Complexity Theory

Instructor: Roei Tell

Stronger forms of P ≠ NP

Course outline

Modeling

Computability

Basic complexity

P vs NP

Strong forms of P ≠ NP

Lemons into lemonade

ETH & fine-grained complexity

PCPs & hardness of approx

Fine-grained complexity

ETH & fine-grained hardness

Exponential time hypothesis

- Observation:

  All known algorithms for $\mathbf{NP}$-complete problems run in exponential time, i.e. time $2^{n^{\wedge}\varepsilon}$ for some $\varepsilon = \varepsilon_L > 0$.

  - 3-SAT over n variables: time $2^{\Omega(n)}$
  - 3-COLOR over n vertices: time $2^{\Omega(n)}$
  - k-CLIQUE over n vertices: time $2^{\Omega(n)}$ or $n^{\Omega(k)}$
  - …

Exponential time hypothesis

- Claim:

  If there is an $\mathbf{NP}$-complete problem L that can be solved in time $2^{n^{o(1)}}$ then all $\mathbf{NP}$ problems can be solved in time $2^{n^{o(1)}}$.

Exponential time hypothesis

- Conjecture (ETH):

  There is $\varepsilon > 0$ such that 3-SAT on n variables cannot be solved in time less than $2^{\varepsilon \cdot n}$.

Exponential time hypothesis

› Lemma (sparsification lemma):

If ETH is true, then there is $\varepsilon > 0$ such that 3-SAT on n variables with $m = O(n)$ clauses cannot be solved in time less than $2^{\varepsilon \cdot n}$.

Fine-grained complexity

- Fix your favorite model (e.g., RAM, or k-tape machine)
- Some problems in $\mathbb{P}$ aren't known to be solvable in linear time

  k-CLIQUE, k-COLOR, k-SUM, APSP, Edit Distance,

  Longest Common Subsequence, Linear Programming, …

Fine-grained complexity

- Some of these problems resisted decades of attempts to design faster algorithms
- Is there a reason for failure?
- Fine-grained lower bounds

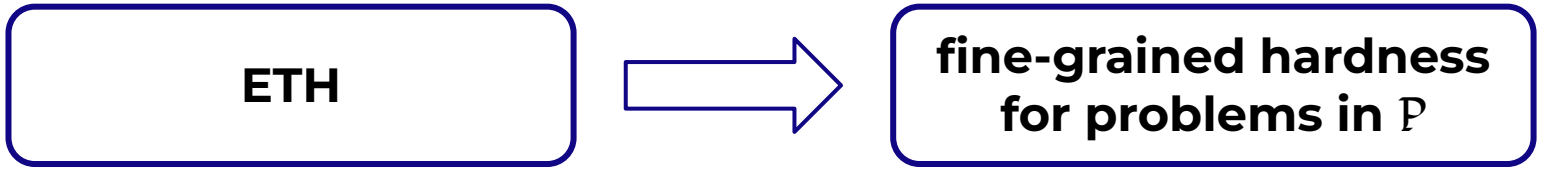
  “Quadratic is optimal”, “Cubic is optimal”, etc.

Key connection

ETH

fine-grained hardness for problems in P

Key connection



› <u>The blowup idea (an example):</u>

Transform a formula over n vars into a graph with $N = 2^{\varepsilon \cdot n}$ vertices

A linear-time O(N) algorithm that solves the graph problem yields an $O(2^{\varepsilon \cdot n})$-time algorithm that solves satisfiability for the formula

Hence, assuming ETH, no linear-time alg for the graph problem!

Fine-grained hardness

› Theorem:

Assuming ETH, there is $\varepsilon > 0$ st for every large enough constant k, k-CLIQUE cannot be solved in time less than $n^{\varepsilon \cdot k}$.

Fine-grained hardness

› Theorem:

Assuming ETH, there is $\varepsilon > 0$ st for every large enough constant k, k-SUM cannot be solved in time less than $n^{\varepsilon \cdot k}$.

Probabilistically checkable proofs

& hardness of approximation

Reminder: NTIME[T]

- Definition:

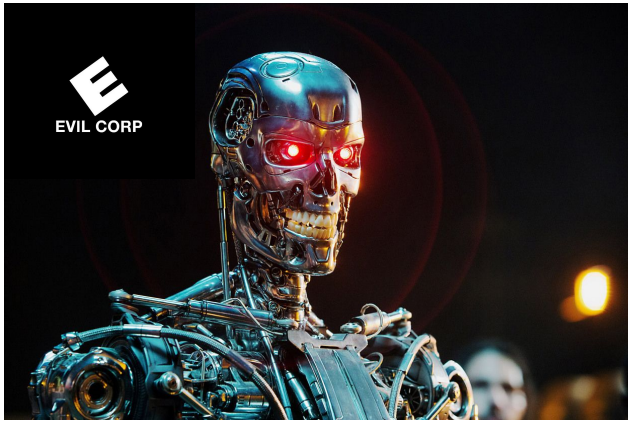
  We say that $L \subseteq \{0,1\}^*$ is **verifiable in time T** if there's a TM V st:

  - $x \in L \quad \Rightarrow \quad \exists\ w \in \{0,1\}^{\leq T(|x|)}, \qquad V(x,w) = 1$
  - $x \notin L \quad \Rightarrow \quad \forall\ w \in \{0,1\}^{\leq T(|x|)}, \qquad V(x,w) = 0$
  - V(x,w) runs in time $O(T(|x|))$

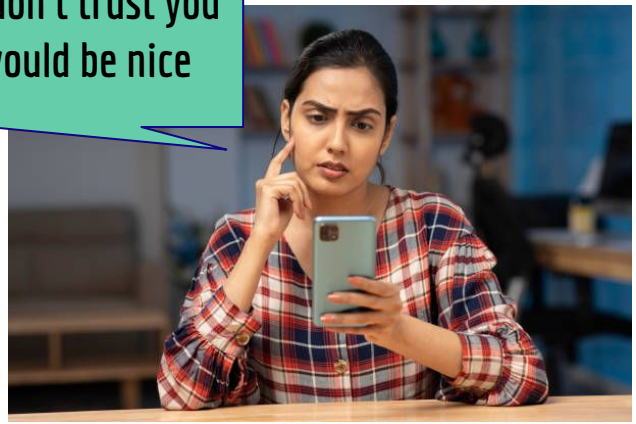
Proof system

$x \in \{0,1\}^n$

Proof system

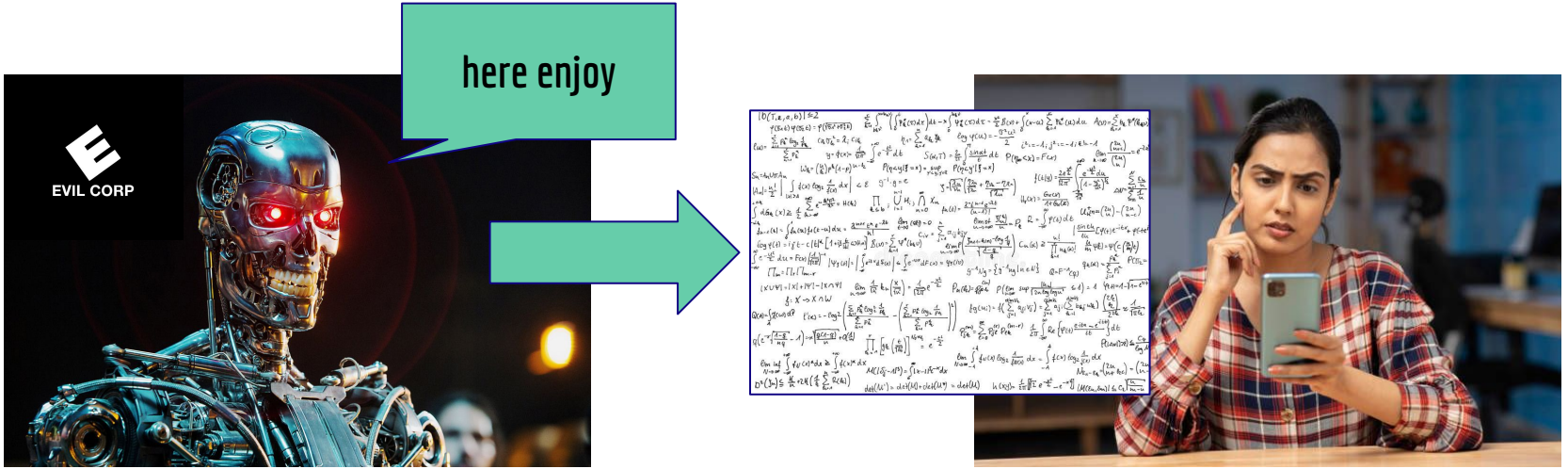


it's not exactly that I don't trust you but maybe a proof would be nice



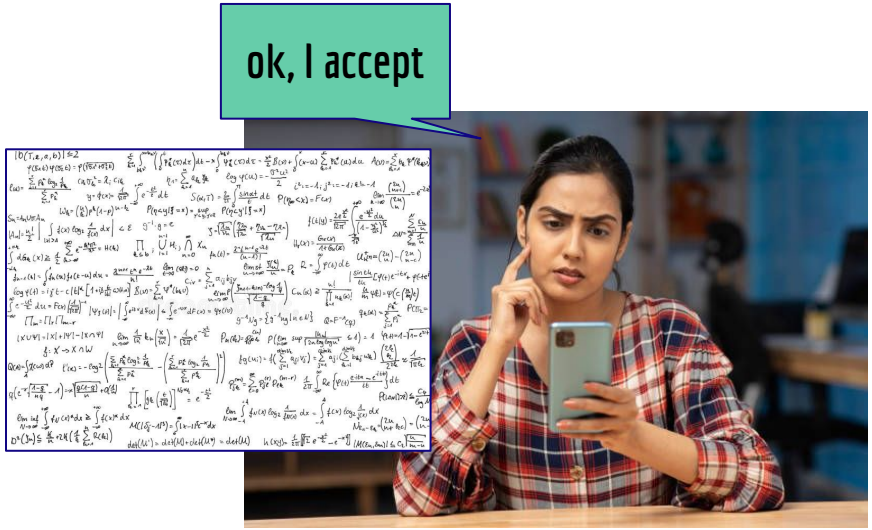
$x \in \{0,1\}^n$

Proof system



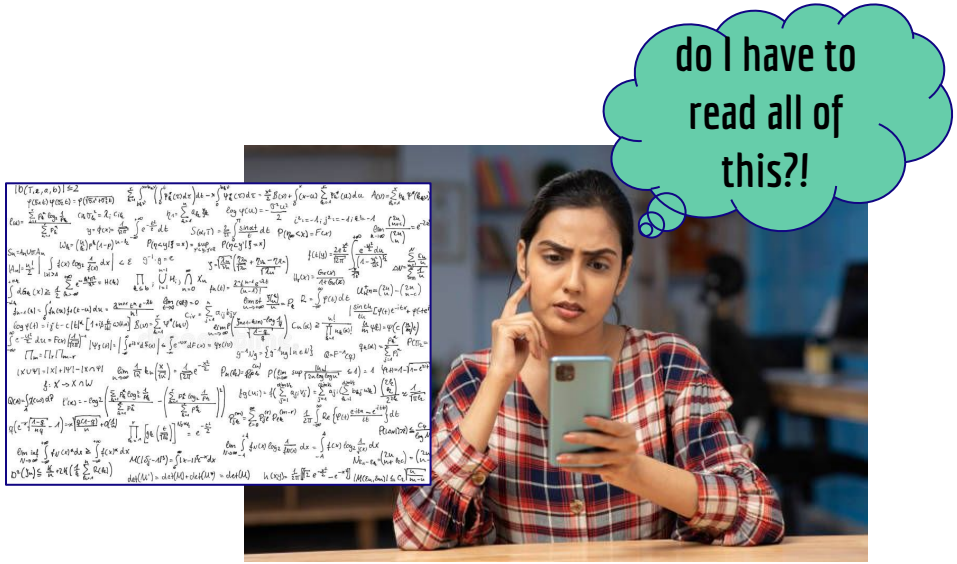
$$x \in \{0,1\}^n$$

Proof system



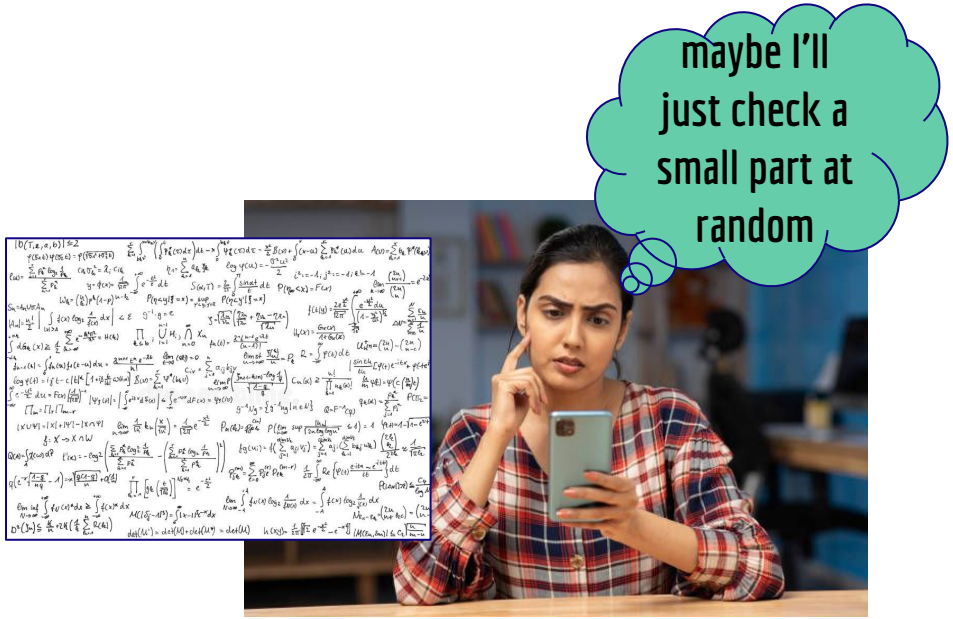
$$x \in \{0,1\}^n$$

Proof system



$$x \in \{0,1\}^n$$

Proof system



$$x \in \{0,1\}^n$$

Naive approaches are unsound

- Examples

  Probabilistically checking the naive proof for CLIQUE

  Probabilistically checking the naive proof for QUAD

Naive approaches are unsound

- Examples

  Probabilistically checking the naive proof for CLIQUE

  Probabilistically checking the naive proof for QUAD

- Lesson:

  To pull this off, proofs will need to be special.

Probabilistically checkable proofs

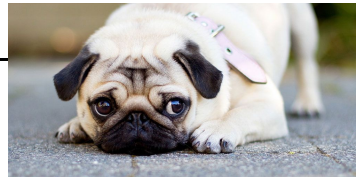
- Oracle machines:

  An oracle Turing machine gets input x and oracle access to a string π. It has a special tape where it can write $i \in N$, go into a special "query" state, and on another special tape get answer:

  - π_i // i^{th} bit of π
  - $|\pi| < i$ // "bad query"

[1] in many sources oracles are defined as functions rather than strings, but the latter is more general

Probabilistically checkable proofs



- Reminder about probability:

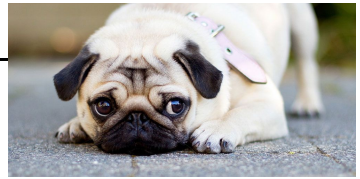
  A probably space is:

  - a sample-space Ω // set of outcomes
  - $p: \Omega \to [0,1]$ such that $\sum_{\omega \in \Omega} p(\omega) = 1$ // probability of outcome

  An event is a subset $E \subseteq \Omega$, and we define $\Pr[E] = \sum_{\omega \in E} p(\omega)$

  A random variable X is a function $X : \Omega \to R$ // "result" of outcome

Probabilistically checkable proofs



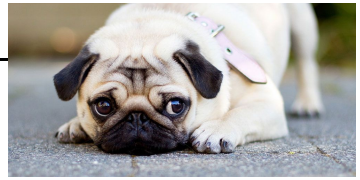
› Probabilistic machines, version 1:

The machine can go into a special “coin toss” state, the output of the machine is a random variable. // over the sample space of toss outcomes

$x \in L \quad \Rightarrow \quad \Pr[\, M(x) = 1 \,] = 1$

$x \notin L \quad \Rightarrow \quad \Pr[\, M(x) = 1 \,] \leq \frac{1}{2}$

Probabilistically checkable proofs



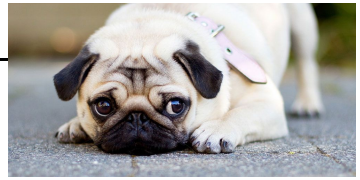
› Probabilistic machines, version 2:

The machine gets a second input, used as random choices, and we analyze the machine's behavior when the second input is random.

$x \in L \quad \Rightarrow \quad \Pr_r[\, M(x,r) = 1 \,] = 1$

$x \notin L \quad \Rightarrow \quad \Pr_r[\, M(x,r) = 1 \,] \leq \frac{1}{2}$

Probabilistically checkable proofs



› Probabilistic machines, version 2:

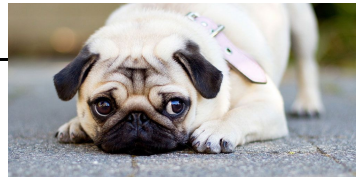
The machine gets a second input, used as random choices, and we analyze the machine's behavior when the second input is random.

$x \in L \quad \Rightarrow \quad \Pr_r[\, M(x,r) = 1 \,] = 1$

$x \notin L \quad \Rightarrow \quad \Pr_r[\, M(x,r) = 1 \,] \leq \frac{1}{2}$

› For this course the versions are equivalent, we'll use version 2

Probabilistically checkable proofs



› Theorem (error reduction):

If L can be decided by a probabilistic polytime machine with error ½, then for any constant $\varepsilon > 0$, L can be decided by a probabilistic polytime machine with error ε.

Probabilistically checkable proofs

- Definition (informal):

  We say that $L \subseteq \{0,1\}^*$ has a PCP verifier **with O(1) queries** if there's a polynomial-time oracle machine V st:

  - $x \in L \Rightarrow \exists\ \pi$ st $V^{\pi}(x,r)$ accepts (wp over r)
  - $x \notin L \Rightarrow \forall\ \pi$ $V^{\pi}(x,r)$ rejects (whp over r)
  - $V^{\pi}(x)$ runs in polytime and makes O(1) queries to π

[1] we define PCP verifiers as making queries non-adaptively; more general definitions are possible

Probabilistically checkable proofs

› Definition:

We say that $L \subseteq \{0,1\}^*$ has a PCP verifier **with randomness ℓ and O(1) queries** if there's a polynomial-time oracle machine V st:

› $x \in L \Rightarrow \exists\ \pi \in \{0,1\}^{O(2^{\wedge}\ell(n))}, \quad \Pr_{r \in \{0,1\}^{\wedge}\ell(n)} [\ V^{\pi}(x,r) = 1\ ] = 1$

› $x \notin L \Rightarrow \forall\ \pi \in \{0,1\}^{O(2^{\wedge}\ell(n))}, \quad \Pr_{r \in \{0,1\}^{\wedge}\ell(n)} [\ V^{\pi}(x,r) = 1\ ] \leq ½$

› $V^{\pi}(x)$ runs in polytime and makes O(1) queries to π

[1] we define PCP verifiers as making queries non-adaptively; more general definitions are possible

Probabilistically checkable proofs

- PCP verifier as “peeking into π through a window”:
  - for each choice of $r \in \{0,1\}^{\ell}$ the verifier chooses k = O(1) locations in π to look at
  - think of the k locations as “a k-sized window into π“ (“window” = indices in π)
  - the verifier decides based on x,r and the values it sees in π “through the window”

Probabilistically checkable proofs

- Comments:
  - main parameter: randomness (or proof length)
  - no use in proofs of length > $O(2^{\ell})$

    (2^{ℓ} windows of size k, so at most $k \cdot 2^{\ell}$ choices of indices to query)
  - error reduction, go from ½ to 2^{-k} by repeating k times

Probabilistically checkable proofs

- Claim:

  If L has a PCP verifier with randomness O(1) and O(1) queries, then $L \in \mathbb{P}$.

Probabilistically checkable proofs

- Claim:

  If L has a PCP verifier with randomness O(log n) and O(1) queries, then L $\in \mathbf{NP}$.

Probabilistically checkable proofs

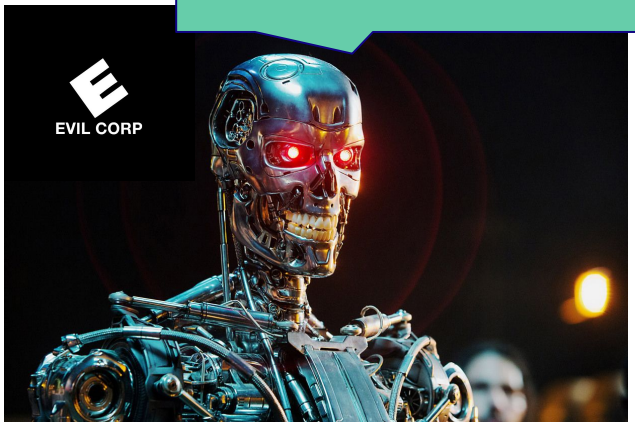
- Theorem (the PCP theorem):

  Every L $\in \mathbf{NP}$ has a PCP verifier with randomness O(log n) and O(1) queries.

Probabilistically checkable proofs

$x \in L$!

trust me I'm a corporation



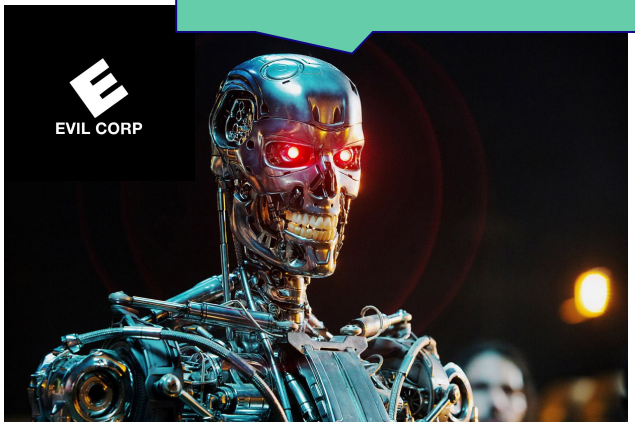
send a probabilistically checkable proof!

$x \in \{0,1\}^n$

Probabilistically checkable proofs

$x \in L$!

trust me I'm a corporation



better yet, publish it on a trusted server, I'll only access O(1) bits

$x \in \{0,1\}^n$

Probabilistically checkable proofs

- Crucial points:
  - for $L \in \mathbf{NP}$, the "PCP proof" π may look very different than proofs that the NP-verifier expects.
  - think of "highly encoded" proofs with a lot of structure
  - known PCPs use algebra & pseudorandom mathematical objects (e.g., expander graphs)

Toy example

- Linearity testing:

  How can we “force” the proof π to be a linear function, using only $O(1)$ queries to it?

- linear function = “π has rigid structure”

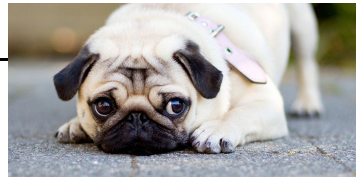
Linearity testing + self-correction

- for simplicity, assume $|\pi| = 2^{\ell}$
- parse π as a truth-table of a function

  $$f = f_{\pi}:\{0,1\}^{\ell} \to \{0,1\} \ , \qquad f(x) = \pi_x$$

- a function f is linear if $f(x) = \sum_{i \in [\ell]} a_i \cdot x_i \pmod 2$,
  for some coefficients $a_i \in \{0,1\}$
- arithmetic from now on is always mod 2

Linearity testing + self-correction



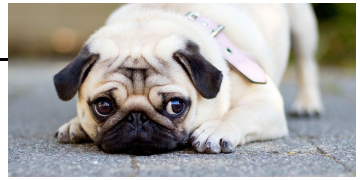
- Observation:

  If f is linear, then $\forall$ x, y $\in \{0,1\}^{\ell}$ we have f(x) + f(y) = f(x+y)

- Lemma (without proof):

  If $\Pr_{x,y \in \{0,1\}^{\wedge}\ell}[\, f(x) + f(y) = f(x+y) \,] \geq 1 - \varepsilon$, then there is a linear function h st f agrees with h on $1 - \varepsilon$ of inputs.

Linearity testing + self-correction



- Step 1 (linearity testing):

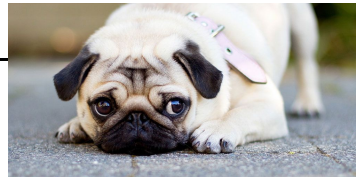
  Choose $x,y \in \{0,1\}^{\ell}$ at random, check if $f(x) + f(y) = f(x+y)$.

  Repeat this for $k = O(1/\varepsilon)$ times.

- Claim:

  With probability at least 0.99 over the 2k choices, if all tests pass, then there is a linear h that agrees with f on $1 - \varepsilon$ of inputs.

Linearity testing + self-correction



- Step 2 (self-correction):

  When we want to query $f = f_\pi$ at point $x \in \{0,1\}^\ell$,
  choose $y \in \{0,1\}^\ell$ at random and output $f(y) + f(x+y)$.

- Claim:

  If there is a linear h that agrees with f on $1 - \varepsilon$ of inputs, then
  for every $x \in \{0,1\}^\ell$ we have $\Pr_y[\, f(x) = f(y) + f(x+y) \,] \geq 1 - \varepsilon$.

Linearity testing + self-correction

- Moral:

  after O(1) queries, we can effectively treat π as representing a linear function

Key connection

PCPs

hardness of approximating problems in NP

Hardness of approximation

- Definition:

  Fix a k-CSP Φ over variable-set $x = x_1,\ldots,x_n$ with constraints $C_1,\ldots,C_m$.
  The (1,½)-approximation problem of Φ calls for deciding between:

  - "YES": $\exists$ x st all constraints are satisfied
  - "NO": $\forall$ x, at most ½ of the constraints are satisfied

Hardness of approximation

- Definition (obvious extension to $0 < s < c < 1$) :

  Fix a k-CSP Φ over variable-set $x = x_1,\ldots,x_n$ with constraints $C_1,\ldots,C_m$.
  The (c,s)-approximation problem of Φ calls for deciding between:

  - "YES": $\exists$ x st at least c/m of constraints are satisfied
  - "NO": $\forall$ x, at most s/m of constraints are satisfied

Hardness of approximation

- Theorem:

  For any L, the following two statements are equivalent:

  1. L has a PCP verifier with rand. O(log n) & k=O(1) queries
  2. L is Karp-reducible to (1,½)-approximating k-CSP

Hardness of approximation

- Corollary:

  If $\mathbf{P} \neq \mathbf{NP}$, for a large enough k = O(1) there's no polytime alg for (1, ½)-approximating k-CSP.

Hardness of approximation

- Corollary:

  If $\mathbf{P} \neq \mathbf{NP}$, for a large enough k = O(1) there's no polytime alg for (1, ½)-approximating k-CSP.

- Corollary (relying on error-reduction):

  If $\mathbf{P} \neq \mathbf{NP}$, for any constant $\varepsilon > 0$ and large enough $k = O_{\varepsilon}(1)$ there's no polytime alg for $(1, \varepsilon)$-approximating k-CSP.

Hardness of approximation

- Theorem (stated without proof):

  If $\mathbf{P} \neq \mathbf{NP}$, then there is no polynomial-time machine that can solve (1, 1-ε)-approximation for 3SAT, for some $\varepsilon > 0$.

- Theorem (informal):

  For many problems in $\mathbf{NP}$, if $\mathbf{P} \neq \mathbf{NP}$, then we know exactly which approximation parameters for them are achievable in $\mathbf{P}$ and which approximation parameters for them are $\mathbf{NP}$-hard.

Hardness of approximation

- Theorem (tutorials):

  If $\mathbf{P} \neq \mathbf{NP}$, then for some s(n) = Ω(n), there's no polytime alg that gets a graph G=([n],E) and decides between the cases:

  - "YES": G has a clique of size s(n)
  - "NO": G has no clique of size s(n)/2