend $a \in \{0,1\}^\ell$ to $a' \in \{0,1\}^{\ell'}$ s.t. $val(\phi', a') \geq 1-\varepsilon_c$.

(If ϕ_j is SAT then $\{\phi_{j,k,t}\}_{k,t}$ are all SAT. If ϕ_j is not SAT then, in the worst case, $\{\phi_{j,k,t}\}_{k,t}$ are all not SAT.)

If, $\forall a \in \{0,1\}^\ell$, $val(\phi, a) \leq \varepsilon_s$ then any extension $a' \in \{0,1\}^{\ell'}$ satisfies $val(\phi', a') \leq 1 - \frac{1-\varepsilon_s}{q\cdot 2^q}$.

(If ϕ_j is not SAT then, in the worst case, only one of $\{\phi_{j,k,t}\}_{k,t}$ is not SAT.)

■

Inapproximability of Max3SAT [2/3]

lemma: There is an expected polynomial-time algorithm with approximation ratio $\alpha = 8/7$ for MaxE3SAT.

formula is an E3CNF (each clause has exactly 3 literals)

proof:

$A(\phi)$: 1. Sample a random $a \in \{0,1\}^n$.

2. If $\mathrm{val}(\phi, a) < 7/8$ go to 1. Else output a.

The algorithm A outputs a s.t. $\mathrm{val}(\phi, a) \geq \frac{7}{8} = \frac{1}{8/7} \cdot 1 \geq \frac{1}{8/7} \cdot \mathrm{val}(\phi)$.

We are left to analyze its expected running time.

For $j \in [m]$, $Z_j :=$ "indicator that a random $a \in \{0,1\}^n$ satisfies j-th clause of ϕ".

Note that $\mathbb{E}[Z_j] = \Pr[Z_j = 1] = 1 - \Pr[Z_j = 0] = 1 - \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = 7/8$. (Each clause has exactly 3 literals.)

Hence $\mathbb{E}[\sum_{j=1}^{m} Z_j] = \sum_{j=1}^{m} \mathbb{E}[Z_j] = 7/8 \cdot m$.

We deduce that $\exists\, a \in \{0,1\}^n$ that satisfies $7/8 \cdot m$ clauses.

In fact, $p := \Pr[\sum_{j=1}^{m} Z_j \geq 7/8 \cdot m] \geq \frac{1}{8m}$ because, letting $p_k := \Pr[\sum_{j=1}^{m} Z_j = k]$,

$$\frac{7}{8} \cdot m = \mathbb{E}\left[\sum_{j=1}^{m} Z_j\right] = \sum_{0 \leq k \leq \lfloor \frac{7}{8}m - 1 \rfloor} k \cdot p_k + \sum_{\frac{7}{8}m \leq k \leq m} k \cdot p_k \leq \left(\frac{7}{8}m - \frac{1}{8}\right) \sum_{0 \leq k < \frac{7}{8}m} p_k + m \cdot \sum_{\frac{7}{8}m \leq k \leq m} p_k \leq \left(\frac{7}{8}m - \frac{1}{8}\right) \cdot 1 + m \cdot p \longrightarrow p \geq \frac{1}{8m}.$$

So the expected number of samples is $8 \cdot m$, and thus A runs in expected time $O(m^2)$. ■

Inapproximability of Max3SAT [3/3]

We showed that $\exists \varepsilon_s \in (0,1)$ s.t. distinguishing if a E3CNF ϕ has $val(\phi)=1$ or $val(\phi) \leq \varepsilon_s$ is NP-hard.

Q: How small can ε_s be?

We expect that $\varepsilon_s \geq 7/8$ because of the (expected-time) $\frac{8}{7}$-approximation algorithm for MaxE3SAT.
(That algorithm can be made deterministic via the method of Conditional Expectations.)

The approximation ratio cannot be improved:

<u>theorem</u>: $\forall \gamma > 0$ it is NP-hard to distinguish if a E3CNF ϕ has $val(\phi)=1$ or $val(\phi) \leq \frac{7}{8}+\gamma$

We do not prove this theorem.

The proof shows that $NP \subseteq PCP\left[\varepsilon_c=0, \varepsilon_s \leq \frac{7}{8}+\gamma, \Sigma=\{0,1\}, \ell=poly(n), q=3, r=O(\log n)\right]$ where the (non-adaptive) PCP verifier decides via an E3CNF clause.

Tools include: PCP Theorem, parallel repetition, long code, boolean Fourier analysis.

The soundness error can be improved if the (non-adaptive) PCP verifier can be arbitrary:

$$\forall \gamma > 0,\ NP \subseteq PCP\left[\varepsilon_c=0, \varepsilon_s \leq \frac{5}{8}+\gamma, \Sigma=\{0,1\}, \ell=poly(n), q=3, r=O(\log n)\right]$$