Locally Testable Codes

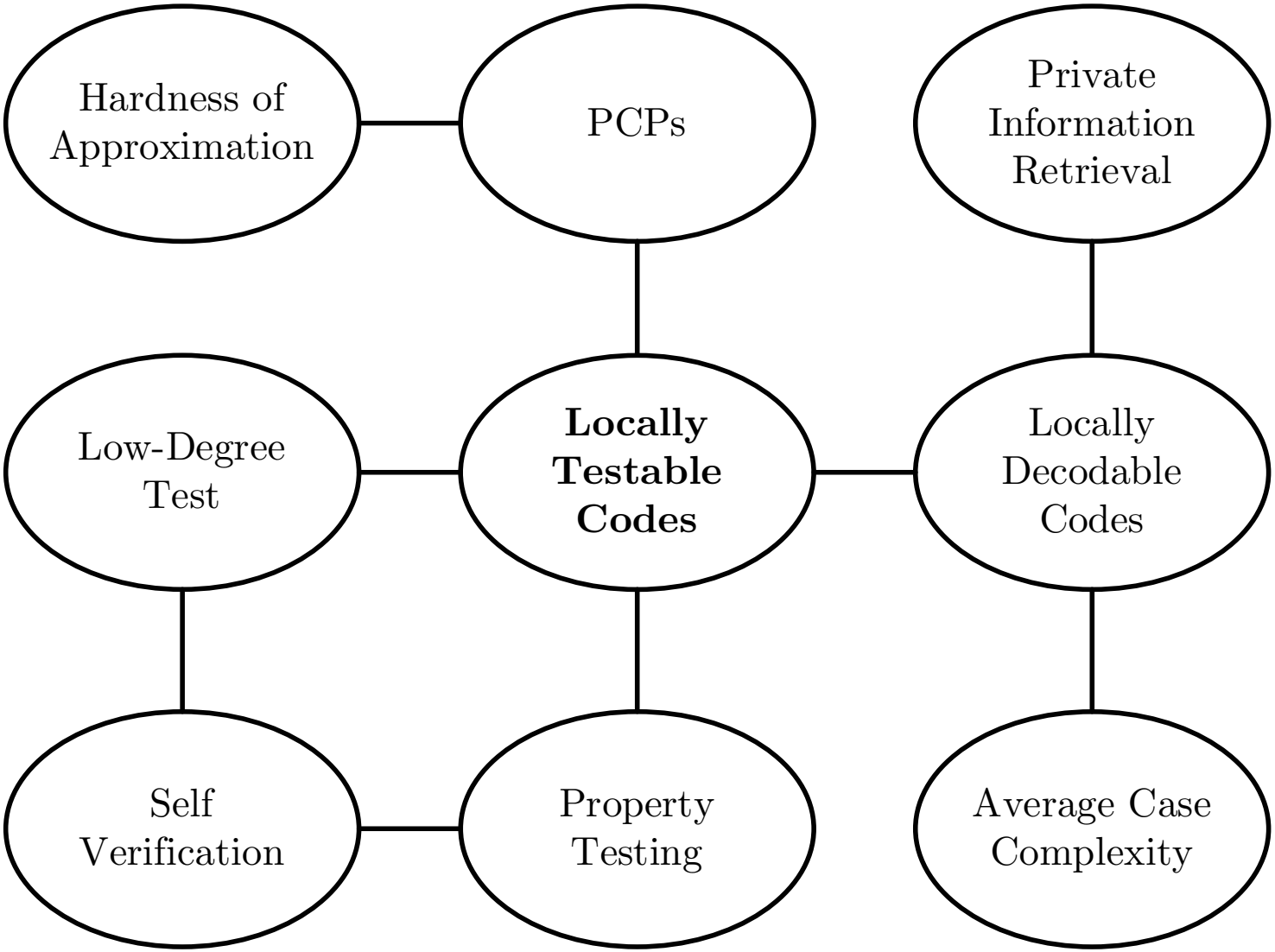
Mahdi Cheraghchi

mahdi.cheraghchi@epfl.ch

École Polytechnique Fédérale de Lausanne

July 05, 2005

Related Research Areas



Error Correcting Codes

- Encoding: $\mathcal{C} : \Sigma^k \rightarrow \Sigma^n$
- Message length: k
- Block length: $n(k)$
- Rate: $R(k) = \frac{k}{n(k)}$
- Alphabet: $\Sigma(k)$

Error Correcting Codes

- Encoding: $\mathcal{C} : \Sigma^k \rightarrow \Sigma^n$
- Message length: k
- Block length: $n(k)$
- Rate: $R(k) = \frac{k}{n(k)}$
- Alphabet: $\Sigma(k)$
- Absolute Hamming distance: $\Delta(\vec{x}, \vec{y}),\ \Delta(\vec{x}, \mathcal{C})$
- Distance: $\boldsymbol{\delta}(\vec{x}, \vec{y}),\ \boldsymbol{\delta}(\vec{x}, \mathcal{C})$
- Absolute minimum distance: $\min\{\Delta(\vec{w}, \mathcal{C})\}$
- Minimum distance: $\min\{\boldsymbol{\delta}(\vec{w}, \mathcal{C})\}$
- δ-close: $\boldsymbol{\delta}(\vec{x}, \vec{y}) \leq \delta,\ \boldsymbol{\delta}(\vec{w}, \mathcal{C}) \leq \delta$
- δ-far: $\boldsymbol{\delta}(\vec{x}, \vec{y}) \geq \delta,\ \boldsymbol{\delta}(\vec{w}, \mathcal{C}) \geq \delta$

Probabilistic Oracle Machine

- Standard Turing machine
- Receives an *explicit* (normal) input.
- Complexity measures are functions of explicit input length.
- Receives a sequence of iid and uniform random bits as an extra input.
- One (or more) extra random-access and read-only *oracle* tape(s)
- Can *query* an oracle at a particular position.
- Oracles can be thought of as *black box* functions.
- *Query complexity* $q(n)$: The maximum number of probes into the oracles, over all possible inputs and random choices.
- *Randomness complexity* $r(n)$: The maximum number of random bits used, over all possible inputs of certain length.

Non-Adaptive Probabilistic Oracle Machine

- Each query is independent of the outcome of all previous queries.
- Based on the explicit input and random bits, computes:
  1. A sequence of query locations $I = (i_1, \ldots, i_q)$,
  2. A boolean circuit D: $\{0,1\}^q \to \{0,1\}$.
- Queries the oracle π and accepts iff $D(\pi|_I) = 1$.
- *Decision complexity:* Maximum size of D on inputs of a certain length.
- We will focus on non-adaptive probabilistic oracle machines.

Polynomial Low-Degree Testing

A Software Verification Problem

A program is given as a black-box that computes some function $f(\mathbf{x})$. *Is* $f(\mathbf{x})$ *a polynomial of degree* d*?*

- *Formal setting: Given a function* f *as an oracle, and inputs* $d \in \mathbb{N}$ *and* $\delta \in (0, 1)$, *efficiently distinguish between the case that*
  1. f *is a degree* d *polynomial,*
  2. f *is* δ*-far from degree* d *polynomials.*

Polynomial Low-Degree Testing

A Software Verification Problem

A program is given as a black-box that computes some function $f(\mathbf{x})$. Is $f(\mathbf{x})$ a polynomial of degree d?

- *Formal setting: Given a function f as an oracle, and inputs $d \in \mathbb{N}$ and $\delta \in (0,1)$, efficiently distinguish between the case that*
  1. *f is a degree d polynomial,*
  2. *f is δ-far from degree d polynomials.*

Self-Correcting Software

A program is given as a black-box. We know that the function $f(\mathbf{x})$ it computes is close to some polynomial $g(\mathbf{x})$ of degree d. Efficiently compute $g(\mathbf{x})$.

Linearity Test (BLR)

- Low-degree test for degree one:

$$f(x_1, \ldots, x_n) = \alpha_1 x_1 + \cdots + \alpha_n x_n.$$

- Characterization: Over $\mathbb{F}_2$, f is linear iff

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n : f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y}).$$

- Linearity Testing Algorithm:
  1. Pick $\mathbf{x}, \mathbf{y} \in_R \mathbb{F}_2^n$.
  2. Accept if and only if $f(\mathbf{x}) + f(\mathbf{y}) = f(\mathbf{x} + \mathbf{y})$.
- Requires only three queries.

Linearity Test (BLR)

Theorem (Simplified)

BLR-Test is complete and sound:

- *If f is linear it accepts with probability* 1.
- *If for some $\delta \leq 0.2$, the probability that the test accepts is more than $1 - \delta$ then f is 2δ-close to linear functions.*

Linearity Test (BLR)

Theorem (Simplified)

BLR-Test is complete and sound:

- *If f is linear it accepts with probability* 1.
- *If for some $\delta \leq 0.2$, the probability that the test accepts is more than $1-\delta$ then f is 2δ-close to linear functions.*

- Linearity Self-Correction Algorithm:
  1. Given $\mathbf{x}$, pick $\mathbf{y} \in_R \mathbb{F}_2^n$.
  2. Output $f(\mathbf{x}+\mathbf{y}) - f(\mathbf{y})$.
- Requires only two queries.
- Has perfect completeness.
- Soundness: If f is δ-close to a linear function g, it outputs $g(\mathbf{x})$ with probability at least $1-2\delta$.

Univariate Low-Degree Test for Arbitrary Degree

- Characterization: f is a univariate polynomial of degree d over a field of size at least $d+2$ iff the evaluations of f on any subset of $d+2$ points can be interpolated by a degree d polynomial.
- Univariate Test:
  1. Pick distinct points $x_0, \ldots, x_{d+1} \in_R \mathbb{F}$.
  2. Accept iff f a degree d polynomial on $x_0, \ldots, x_{d+1}$.
- The test has perfect completeness. Moreover, if f is δ-far from degree d polynomials, the test rejects with probability $\Omega(\delta)$.
- Requires $d+2$ queries.
- Problems:
  1. Does not easily extend to multivariate case,
  2. Is not well suited for application in program testing.

Multivariate Low-Degree Test for Arbitrary Degree

- Characterization: A function $f : \mathbb{Z}_p \to \mathbb{Z}_p^n$ is a polynomial of degree d iff

$$\forall \mathbf{x}, \vec{h} \in \mathbb{Z}_p^n, \quad \sum_{i=0}^{d+1} \alpha_i f(\mathbf{x} + i \cdot \vec{h}) = 0,$$
$$\alpha_i \stackrel{\text{def}}{=} (-1)^{(i+1)} \binom{d+1}{i}.$$

- Evenly-Spaced-Points Test: Repeat $O(d^2 \log(\frac{1}{\beta}))$ times,
  1. Pick $\mathbf{x}, \vec{h} \in_{\mathsf{R}} \mathbb{Z}_p^n$.
  2. Reject if $\sum_{i=0}^{d+1} \alpha_i f(\mathbf{x} + i \cdot \vec{h}) \neq 0$.
- Perfect completeness.
- Soundness: If f is not $O(\frac{1}{d^2})$-close to a degree d polynomial, it is rejected with probability at least $1 - \beta$.
- Can be implemented using addition and subtraction only.

Locally Testable Codes (LTC)

- A *codeword tester* with query complexity q, proximity parameter δ and soundness error ϵ for a code $\mathcal{C}$ (q-local δ-tester)
  1. Is a (non-adaptive) probabilistic oracle machine.
  2. Runs in polynomial time.
  3. Has query complexity at most q.
  4. Receives an *alleged* codeword as oracle.
  5. **Completeness**: Always accepts codewords.
  6. **Soundness**: Rejects oracles that are δ-far from the code with probability at least ϵ.
  7. Can behave arbitrarily otherwise.
- A code is (q, δ)-locally testable if there exists a q-local δ-tester for it.
- By default, δ and ϵ are absolute constants in $(0, 1)$.
- *Detection probability:* $1 - \epsilon$.

Locally Testable Codes, Variations

- Stronger requirement: $\epsilon = \Omega(\delta)$.
- More flexibility: Allowing two-sided error and/or adaptivity.
- Tolerance: Extending the completeness margin by adding another proximity parameter.
- Robustness: Non-codewords must be *far from being accepted* with high probability.
  - Assume $(I, D) = T^{\vec{w};\vec{r}}$.
  - $\rho^T(\vec{w}, \vec{r}) \stackrel{\text{def}}{=} \boldsymbol{\delta}(\vec{w}, \{\vec{x} \mid D(\vec{x}|_I) = 1\})$.
  - $\rho^T(\vec{w}) \stackrel{\text{def}}{=} \mathbb{E}_{\vec{r}}[\rho^T(\vec{w}, \vec{r})]$.
  - A c-robust tester has perfect completeness and $\forall \vec{w}, \rho^T(\vec{w}) \geq \boldsymbol{\delta}(\vec{w}, \mathcal{C})/c$.
  - A code is c-robust if it has a c-robust tester.
  - c-robust codes are locally testable with $c \cdot q$ queries. (for q being the query complexity of the c-robust tester).

Locally Testable Codes, Parameters

- We aim to optimize:
  1. The alphabet size (ideal case: $|\Sigma| = 2$),
  2. Query complexity (ideal case: $q = O(1)$),
  3. Rate (ideal case: $R = O(1)$),
  4. Randomness complexity (ideal case: $r = \lg n + O(1)$).
- Optimization of the two latter parameters coincide, as $n \leq q2^r$.
- We focus on constant distance codes, even though it's not a necessity.

Locally Testable Codes, Parameters

- We aim to optimize:
  1. The alphabet size (ideal case: $|\Sigma| = 2$),
  2. Query complexity (ideal case: $q = O(1)$),
  3. Rate (ideal case: $R = O(1)$),
  4. Randomness complexity (ideal case: $r = \lg n + O(1)$).
- Optimization of the two latter parameters coincide, as $n \leq q2^r$.
- We focus on constant distance codes, even though it's not a necessity.
- Can we optimize all parameters simultaneously?!
- Ultimate goal: Asymptotically optimal binary LTCs.
- State of the art: Binary LTCs with $O(1)$ queries and inverse poly-log rate. (Dinur 2005)

Locally Testable Codes

- Limitations:
  - A 2-query LTC with minimum distance δ can have at most $|\Sigma|^{3/\delta}$ codewords.
  - With high probability, a random (c, d)-regular LDPC code needs $\Omega(n)$ queries, even for adaptive testers with two-sided error.
- Steps towards asymptotically optimal codes:
  1. *Constant* nearly linear length: For every $\epsilon > 0$, $n = O(k^{1+\epsilon})$.
  2. *Tight* nearly linear length: $n = O(k^{1+f(k)})$, where $f(k) = o(1)$. Examples: $f(k) = 1/\log\log n$, $f(k) = 1/\log^c n$, $f(k) = \exp((\log\log\log n)^c)/\log n$.
  3. *Polylog* nearly linear length: $n = k \cdot \text{poly}(\log k)$. Tight nearly linear with $f(k) = \text{poly}(\log\log n)/\log n$.
  4. Linear length(?!!)

Hadamard Code

- A message $\vec{u} \in \mathbb{F}_2^k$ is considered as the linear function

$$\ell_{\vec{u}}(x_1, \ldots, x_k) \stackrel{\text{def}}{=} \langle \vec{u}, \vec{x} \rangle = u_1 x_1 + \cdots + u_k x_k.$$

- The encoding of $\vec{u}$ is the evaluation vector of ℓ_u.
- Equivalently, the encoding of $\vec{u}$ is the evaluation of all k-variate linear functions at $\vec{u}$. $(\ell_{\vec{u}}(\vec{x}) = \ell_{\vec{x}}(\vec{u}))$
- Block length: 2^k.

Schwartz-Zippel Lemma

A nonzero m-variate polynomial of degree ℓ can be zero on at most an ℓ/q fraction of points in $\mathbb{F}_q^m$.

- Minimum distance: $1 - \frac{\ell}{q} = 1 - \frac{1}{2} = \frac{1}{2}$.
- Locally testable by linearity testing.

Reed-Muller Codes

- Fix a field $\mathbb{F}_q$ and parameters m and ℓ.
- Message is considered as an m-variate polynomial of total degree ℓ.
- Codeword is the evaluation vector of the message polynomial.
- $k = \binom{m+\ell}{m}$.
- $n = q^m$.
- Minimum distance: $1 - \ell/q$.
- Locally testable by Evenly-Spaced-Points test.

Polynomial-Line Testing

- A *proof-assisted* low-degree test for multivariate polynomials.
- Assumes the existence of a proper *auxiliary* proof.
- Line through $\mathbf{x}$ with slope $\vec{h}$: $\ell(t) = \mathbf{x} + t \cdot \vec{h}$.
- Characterization:
  - A finite field $\mathbb{F}_q$ with characteristic p such that $q - \frac{q}{p} \geq d + 1$.
  - $f : \mathbb{F}^m \longrightarrow \mathbb{F}$ is a polynomial of degree at most d iff for all $\mathbf{x}, \vec{h} \in \mathbb{F}^m$, f restricted to the line $\ell_{\mathbf{x},\vec{h}}$ is a univariate polynomial of degree d.
- Robustness: If restriction of f is *close* to being degree d on *most* lines, f is itself close to being a degree d polynomial.
- Line representation of a polynomial $f \in \mathbb{F}^{(d)}[X^m]$,

$$\text{lines}_f : \mathbb{F}^{2m} \longrightarrow \mathbb{F}^{d+1},$$

maps a line ℓ to the restriction of f to ℓ.

Polynomial-Line Testing

- Polynomial-Line Test: Assumes auxiliary oracle $B : \mathbb{F}^{2m} \longrightarrow \mathbb{F}^{d+1}$
  1. Pick $\mathbf{x}, \vec{h} \in_R \mathbb{F}^m$ and $t \in_R \mathbb{F}$.
  2. Accept iff $f(\mathbf{x} + t \cdot \vec{h}) = B(\mathbf{x}, \vec{h})(t)$.
- Needs only two queries, but over a large alphabet.
- Completeness: If f is a degree d polynomial, it is accepted with the oracle lines_f.
- Soundness: For every constant $\epsilon > 0$ and $\delta \leq \frac{1}{8} - \epsilon$, if $|\mathbb{F}| = \Omega(d)$ and f is δ-far from being degree d, then it is rejected with probability at least $\delta/2$, no matter what the auxiliary oracle is.

Polynomial-Line Correction

- Polynomial-Line Corrector: Is given a point $\mathbf{x}$ and assumes an auxiliary oracle $B : \mathbb{F}^{2m} \longrightarrow \mathbb{F}^{d+1}$
  1. Pick $\vec{h} \in_R \mathbb{F}^m$ and $t \in_R \mathbb{F} \setminus \{0\}$.
  2. Let $q = B(\mathbf{x}, \vec{h})$.
  3. **if** $q(t) \neq f(\mathbf{x} + t \cdot \vec{h})$ **then** reject **else** output q(0).
- Has perfect completeness.
- Soundness: If f is δ-close to a degree d polynomial g, then the probability of not returning $g(\mathbf{x})$ is at most $2\sqrt{\delta} + \frac{d}{|\mathbb{F}|-1}$.

Polynomial-Line Codes

- Is based on the Polynomial-Line test.
- Message: The coefficients of an m-variate polynomial of degree $d \geq m$ over a field of size $\Omega(d)$.
- Codeword: For a message f, lines$_f$
- Test:
  1. Pick $\mathbf{r} \in_R \mathbb{F}^m$.
  2. Pick two lines ℓ, ℓ' in $\mathbb{F}^m$ going through $\mathbf{r}$, uniformly and independently at random.
  3. Accept iff w_ℓ and $w_{\ell'}$ agree at the intersection $\mathbf{r}$.
- Is complete and sound.
- Alphabet: $\mathbb{F}^{d+1}$.
- Codeword length: $n = \mathbb{F}^{2m}$.
- Message length: $k = \binom{m+d}{m}/(d+1) \overset{m \ll d}{\approx} (d/m)^m$.
- For $n = \text{poly}(k)$, we need $|\mathbb{F}| = \text{poly}(d)$ and $m \ll d$.

Polynomial-Line Codes

- Problem: Bad rate!
- Not linear but $\mathbb{F}$-linear.
- Randomly truncate the code to $n = O(|\mathbb{F}|^m \log |\mathbb{F}|)$ lines.
- Instantiations:
  1. Tight nearly linear: $d = m^m$, $\mathbb{F} = O(d)$.
     - $k \approx m^{m^2-2m}$
     - $n = O(|\mathbb{F}|^m \log |\mathbb{F}|) = m^{m^2+o(m)}$
     - $n \approx \exp(\sqrt{\log k}) \cdot k$
     - $\log |\Sigma| = \log |\mathbb{F}|^{d+1} \approx d \log d \approx \exp(\sqrt{\log k})$
  2. Constant nearly linear: $d = m^c$ for $c > 1$.
     - $k \approx m^{(c-1)m}$
     - $n \approx m^{cm} = k^{c/(c-1)}$
     - $\log |\Sigma| \approx d \log d \approx \log^c k$
- For $1 - o(1)$ truncation choices, soundness is preserved.

Polynomial-Line Codes

- Reduce the alphabet to $\mathbb{F}$.
- Concatenate with an inner code.
- The inner code maps multilinear forms of total degree d' to their evaluations.
- Assume $h^{d'} = d + 1$, and $d'h$ variables $x_i^{(j)}$ for $j \in [d']$ and $i \in [h]$.
- Message: Coefficients of

$$f(\vec{x}) \stackrel{\text{def}}{=} \sum_{i_1,\ldots,i_{d'} \in [h]} q_{i_1,\ldots,i_{d'}} \cdot x_{i_1}^{(1)} \cdot x_{i_2}^{(2)} \cdots x_{i_{d'}}^{(d')}$$

- Codeword: Evaluation vector of $f(\vec{x})$.
- $n' = |\mathbb{F}|^{d'h} = \exp(d' \cdot (k')^{\frac{1}{d'}} \cdot \log |\mathbb{F}|)$.
- Inner tester: d' linearity tests and one total degree test.

Polynomial-Line Codes

- Testing concatenated code: *Emulate* the outer test.
- Each query only needs the value of a univariate polynomial q of degree d at some point $t \in \mathbb{F}$.
- Write $q(t)$ as

$$q(t) = \sum_{i_1,\ldots,i_{d'} \in [h]} q_{i_1,\ldots,i_{d'}} t^{(i_1-1)+(i_2-1)h+\cdots+(i_{d'}-1)h^{d'-1}}.$$

- So we need the inner-code entry corresponding to

$$\langle t^0, \ldots, t^{h-1}, t^{0\cdot h}, \ldots, t^{(h-1)\cdot h}, \quad \ldots\ldots \quad, t^{0\cdot h^{d'-1}}, \ldots, t^{(h-1)\cdot h^{d'-1}} \rangle.$$

- Use an interpolating self-corrector to retrieve above $\Rightarrow d'+1$ queries.
- Total amount of queries: $2(d'+1) = O(d')$.

Polynomial-Line Codes

- Instantiation:
  - $d := m^c$ and $d' := 2c$, for constant $c > 1$.
  - $n \approx k^{c/(c-1)}$
  - $k' = d + 1$
  - $n' \approx \exp(d^{1/d'}) \approx \exp((\log k)^{c/d'}) = \exp(\sqrt{\log k}) = k^{o(1)}$
  - $nn' \approx (kk')^{c/(c-1)} \Rightarrow$ *constant* nearly linear.
  - Alphabet size: $O(d) \approx \log^c k$
- Obtaining a binary code: Concatenate with Hadamard code.
- Observe: Above tester checks a constant number of linear constraints $\Sigma_i \alpha_i a_i = 0$ over $\mathbb{F}$. (for query outcomes a_i)
- Pick a random binary sequence r and check $\langle r, \sum_i \alpha_i a_i \rangle \equiv 0 \mod 2$.
- Write it as $\sum_i \langle r, \alpha_i a_i \rangle \equiv 0 \mod 2$.
- Hence we need to check if a linear combinations of the bits of a_i is zero.
- Obtain the combined value in *one shot* from the Hadamard encoding of a_i.

Polynomial-Line Codes

Final parameters for the binary code $\mathbb{F}_2^{kk'k''}$ to $\mathbb{F}_2^{nn'n''}$:

- $nn' \approx (kk')^{c/(c-1)}$
- $n'' = 2^{k''} = |\mathbb{F}| = \text{poly}(\log k) = k^{o(1)}$
- $nn'n'' \approx (kk'k'')^{c/(c-1)}$
- *constant* nearly linear length binary LTC.

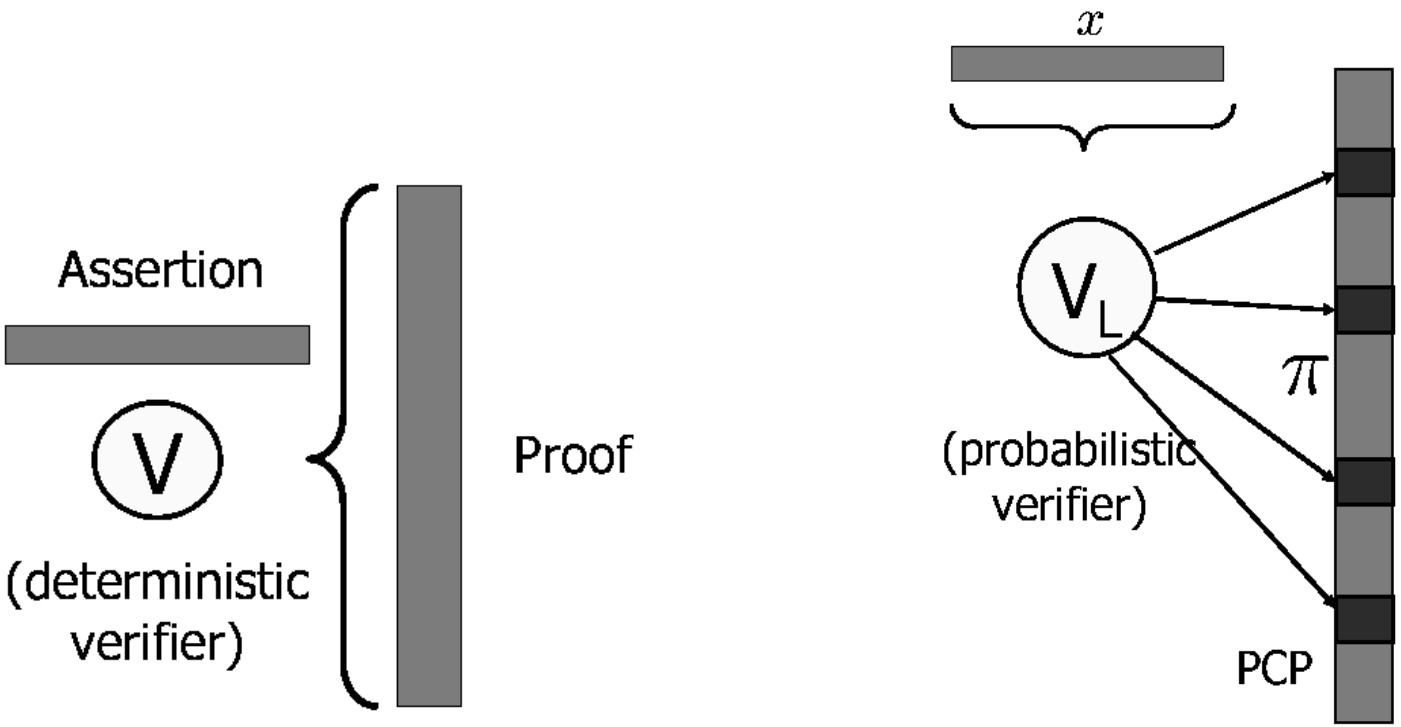
Tensor Product Codes

- Fully combinatorial!
- An $n_2 \times n_1$ matrix is a codeword of $\mathcal{C}_1 \otimes \mathcal{C}_2$ iff every row is a codeword of $\mathcal{C}_1$ and every column is a codeword of $\mathcal{C}_2$.
- Product tester: Pick a row (column) at random and verify if is corresponds to a codeword of $\mathcal{C}_1$ ($\mathcal{C}_2$).
- Generalize to m-dimensions by considering $\mathcal{C}^m$.
- m-Product tester is 2^{16}-robust for $\mathcal{C}^m$ for $(\frac{d-1}{n})^m$.
- $\Rightarrow$ Tensor product gives LTCs.

Theorem

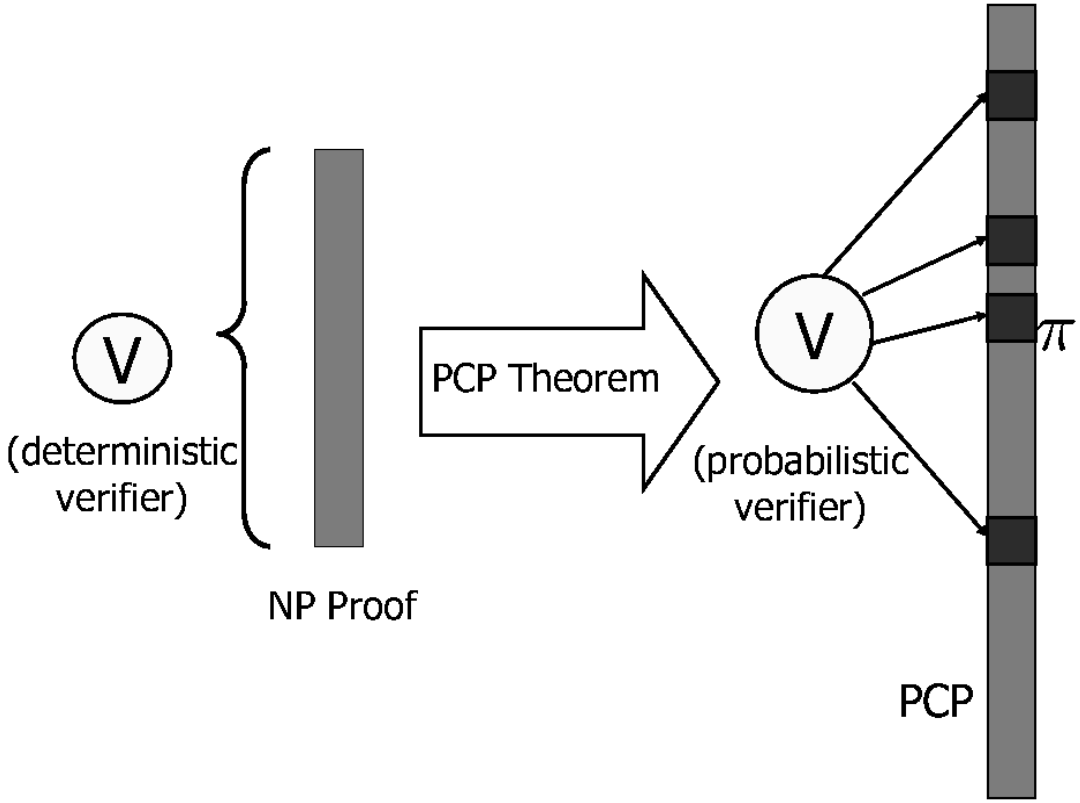
For any family $\{\mathcal{C}_i\}$ with polynomial block length, let t_i be a sequence of integers s.t. $m_i := 2^{t_i}$ satisfies $d_i/n_i \geq 1 - \frac{1}{7m_i}$. Then $\{\mathcal{C}_i^{m_i}\}$ is locally testable with polylog query complexity and polynomial length.

Probabilistically Checkable Proofs (PCP)



- A probabilistic polynomial time verifier.
- **Completeness:** If $x \in L$, then $\exists \pi$: $\Pr[V^{\pi}(x) = 1] = 1$.
- **Soundness:** If $x \notin L$, then $\forall \pi, \Pr[V^{\pi}(x) = 1] \leq \frac{1}{2}$.

Probabilistically Checkable Proofs (PCP)



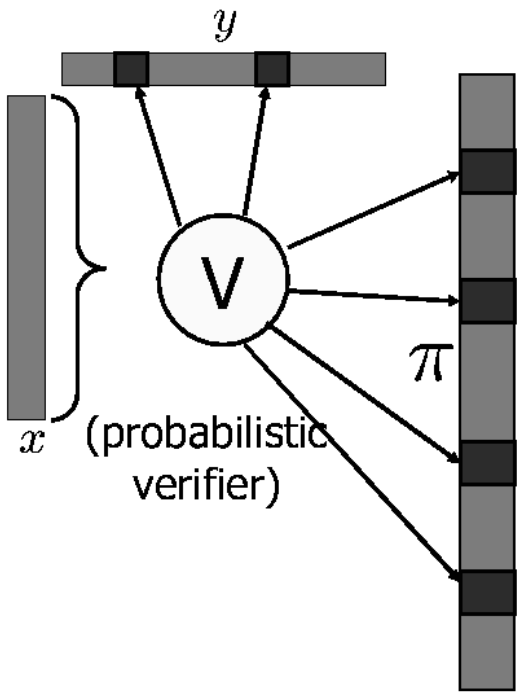
The PCP Theorem

NP $=$ PCP$(O(\log n), O(1))$.

PCP vs. LTC

- LTCs can be considered as combinatorial counterparts of PCPs.
- Combinatorial objects are more intuitive.
- Idea: Obtain PCPs from LTCs.
  - Goldreich and Sudan 2002 (weaker result),
  - Ben-Sasson and Sudan 2005 (polylog rate and query).
- Stronger LTC results are only obtained from PCPs!
- Challenge: Obtain short PCPs with low (and constant) query complexity.
- Problems with obtaining LTCs from PCPs:
  - Unique encoding,
  - Loss of linearity in compositions,
  - Padding problems, dealing with auxiliary variables, etc.
- A *generic solution:* PCPs of Proximity.

PCPs of Proximity



- A pair language $L = \{(x, y)\}$, x is given as explicit input, y as oracle.
- **Completeness:** If $(x, y) \in L$, then $\exists \pi$: $\Pr[V^{y \circ \pi}(x) = 1] = 1$.
- **Soundness:** If y is $\delta(|x|)$-far from $L(x)$, then $\forall \pi, \Pr[V^{y \circ \pi}(x) = 1] \leq s(|x|)$.
- $\delta(\cdot)$: Proximity parameter, $s(\cdot)$: soundness error.

PCPs of Proximity

PCPP vs. PCP:

- $\textsc{CktVal} \equiv \{(x, C(x)) \mid C(x) = 1\}$, P-complete.
- $\textsc{CktSat} \equiv \{C \mid C \text{ is a satisfiable circuit}\}$, NP-complete.
- Assignment tester: PCPP verifier for $\textsc{CktVal}$.
- An assignment tester is also a PCP verifier for $\textsc{CktSat}$.

PCPP vs. LTC:

- Using an auxiliary code $\mathcal{C}_0$ with good parameters and a PCPP verifier for membership in $\mathcal{C}_0$, an LTC can be built:
  - Codewords: $(\mathcal{C}_0(x)^t, \pi(x))$ s.t. only $1/d$-fraction goes to $\pi(x)$.
  - Distance: $\delta(\mathcal{C}_0) - 1/d$.
  - Tester: PCPP + a constant number of consistency checks.
  - Proximity: Similar to PCPP.
  - Block length: PCPP length times d.
- Corollary: Assignment testers imply LTCs with (almost) similar parameters.

Best Known PCPPs

- Ben-Sasson *et al.*:
  1. Query complexity $O(1/\epsilon)$, randomness $\lg n + O(\log^{\epsilon} n) \Rightarrow$ proof length $n \cdot \exp(\log^{\epsilon} n)$.
  2. Query complexity $o(\log \log n)$, randomness $\lg n + O((\log \log n)^2 / \log \log \log n) \Rightarrow$ proof length with quasi-polylogarithmic blowup.
- Dinur: Query complexity $O(1)$, randomness $\lg n + O(\log \log n) \Rightarrow$ poly-log rate.
- They also correspond to best known binary LTCs.

Locally Decodable Codes (a bird-eye view)

- Assume: the sequence is δ-close to a codeword.
- Idea: Retrieve one message symbol using a BPP oracle decoder with low query complexity.
- Basic constructions: Based on polynomial self-correctors, e.g., Reed-Muller and Hadamard codes.
- Problems:
  - Quasi-exponential block length in best known construction.
  - Lower bounds: Binary 2-query: $n = 2^{\Omega(k)}$, Arbitrary alphabet 2-query: $n \geq 2^{\frac{\epsilon\delta k}{4}-1}$, Binary q-query: $\tilde{\Omega}(k^{q/(q-2)})$, ...
- Theoretical applications: private information retrieval, average case complexity, etc.
- Nearly linear length PCPPs imply nearly linear length LDCs in a *relaxed* sense.

Discussion and Selected References

- S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.
- E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *STOC'04*.
- I. Dinur and O. Reingold. Assignment testers: Toward a combinatorial proof of the PCP-theorem. In *Proceedings of FOCS'04*.
- K. Friedl and M. Sudan. Some improvements to total degree tests. In *Proceedings of ISTCS'95*, pages 190–198.
- O. Goldreich and M. Sudan. Locally testable codes and PCPs of almost-linear length. In *Proceedings of FOCS'02*, pages 13–22.
- R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comp.*, 25(2):252–271, 1996.
- M. Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*. PhD thesis, U.C. Berkeley, 1992.