Probabilistically Checkable Proofs (PCP)

Madhu Sudan
MIT

The Theorem

- $\exists$ a format for writing proofs ...
- ... and a probabilistic method to verify the proofs, ...
- such that the verification method reads only 3 bits of the proof, and guarantees:
  - Completeness: Correct proofs are accepted with probability 1.
  - Soundness: "Proofs" of incorrect assertions accepted with probability at most $\frac{3}{4}$.
- Furthermore, the new proof is only polynomially longer than "classical" ones.

Significance

- Surprising co-existence of probability and logic!
- Major impact on the study of combinatorial optimization!

Consequence: For many NP-hard combinatorial optimization problems finding near optimal solutions is also NP-hard.

Our Goal

Insight into construction of these proofs.

And more importantly, the verifier!

Will not pursue consequence to combinatorial optimization.

PCP: The definition

- Verifier: Prob. poly-time algorithm V,
  with oracle-access to "proof".

- (r, q)-restricted:
  - On input x of length n.
  - Tosses $r(n)$ random coins.
  - Queries $q(n)$ bits from oracle.

- $L \in \mathrm{PCP}_{c,s}[r, q]$ if:

  Completeness $x \in L \Rightarrow$
  $\exists\ \pi$, s.t. $V^{\pi}(x)$ accepts w.p. $\geq c$.

  Soundness $x \notin L \Rightarrow$
  $\forall\ \pi$, $V^{\pi}(x)$ accepts w.p. $< s$.

Notes on Definition

- Four parameters: c, s, r, q.
- Randomness vs. size: Roughly:
  Size = Exponential in Randomness.
- Only one subscript implies $c = 1$.
- No subscripts implies $c = 1, s \approx \frac{1}{2}$.

History of Definition

- Originates from IP.
  [Goldwasser, Micali, Rackoff 1985]
  [Babai 1985]

- Probabilistic Oracle Machine Verifier:
  [Fortnow, Rompel, Sipser '88]

- Transparent/Holographic Proofs: Focus on computation time of verifier and size of proof.
  [Babai, Fortnow, Levin, Szegedy 1991]

- PCP: Implicit
  [Feige, Goldwasser, Lovasz, Safra, Szegedy 1991]
  Explicit in [Arora and Safra, 1992]

History of results

Phase 0: Triviality:

$$\text{NP} = \text{PCP}[0, \text{poly}(n)] = \text{PCP}[\log n, \text{poly}(n)]$$

Phase 1: Opening results:

$$\text{NP} \subseteq \text{PCP}[\text{poly} \log n, \text{poly} \log n]$$

[Babai, Fortnow, Lund 1990]
[BFLS 1991]
[FGLSS 1991]

+ Exponential reduction in queries.
– Superlogarithmic randomness and hence not an equality
([BFLS 1990]: polynomial sized proofs.)

History of results (contd.)

Phase 2: Characterizations:

$$\mathsf{NP} = \mathsf{PCP}[O(\log n), O(1)]$$

[AS 1992]
[Arora, Lund, Motwani, Sudan, Szegedy 1992]

+ Asymptotically optimal in both parameters.
− Not tight!

How to abuse notation

$$\mathsf{NP} = \mathsf{PCP}[O(\log n), O(1)]$$

Actually means

$\exists c_q$ such that
$\forall L \in \mathsf{NP}$,
$\exists c_r$ such that

$$L \in \mathsf{PCP}[c_r \cdot \log n, c_q]$$

History of results (contd.)

Phase 3: (Near-) Optimal(?) Characterizations:

- $\mathrm{SAT} \in \mathsf{PCP}[(1+\epsilon)\log n, O(1)], \quad \forall \epsilon > 0$
  [Polishchuk Spielman 1994]

- $\mathsf{NP} = \mathsf{PCP}_{1-\epsilon,\frac{1}{2}}[O(\log n), 3] \quad \forall \epsilon > 0$
  [Hastad 1997]

Towards the Proof

- Will roughly follow historical path.
- First show main steps in

$$\mathsf{NP} = \mathsf{PCP}[O(\log n), \mathrm{poly}\log n]$$

  - Arithmetization.
  - Low-degree testing.
- Some digressions
- Composition of PCPs, to get:

$$\mathsf{NP} = \mathsf{PCP}[O(\log n), O(1)]$$

$$\mathsf{NP} = \mathsf{PCP}_{1-\epsilon,\frac{1}{2}}[O(\log n), 3]$$

Phase 1

Constraint Satisfaction Problems

Max w-CSP(B):

Instance: n B-ary variables;
t w-ary constraints $C_1, \ldots, C_t$.

Find: assgnt. max'ing # sat. constraints.
(or approximate this number).

CSPs arise naturally in study of PCP.

PCP$[r, q] \equiv$ Max w-CSP(2).

- Bits of proof $=$ variables.
- V's verdict on fixed \$\$ $=$ constraint.
- $n =$ proof size; $t = 2^r$;
  $w = q$ (say queries non-adaptive).
- Acc. prob. $=$ # sat. constraints.

Gap Problems

- Complexity likes Decision Problems.
- How to deal with approximations?
- Answer: Gap problems.

Example: Gap w-CSP$_{c,s}(B)$:

YES instances: $\exists$ assignment satisfying c fraction of constraints.
NO instances: No assignment satisfies s fraction of the constraints.

Task: Distinguish YES from NO.

Goal: Show Gap 3-CSP$_{1-\epsilon,\frac{1}{2}}(2)$ is NP-hard.

Polynomial Constraint Satisfaction

– $B = \mathbb{F}$ — a finite field.
– $n = \mathbb{F}^m$: m-dimensional vector space.
– Assignments: $f : \mathbb{F}^m \to \mathbb{F}$.

w-Polynomial Constraint Satisfaction$(d, \mathbb{F})$:

Instance: w-ary constraints on functions $f : \mathbb{F}^m \to \mathbb{F}$.
Goal: Find m-variate polynomial f of total degree at most d that satisfies most constraints.

- Constraints "easy" to verify.

- Essentially a constraint satisfaction problem. But has low-degree restriction.

Computational complexity of PCS

Defn: Gap $\mathrm{PCS}_{1,\epsilon}(m, w, d, q)$:
Instances restricted to

$m = m(t), d = d(t), w = w(t)$,

and $|\mathbb{F}| = q(t)$

(With completeness/soundness as usual).

Lemma 1: $\forall \epsilon > 0$, Gap $\mathrm{PCS}_{1,\epsilon}(m, w, d, q)$
is NP-hard, for

$w, d, q = \mathrm{poly} \log t$

and $m = \frac{\log t}{\log \log t}$

Comp. Comp. of PCS (contd.)

+ We have a gap!

+ $q^m = \text{poly}(t)$; Thus, table of values of f is a reasonable proof to ask for.

+ Can pick random constraint and verify if it is satisfied in time $\text{poly}\log t$.

– Low-degree restriction on f?

Low-degree Testing

Problem Definition

Given: $d \in \mathbb{Z}^+$; and oracle $f : \mathbb{F}^m \to \mathbb{F}$.

Task: Verify f is a degree $\leq d$ poly,
in time $\mathrm{poly}(m, d)$. I.e.,

Completeness: $\deg(f) \leq d \Rightarrow$ accept.
Soundness: $\deg(f) > d \Rightarrow$ reject w.h.p.

- Not possible! Have to weaken soundness.

Low-degree Testing (contd).

Defn: f, g are δ-close if

$$\Pr_x[f(x) \neq g(x)] \leq \delta.$$

Problem Definition (revised)

Given: δ, d; oracle $f : \mathbb{F}^m \to \mathbb{F}$.

Verify: f close to deg $\leq d$ poly, I.e.,

Completeness: $\deg(f) \leq d \Rightarrow$ accept.
Soundness: Reject w.h.p. if f is not δ-close to any deg. d polynomial.

Lemma 2: $\exists$ A low-degree test with running time $\mathrm{poly}(m, d, \frac{1}{\delta})$ and tosses $O(m \log |\mathbb{F}|)$ random coins.

Self-correction of polynomials

Problem Definition:

Given: δ, d, $x \in \mathbb{F}^m$; oracle $f : \mathbb{F}^m \to \mathbb{F}$ s.t. f is δ-close to poly p of degree d.

Task: Compute $p(x)$.

(Is p even uniquely specified?)

Lemma 3: There exists a randomized Self-corrector for polynomials that runs in time $\mathrm{poly}(m, d, \frac{1}{\delta})$ and tosses $O(m \log |\mathbb{F}|)$ random coins, provided δ is suff. small.

A PCP Verifier for NP Languages

- Given x, supposedly in L.
- Compute ϕ supposedly in Gap PCS.
- Prover supplies oracle for assignment f.
- Run Low-degree Test on f. Reject if it rejects.
- Pick random constraint C of ϕ and verify that Self-correct(f) satisfies C. Reject if not.
- Accept otherwise.

Analysis (modulo Lemmas 1-3)

- Queries $\text{poly}\log|x|$ bits.
- Tosses $O(\log|x|)$ coins.
- Has $c = 1$ and $s \ll \frac{1}{2}$.

Theorem: $\mathsf{NP} \subseteq \mathsf{PCP}[\log n, \text{poly}\log n]$.

Low-degree Polynomials

Agenda

- Prove Lemma 1, 2, and 3.
- Lemma 1: Something about polynomials.
- Lemma 2: Something about polynomials.
- Lemma 3: Something about polynomials.

- Polynomials?

Robustness of PCPs

- Can flip 1% of the bits of proof, without significantly altering its power to convince.
- Must have some error-correction built into them.
- Inherit this property from error-correcting property of polynomials.

Encoding with (univariate) polynomials

- To encode $a_1, \ldots, a_n \in \{0, 1\}$.
- Pick $\mathbb{F}$ of size $\approx n^2$.
- Find polynomial f of degree n s.t.

$$f(1) = a_1, \ldots, f(n) = a_n$$

- Then $\langle f(x) \rangle_{x \in \mathbb{F}}$ is a redundant representation of $a_1, \ldots, a_n$.
- Specifically: Can reconstruct info even if $|\mathbb{F} - n|$ values are erased, or half as many are erroneous.

Univ. Polys (contd.)

+ Lots of redundancy.

– High algebraic degree!

– $\Rightarrow$ low-degree test takes $\Omega(n)$ queries.

- Do better?

Bivariate polynomials

– Info $= a_{11}, \ldots, a_{\sqrt{n}\sqrt{n}} \in \{0,1\}$.
– Pick $\mathbb{F}$ of size $\approx n$.
– Find polynomial $f(\cdot,\cdot)$ of degree $\sqrt{n}$ in each variable s.t.

$$f(i,j) = a_{ij}$$

– Then $\langle f(x,y)\rangle_{x,y\in\mathbb{F}}$ is a redundant representation of $a_1, \ldots, a_n$.

m-variate polynomials

- Pick $\mathbb{F}$ of suff. cardinality.
- Pick $H \subseteq \mathbb{F}$ of size $n^{1/m}$.
- Info $a : H^m \to \{0, 1\}$.
- Find m-variate polynomial f
  of degree $|H|$ in each variable
  s.t. $f(x) = a(x)$ for each $x \in H^m$.
- Schwartz-Zippel lemma yields redundancy of f.
- f is a low-degree extension of a.

Choosing parameters

- Pick $m(n) = O(\frac{\log n}{\log \log n})$.
- Yields $|H| = \text{poly} \log n$.
- Thus $\deg(f) = \text{poly} \log n$.
- Can work with $|\mathbb{F}| = \text{poly} \log n$ ...
- ... and still have size of encoding
  $= \mathbb{F}^m = \text{poly}(n)$.
- Will stick with such choices for this hour.

Hardness of Gap PCS

Recall: Gap PCS

Gap $\text{PCS}_{1,\epsilon}(m, w, d, q)$:

Instance: w-ary constraints on $f : \mathbb{F}^m \to \mathbb{F}$.

YES: Some deg d poly. sat. all constraints.

NO: No poly sat. ϵ-fraction of constraints.

Lemma 1: Gap $\text{PCS}_{1,\epsilon}(m, w, d, q)$ is NP-hard
$w, d, q = \text{poly} \log t$, $m = \frac{\log t}{\log \log t}$.

Arithmetizing 3-SAT

- 3-SAT:
  Instance: n variables t clauses $C_1, \ldots, C_t$,
  where $C_j : x_{i_1} = b_1$ or $x_{i_2} = b_2$
  or $x_{i_3} = b_3$.

- Thus $\phi : [n]^3 \times \{0,1\}^3 \to \{0,1\}$.

- Set $[n] = H^m$, extend $\{0,1\}$ to H, to get:
  $\phi : H^l \to \{0,1\}$, where $l = 3m + 3$.
  ($\phi(\cdots) = 0$ if args don't make sense, or clause does not appear.)

- Restating 3-SAT: Want $a : H^m \to \{0,1\}$
  s.t. $\forall\ i_1, i_2, i_3 \in H^m,\ b_1, b_2, b_3 \in H$
  $\phi(i_1, i_2, i_3) = 0$ or $A(i_1) = b_1$
  or $A(i_2) = b_2$ or $A(i_3) = b_3$

Arithmetizing 3-SAT (contd.)

- Let $\hat{\phi}$, A = low-degree extensions of ϕ, a.

- Want $A : \mathbb{F}^m \to \mathbb{F}$, $P_1 : \mathbb{F}^l \to \mathbb{F}$ s.t.
  $(C0) : \forall z = \langle i_1, i_2, i_3, b_1, b_2, b_3 \rangle \in \mathbb{F}^l$
  $$P_1(z) = \hat{\phi}(z) \cdot (A(i_1) - b_1) \cdot (A(i_2) - b_2) \cdot (A(i_3) - b_3)$$

- And
  $(C1') : \forall z = \langle i_1, i_2, i_3, b_1, b_2, b_3 \rangle \in H^l$
  $$P_1(z) = 0$$

Towards Gap PCS

+ Have hardness of a PCS problem (actually, two polynomials but thats ok).

– No gap!

+ (C0) is a robust collection of constraints: Either all satisfied or very few.

– (C1') is not!

Robustness of (C0)

Claim: If $A, P_1, \hat{\phi}$ are degree d polynomials
that violate (C0)(z) for some z,
then they violate (C0)(z) for
$\geq (1 - \frac{4d}{|\mathbb{F}|})$ fraction of the z's.

Proof: Let

$$Q_1(z) = \hat{\phi}(z)(A(i_1) - b_1)(A(i_2) - b_2)$$
$$(A(i_3) - b_3)$$

Then $\deg(P_1), = deg(Q_1) \leq 4d$.

By Schwartz-Zippel lemma, if they disagree
then they disagree w.p. $\geq (1 - \frac{4d}{|\mathbb{F}|})$.

Percolating errors in (C1')

- Let $z_1, \ldots, z_l$ denote variables of P_1.
- Let $P_2(r_1, {\scriptstyle z_2, \ldots, z_l}) = \sum_{j=1}^{|H|} r_1^j P_1(j, {\scriptstyle z_2, \ldots, z_l})$.
- Fixing $z_2, \ldots, z_l$, have a polynomial in r_1.
- Identically zero iff P_1 is zero on H.
- Thus $P_2 = 0$ over $F \times H^{l-1}$ iff $P_1 = 0$ over H^l.

Percolating errors (contd.)

- Conditions:

  (C1) $\forall z = \langle r_1, z_2, \ldots, z_l \rangle \in F^l$
  $$P_2(z) == \sum_{j=1}^{|H|} r_1^j P_1(j, z_2, \ldots, z_l).$$

  (C2') $\forall z \in F \times H^{l-1} P_2(z) = 0.$

$+$ (C1') replaced by (C1) and (C2').

$+$ (C1) robust,

$-$ (C2') not robust, but getting better.

Percolating (contd).

- For $i = 1, \ldots, l$ let

$$P_{i+1}(r_1, \ldots, r_{i-1}, r_i, z_{i+1}, \ldots, z_l) = \sum_{j=1}^{|H|} r_i^j P_i(r_1, \ldots, r_{i-1}, j, z_{i+1}, \ldots, z_l).$$

- Constraints:

  (Ci) enforcing above

  (C(i+1)'):$\forall z \in F^i \times H^{l-i}, P_{i+1}(z) = 0$

  (Ci') replaced by (Ci) and (C(i+1)').

+ (C(l+1)) = (C(l+1)') is robust!

Tidying up

- Have many polynomials! What to do with them?
- Let $P : F^{l+1} \to F$ unify all with $P(i, \cdots) = P_i(\cdots)$ etc.
- Thus 3-SAT reduces to:

  Find P of degree $O(d + l|H|)$ such that for every $z \in f^l$, P satisfies condition (C)(z),

  where (C)$(z) = \wedge_i (Ci)(z)$
- Thus we prove Lemma 1.

Low-degree testing

Recall: Low-degree Testing

Problem Definition

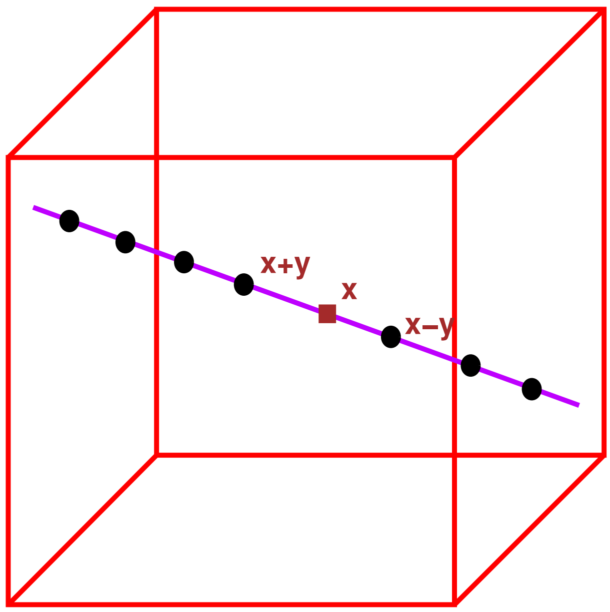
Given: δ, d; oracle $f : \mathbb{F}^m \to \mathbb{F}$.

Verify: f close to deg $\leq d$ poly, I.e.,

Completeness: $\deg(f) \leq d \Rightarrow$ accept.
Soundness: Reject w.h.p. if f is not δ-close to any deg. d polynomial.

Lemma 2: $\exists$ A low-degree test with running time $\mathrm{poly}(m, d, \frac{1}{\delta})$ and tosses $O(m \log |\mathbb{F}|)$ random coins.

The Testing Algorithm



- Let $f_{x,y}(t) = f(x + ty)$.

- f is a degree d polynomial iff for every pair $x, y \in \mathbb{F}^m$, $f_{x,y}$ is a polynomial of degree d.

- Test: Pick random x, y and verify that $f_{x,y}$ is a degree d polynomial.

Analyzing (a weaker) test

Test': Pick $x, y \in \mathbb{F}^m, t \in \mathbb{F}$ at random. Ask prover for coeffs of the "polynomial" $f_{x,y}$. Verify $f_{x,y}(t) = f(x + ty)$.

Theorem:

Easy part: If f is a degree d poly, then there exists responses $f_{x,y}$ s.t. Test' always accepts.
Hard part: $\exists \delta > 0$ s.t. for all $m, d, \mathbb{F}$, if f is any function s.t. there exists responses $f_{x,y}$ s.t. Test' accepts with probability $1 - \delta$, then f is 2δ-close to some degree d polynomial.

Self-correction

Recall: Self-correction

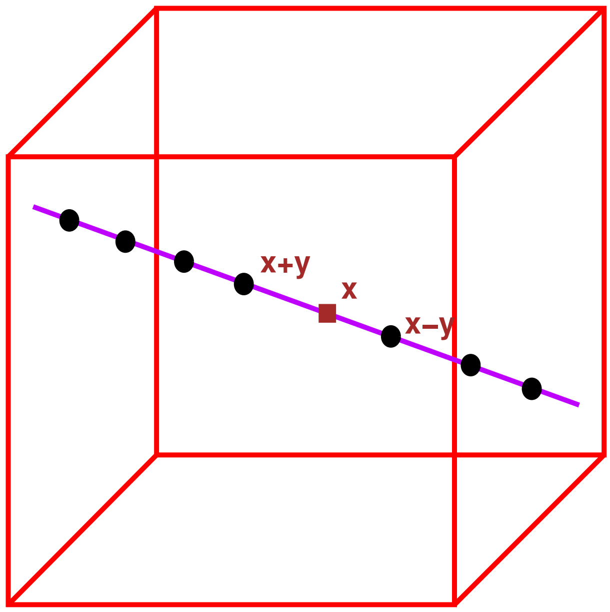
Problem Definition:

Given: δ, d, $x \in \mathbb{F}^m$; oracle $f : \mathbb{F}^m \to \mathbb{F}$ s.t. f is δ-close to poly p of degree d.

Task: Compute $p(x)$.

Lemma 3: There exists a randomized Self-corrector for polynomials that runs in time $\text{poly}(m, d, \frac{1}{\delta})$ and tosses $O(m \log |\mathbb{F}|)$ random coins, provided δ is suff. small.

Self-correction Algorithm



- Pick $y \in \mathbb{F}^m$ at random.
- Query $f(x+y), \ldots, f(x+(d+1)y)$.
- Let $b_1, \ldots, b_{d+1}$ be responses.
- Compute $h(t)$ of degree d s.t. $h(i) = b_i$.
- Output $h(0)$.

Analysis

- If f is a degree d polynomial, then obviously output is right!
- But f is only δ-close to the polynomial p.
- Still $x + iy$ is a random point in $\mathbb{F}^m$.
- Thus, $\mathrm{Pr}_y[f(x+iy) \neq p(x+iy)] \leq \delta$.
- Thus, $\mathrm{Pr}_y[\exists i, \quad f(x+iy) \neq p(x+iy)] \leq (d+1)\delta$.
- With probability, $1-(d+1)\delta$ we are "obviously" right.

Concluding Phase 1

- Recall had to prove three things about polynomials.
- We proved three things.
- With any luck, these multisets are identical.
- Done with Phase 1.

Digression I: Multiprover Interactive Proofs (MIP)

p-prover One round MIPs

Informally: What can probabilistic verifier interacting with p-provers verify, if allowed to ask one question each?

Formal: (p, r, a)-restricted MIP verifier, asks one question to p provers and gets a-bit answers. Tosses r-coins.

$L \in \mathrm{MIP}_{c,s}[r, p, a]$ if there exists an (r, p, a)-restriced MIP verifier checking $x \in L$ with completeness c and soundnesss.

MIP: Motivation

- Clean model in scope of interactive proofs.
- More restrictive than PCP:
  $\mathsf{MIP}[p, r, a] \subseteq \mathsf{PCP}[r, pa]$
- Necessary:
  $\mathsf{PCP}[r, q] \subseteq \mathsf{MIP}_{1-\frac{1}{2q}}[2, r + \log q, q]$
- Central intermediate step in our goal.
- For now, a digression.

A 3-prover MIP for NP

- Recall PCP$[O(\log n), \text{poly} \log n]$ verifier.
  - Performs low-degree test. But this is already a 2-prover 1-round protocol. Ask first prover for value of P on point and another for $P_{x,y}$ on some line.
  - Checks random constraint. Too many questions. Can't ask them together since this reveals the constraint.

- How to obtain value of P, as defined by first prover, at w places?

Reconstruction via curves

- Solution: To query $P(x_1), \ldots, P(x_w)$ find a "random curve" $\mathcal{C}$ through $x_1, \ldots, x_w$ and ask third prover for $P|_{\mathcal{C}}$.

- Either prover responds with $P|_{\mathcal{C}}$, in which case we are fine, or responds with wrong polynomial h in which case at random point t, $P(\mathcal{C}(t)) \neq h(t)$.

Curves through $\mathbb{F}^m$

- Curve $\mathcal{C} : \mathbb{F} \to \mathbb{F}^m$ is a collection of m functions $\mathcal{C}_i : \mathbb{F} \to \mathbb{F}$.
- $\deg(\mathcal{C}) = \max_i\{\deg(\mathcal{C}_i)\}$.
- Prop: For any $x_0, x_1, \ldots, x_w$ there exists a degree w curve $\mathcal{C}$ with $\mathcal{C}(j) = x_j$. (By interpolation.)
- Prop: P of degree d and $\mathcal{C}$ of degree w implies $P|_\mathcal{C}$ of degree wd.
- Prop: For every $x_1, \ldots, x_w$ and random x_0 if $\mathcal{C}(t)$ is a random point in $\mathbb{F}^m$ for $t \notin \{1, \ldots, w-1\}$. (Call such a curve a random curve through $x_1, \ldots, x_w$.)

Summarizing MIP verifier

- Given: Instance of Gap PCS.
  - Pick a constraint C_j at random.
  - Determine random curve through points of P that C_j depends on.
  - Pick random point x on $\mathcal{C}$.
  - Pick random line $l_{x,y}$ through x.
  - Queries:
    $P(x) \rightarrow \Pi_1$, $P_{x,y} \rightarrow \Pi_2$, and $P_{\mathcal{C}} \rightarrow \Pi_3$.
  - Let a, g and h denote responses.
    Reject if $g(x), h(x)$ not equal to a;
    Reject if values of h at $1, \ldots, w$ do not satisfy C_j.
    Accept otherwise.

Steps in analysis

- Consider NO instance.

- Let $\tilde{P}$ denote response of Π_1.
  If $\tilde{P}$ not close to deg. d poly. P,
  then with non-zero prob. $g(x) \neq a$.
  (By low-degree test.)

- $\tilde{P}$ close to P:
  If C_j satisfied by P, then give up. Else:
  - If $h \neq P|_{\mathcal{C}}$, then with non-zero prob. $h(x) \neq P(x)$, while $P(x) = \tilde{P}(x)$.
  - If $\tilde{P}$ is close to some degree d polynomial P and $h = P|_{\mathcal{C}}$, then verifier rejects.

Theorem: $\exists\ \gamma > 0$ s.t.
$\text{NP} \subseteq \text{MIP}_{1,1-\gamma}[3, O(\log n), O(\text{poly} \log n)]$.

Digression II:

$\mathsf{NP} \subseteq \mathbf{PCP}[\mathrm{poly}(n), O(1)]$

Circuit Satisfiability

Ckt-SAT

Instance: Boolean circuit K on k inputs
with $(n-k)$ "$\wedge_2, \oplus_2,$ NOT"-gates.

Goal: Decide if K has a satisfying input.

Well-known: Ckt-SAT is NP-Complete.

Proving membership in Ckt-SAT

- Present satisfying assignment $a_1, \ldots, a_k$.
- OR present satisfying assignment AND output of all gates: $a_1, \ldots, a_k, \ldots, a_n$.
  - Latter easy to verify.
  - If Gate$_j$ has inputs a_{i_1}, a_{i_2} and outputs a_{i_3} then the output should be consistent with the inputs and the operation of the gate.
  - Output gate should output 1.

Satisfiability of Quadratic Polynomials

(An algebraic NP-complete problem)

Instance: t degree 2 polynomials $P_1, \ldots, P_t$ on n elements from $\mathbb{F}_2$.

Goal: Do polynomials have common zero?

Proof (of NP-completeness):

– Variables correspond to values $a_1, \ldots, a_n$.
– Polynomials correspond to checking gate consistency and checking output.

Conclusion: To prove Ckt-SAT, we may prove corresponding polynomials are zero. Will ask prover to write down an appropriate encoding of $a_1, \ldots, a_n$.

The Hadamard Code

- To encode $a = \langle a_1, \ldots, a_n \rangle$.
- Define $A : \mathbb{F}_2^n \to \mathbb{F}_2$ as $A(x) = \sum_{i=1}^{n} a_i x_i$.
- $\langle A(x) \rangle_{x \in \mathbb{F}_2^n}$ is Hadamard encoding of a.
- Claim: Encodings of distinct strings differ in exactly half the bits. (Why?)

- Encodes n bits with 2^n bits.
  But this is not enough.

The Quadratic Functions Code

- Re-viewing Hadamard code:
  Encoding$(a) = \langle L(a) \rangle_L$
  over all linear functions L.
  (Each x corresponds to χ_x
  where $\chi_x(a) = \sum_{i=1}^{n} a_i x_i$.)

- The quadratic function code:
  $\mathcal{Q}(a) = \langle Q(a) \rangle_Q$
  Q ranges over deg. 2 polys.

- Explicitly: a defines $B : \mathbb{F}_2^{n^2} \to \mathbb{F}_2$.
  $B(Q_{11}, \ldots, Q_{nn}) = \sum_{ij} Q_{ij} a_i a_j$.
  and $\mathcal{Q}(a) = \langle B(Q) \rangle_Q$.

- Clearly correlated with satisfiability of quadratic polynomials.

Correcting the Hadamard/Q.F. Code

Given: $x \in \mathbb{F}_2^n$ and oracle $\tilde{A} : \mathbb{F}_2^n \to \mathbb{F}_2$
$\tilde{A}$ is δ-close to linear function A.

Task: Compute $A(x)$.

Lemma 3': Can do so w.p. $1 - 2\delta$
with 2 queries to $\tilde{A}$.

Algorithm: Pick $y \in \mathbb{F}_2^n$ at random.
Output $\tilde{A}(x + y) - \tilde{A}(y)$.

Correctness, and extension to Q.F. code, straightforward.

Testing the Hadamard Code

Given: δ, oracle $\tilde{A} : \mathbb{F}_2^n \to \mathbb{F}_2$

Task: Test if $\tilde{A}$ is δ-close to linear A.
Completeness: Accept if $\tilde{A}$ is linear.
Soundness: If $\tilde{A}$ not δ close to any linear function then reject w.p. at least ϵ.

Lemma 2.1': Can do so with 3 oracle queries to $\tilde{A}$ and $\epsilon = 1 - \delta$.

Algorithm: Pick $x, y \in \mathbb{F}_2^n$ at random.
Accept if $\tilde{A}(x) = \tilde{A}(x + y) - \tilde{A}(y)$.

Correctness NOT straightforward.
Shown by [Blum, Luby, Rubinfeld, 1990].

Testing the Hadamard + Q.F. Code

Given: $\tilde{A} : \mathbb{F}_2^n \to \mathbb{F}_2$, $\tilde{B} : \mathbb{F}_2^{n^2} \to \mathbb{F}_2$ and δ.

Task: Test if $\exists\ a \in \mathbb{F}_2^n$ s.t.
$\tilde{A}, \tilde{B}$ are δ-close to A, B, where
A, B: Hadamard + Q.F. encodings of a.

Lemma 2': Can test with 10 queries to $\tilde{A}, \tilde{B}$.

Testing the Hadamard + Q.F. Code (contd).

Algorithm:

1. Test $\tilde{A}$ is δ-close to linear.
2. Test $\tilde{B}$ is δ-close to linear, i.e., $\exists b_{ij}$ s.t. $\tilde{B}$ δ-close to $B(Q) = \sum b_{ij} Q_{ij}$.
3. Pick random $x, y \in \mathbb{F}_2^n$ and let $Q_{ij} = x_i y_j$.
4. Verify $\underline{B}(Q) = \tilde{A}(x)\tilde{A}(y)$

(How to compute $B(Q)$?)

Correctness relatively straightforward.

Using the Q.F. Code

- Recall: Need to verify a satisfies $P_1, \ldots, P_t$.
- Ask prover to provide Hadamard + Q.F. encoding of a.
- Test above.
- How to verify all P_j's satisfied?
- Good enough to verify $P_r = \sum_j r_j P_j$ is satisfied for random $r \in \mathbb{F}_2^t$. (Why?)

Summarizing the PCP

- To verify Ckt-SAT, generate $P_1, \ldots, P_t$.
- Prover provides oracles $\tilde{A}, \tilde{B}$ (supposedly for Hadamard and Q.F. encoding of satisfying assignment a).
- Verifier tests Hadamard + Q.F. encoding.
- Verifier uses Self-corrector to simulate oracle for Q.F. encoding B of some (unknown) string.
- Verifies $B(P_r) = 0$ for random $r \in \mathbb{F}_2^n$ and $P_r = \sum_j r_j P_j$.

Theorem: NP $\subseteq$ PCP$_{1-\epsilon}[\mathrm{poly}(n), 12]$, for some $\epsilon > 0$.

An improved PCP: Composition of Proofs

Recall our digressions

- A 3-prover MIP for NP.
  + $O(\log n)$ random coins.
  – $\text{poly} \log n$ answers.
  – Verdict of verifier is an easily computable function of answers: E.g. computable by a circuit of $\text{poly} \log n$ size.

- A 12 query PCP for NP.
  + Soundness < 1.
  – High randomness! (Exponential sized proofs).
  – Verifies circuit satisfiability.
  – Proof $=$ Hadamard $+$ Q.F. code
    of assignment $+$ internal gate values.

Composing the verifiers

- Start with the 3-prover MIP verifier.
- Prepare queries q_1, q_2, q_3 and small circuit C that determines verdict.
- Send queries to provers . . . .
- But don't ask for responses a_1, a_2, a_3.
- Instead ask prover to prove that a_1, a_2, a_3 will satisfy C, using 12 query PCP verifier.

Composing the verifiers (contd.)

- Why is this different from the ingredients?
  - Queries: 12 bits!
  - Randomness: $O(\log n) + \text{poly}\log n)$!
  - Soundness: Oops!

- Flaw: Similar to asking a SAT prover to give assignments to each clause separately. How do we know that different clauses referring to same variable use same assignment to variable? Consistency needs to be checked.

Composing the verifiers (contd.)

- The Fix: Force prover to commit to responses to individual queries (e.g. q_1) by writing down appropriate encoding of answer (e.g. Hadamard encoding of a_1).

- Verifier: In addition to verifying that $C(a_1, a_2, a_3)$ accepts, verify that the first component of the proof is consistent with response to query q_1. How?

- Let encoding of a_1 be $\langle A_1(x) \rangle_x$. Proof supposedly consists of Hadamard (+ Q.F.) encoding of some string $a_1 \circ b$ and is of the form $\langle B(y) \rangle_y$ for some function B.

- Consistency check (for $1st$ coordinate): Pick x at random and verify $A_1(x) =$ Self-Corrector$(B(x0^{|b|}))$.

Summarizing

- PCP Verifier for NP with oracle Π.
  - Pick random string R as per MIP verifier and generate questions q_1, q_2, q_3 and circuit C.
  - Let $A_i(\cdot) = \Pi(i, q_i, \cdot)$ and $B(\cdot) = \Pi(0, R)$.
  - Inner verifier for C with oracles A_1, A_2, A_3, B:
    * Run 12 query PCP on oracle B with input C.
    * Do consistency checks on oracle pairs (A_1, B), (A_2, B), and (A_3, B).

Theorem: $\exists \gamma > 0$ (!) s.t.

$\mathsf{NP} \subseteq \mathsf{PCP}_{1-\gamma}[\text{poly} \log n, 21]$

Composition as a paradigm

Ingredients

- Outer verifier: MIP with small p and C, and *low* soundness error. Answer size governs size of problem sent to inner verifier.

- Inner verifier: PCP with small q, and should verify commitment to a proof (rather than the mere existence of one). Further should admit self-corrector (easy computation of correct bits of proof, given a slightly erroneous proof).

"Theorem": Can compose outer and inner verifier to get a verifier that combines the randomness efficiency of outer verifier with query efficiency of inner verifier.

Concluding Phase 2

The Endgame

(Ignoring error parameter!)

- Empirical observation: Verifiers showing $\mathsf{NP} = \mathsf{MIP}[1, 0, \mathrm{poly}(n)] \subseteq \mathsf{MIP}[p, r, a]$ also yield inner verifiers showing: $\mathsf{MIP}[p', r', a'] \subseteq \mathsf{MIP}[p'+p, r'+r \circ a', a \circ a']$

- In particular the MIP of digression I yields: $\mathsf{MIP}[p', r', a'] \subseteq \mathsf{MIP}[p'+3, r'+\log a', \mathrm{poly}\log a']$

- Using $\mathsf{NP} = \mathsf{MIP}[3, \log, \mathrm{poly}\log]$, we get $\mathsf{NP} = \mathsf{MIP}[6, \log, \mathrm{poly}\log\log n]$

- Composing with inner verifier of Digression II, we get:

  Theorem: $\mathsf{NP} = \mathsf{PCP}[\log, 30]$

Towards Optimal PCPs

Ingredients

- Need better MIPs: Low number of provers, say 2, with low soundness error, while having reasonably small answer sizes.

- Need optimal inner verifiers, tuned to simplifying verifiers of 2-prover proof systems.

A 2-prover MIP for NP

- Starting point: NP $\subseteq$ PCP$_{1-\epsilon}[\log n, 30]$
- Converting to MIP [FRS].
  - Pick random string R and generate queries $q_1, \ldots, q_{30}$. Send *all* queries to Prover 1.
  - Pick random index $i \in \{1, \ldots, 30\}$ and send query q_i to Prover 2.
  - Accept if answers of Prover 1 are acceptable to PCP verifier, and answer of Prover 1 on query q_i consistent with that of Prover 2.
  - Claim: Soundness bounded away from 1.

2-Prover MIPs

+ Number of provers is small.

– Error probability is large.

- Can we reduce error by repetition?
  - Natural repetition is sequential. Increases number of rounds.
  - Can try a parallel repetition. But does error reduce?

Formalizing 2-prover protocols

Game-theoretic model $\mathcal{G}$:

- Finite sets $\Omega, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{A}_1.\mathcal{A}_2$.
- Query functions: Q_1, Q_2, $Q_i : \Omega \to \mathcal{Q}_i$.
- Verdict: $V : \Omega \times \mathcal{A}_1 \times \mathcal{A}_2 \to \{0, 1\}$
- Value of Game:

$$\omega(\mathcal{G}) = \max_{P_1, P_2} \mathsf{Exp}_{R \in \Omega}[V(R, a_1, a_2)]$$

where $P_i : \mathcal{Q}_i \to \mathcal{A}_i$ and $a_i = P_i(Q_i(R))$.

Parallel repetition

- $\mathcal{G}^k = k$-repetition of $\mathcal{G}$:
  - Query functions: Q_1^k, Q_2^k where
    $Q_i^k : \Omega^k \to \mathcal{Q}_i^k$,
    $Q_i^k(R_1, \ldots, R_k) = \langle Q_i(R_1), \ldots, Q_i(R_k)\rangle$.
  - Verdict: $V^k : \Omega^k \times \mathcal{A}_1^k \times \mathcal{A}_2^k \to \{0, 1\}$ with $V^k(\vec{R}, \vec{a}_1, \vec{a}_2) = \prod_i V(R_i, (a_1)_i, (a_2)_i))$.

- $\omega(\mathcal{G}^k) = \omega(\mathcal{G})^k$? No!

Theorem: [Raz '95] For every $\mathcal{A}_1, \mathcal{A}_2, \omega < 1$, there exists $\omega' < 1$ s.t. for every game $\mathcal{G}$ with $\omega(\mathcal{G}) = \omega$, and for every k, $\omega(\mathcal{G}^k) \leq (\omega')^k$.

Consequence

Theorem: $\forall\, \epsilon > 0$, $\exists\, l < \infty$ s.t.

$$\mathsf{NP} = \mathsf{MIP}_\epsilon[2, O(\log n), l].$$

Furthermore, the MIP verifier's verdict is given by a function $\sigma : \mathcal{A}_1 \to \mathcal{A}_2$, and it accepts iff $\sigma(a_1) = \sigma(a_2)$.

(Think of verifier's questions to prover one as asking: "Which of the satisfying assignments is your response?")

Towards Optimal Inner Verifiers

Goals of inner verifier

- Given oracle A_1 "committing" to a_1, A_2 "committing" to a_2, verify they are indeed committing to some string a_1, a_2 and that the strings satisfy $\sigma(a_1) = a_2$.

- Design elements:
  - Which code?
  - What to verify?
  - What is a sufficient commitment?
  - Analysis?

Long Code

Since we are talking of strings of constant length, no need to hold back on the code. Use the longest possible code!

Defn: The long code of an l-bit string a is the 2^{2^l}-bit string $L(a) = \langle f(a) \rangle_f$, where f ranges over all functions from l-bits to one bit.

Note: A table of the form $A[f]$ is a long code if $\exists\ a$ s.t. $A[f] = f(a)$ for all f.

Will assume that all tables are "folded": I.e., satisfy $A[1-f] = 1 - A[f]$. (Can simulate this on our own anyway). (Saves one bit of querying.)

The verifier

(The moment of truth!)

- Given: function σ, and oracles A_1, A_2.

- Random choices:
  - Pick $f, g : \{0,1\}^l \to \{0,1\}$ at random.
  - Noise function $\eta : \{0,1\}^l \to \{0,1\}$ randomly with each coordinate picked independently to be 1 w.p. ϵ and 0 w.p. $1-\epsilon$.

- The test: $A_2[f] \oplus A_1[g] \stackrel{?}{=} A_1[g']$
  where $g'(x) = g(x) \oplus f(\pi(x)) \oplus \eta(x)$.

Intuition

- Testing $A[h] \oplus A[g] \stackrel{?}{=} A[h \oplus g]$: If table passes test w.h.p. then it is close to the XOR of several long codes.
  I.e., $\exists\ x_1, \ldots, x_k$ s.t. A close to $\tilde{A}$
  where $\tilde{A}[f] = \oplus_i f(x_i)$.
- Folding: Forces $k \neq 0$.
- Error function η: $A[h] \oplus A[g] \stackrel{?}{=} A[h \oplus g \oplus \eta]$: Acceptance probability $\to \frac{1}{2}$ as $k \to \infty$.
- Above on A_2 & $A_1[f] \oplus A_2[g] \stackrel{?}{=} A_2[f \circ \sigma \oplus g]$: Checks consistency between A_1 and A_2.

- Actual protocol: Trick; Saves two bits!
- Actual analysis: Real analysis (Fourier coefficients, inner products, Cauchy-Schwartz etc. etc.)

Concluding

Have seen the path to:
$\text{NP} = \text{PCP}_{1-\epsilon,\frac{1}{2}}[O(\log n), 3] \quad \forall \epsilon > 0$
[Hastad 1997]

Main inclusions:

- Role of arithmetization.
- Recursive constructions of proofs.
- Nice problem statements.

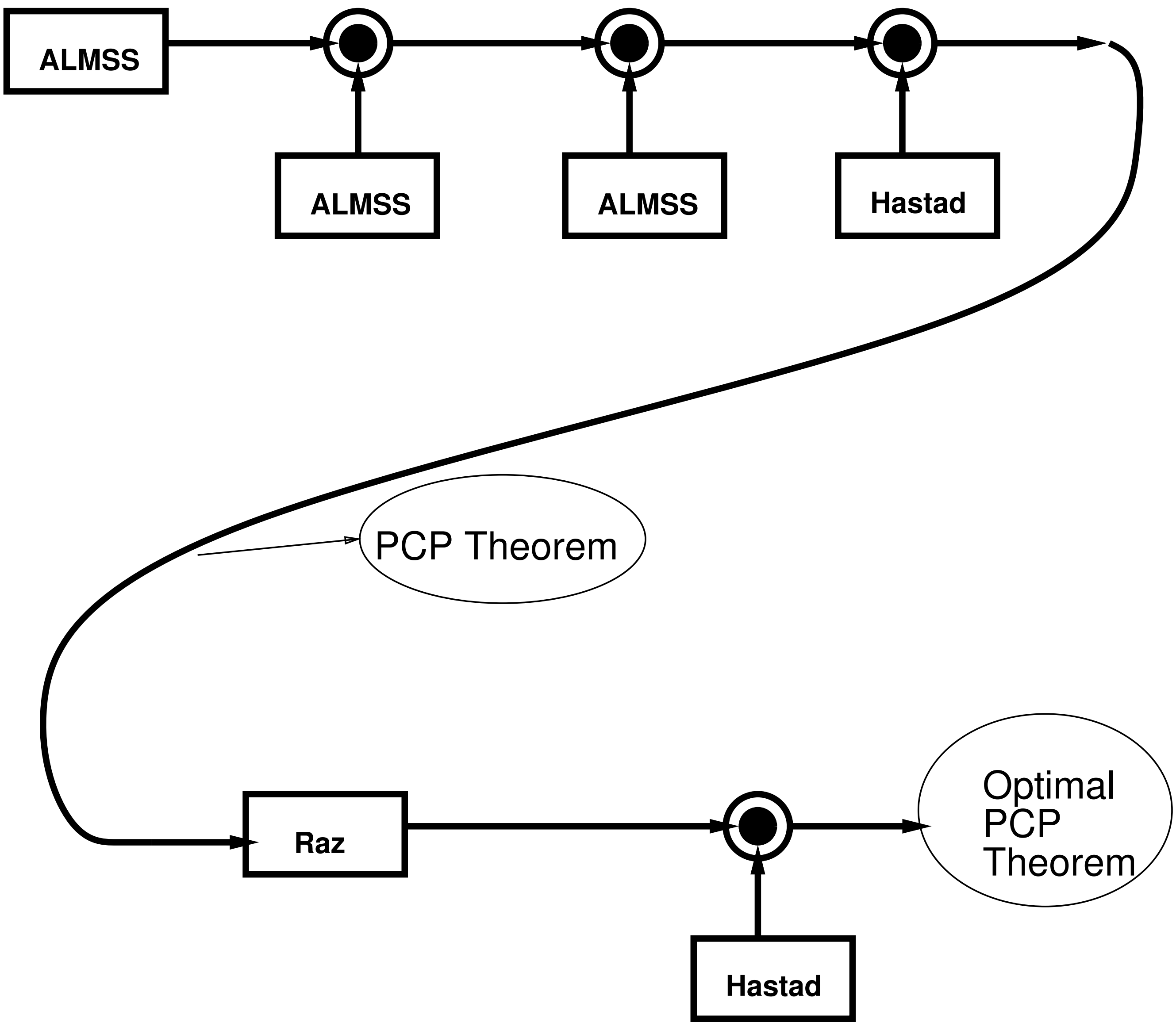
Main omissions:

- Low-degree testing
- Parallel Repetition
- Analysis of Optimal Inner Verifier

Optimal PCP: Ingredients

- Composition: [Arora+Safra] $\odot$
- ALMSS-IV: (Uses BFL(S), AS, RS etc.)
  MIP with $p = O(1)$, $r = \log$, $a = \text{poly}\log$.
- Hastad-IV: $p = 3, r = 2^n, a = 1$.
- Raz-Amplifier:
  $\text{MIP}_{1,1-\epsilon}[2, r, a] \longrightarrow \text{MIP}_{1,(1-\epsilon')^k}[2, kr, ka]$

Optimal PCP: Roadmap



Concluding thoughts

- Is all this really necessary?
- Simpler proofs?
- Even if we allow non-uniformity?
- At least the associated error-correcting codes?