Multiplicity Codes: Locality with High Efficiency

Madhu Sudan
Microsoft Research

Based on **[Kopparty, Saraf, Yekhanin (STOC 2011)]**

Error-Correcting Codes

- Used to store data over (noisy) storage media/communicate data over (noisy) channels.
- Code (over alphabet Σ).
  - $E: \Sigma^k \to \Sigma^n; C = \text{Image}(E);$
  - Terminology:
    - $\text{Domain}(E)$ = messages; C = codewords.
  - Rate$(C) = \frac{k}{n}$.
  - Distance: For $x, y \in \Sigma^n$, $\delta(x,y) = \frac{1}{n} | \{ i \mid x_i \neq y_i \} |$

$$\delta(C) = \min_{\{u \neq v\}} \{\delta(E(u), E(v)\}$$

- Codes of interest: $R(C), \delta(C) > 0$.

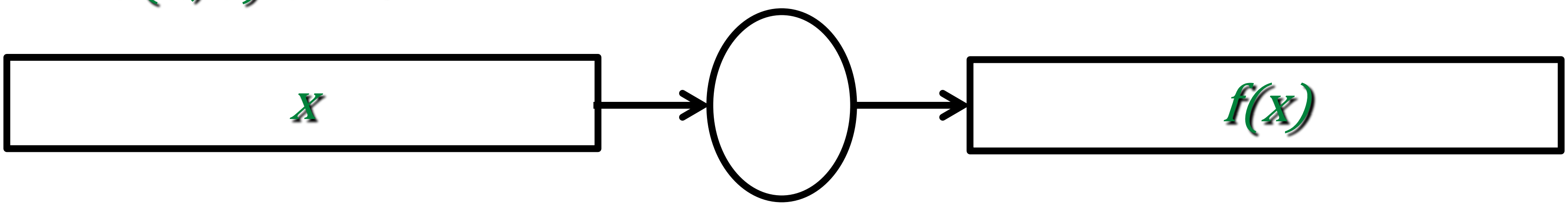
Algorithmic Problems in Coding Theory

(Fix Code C and encoding function E)

- Encoding:
  - Given $m \in \Sigma^k$, compute $E(m)$.
- Error Detection/Testing:
  - Given $w \in \Sigma^n$, determine if $w \in C$.
  - Variations: Determine $\delta(w, C) \triangleq \min_{x \in C} \{\delta(w, x)\}$ (approximately).
- Error Correction:
  - Given $w \in \Sigma^n$ s.t. $\exists\, x \in C$ s.t. $\delta(w, x) \leq \delta$; compute x.

Sublinear Algorithmics

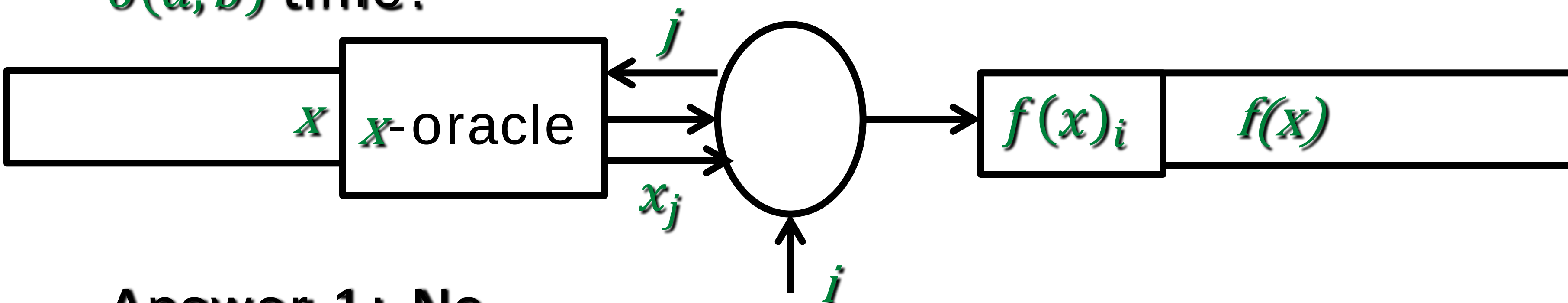
- For $f:\{0,1\}^a \to \{0,1\}^b$, can $f(x)$ be computed in $o(a,b)$ time?



- Answer 1: No

Sublinear Algorithmics

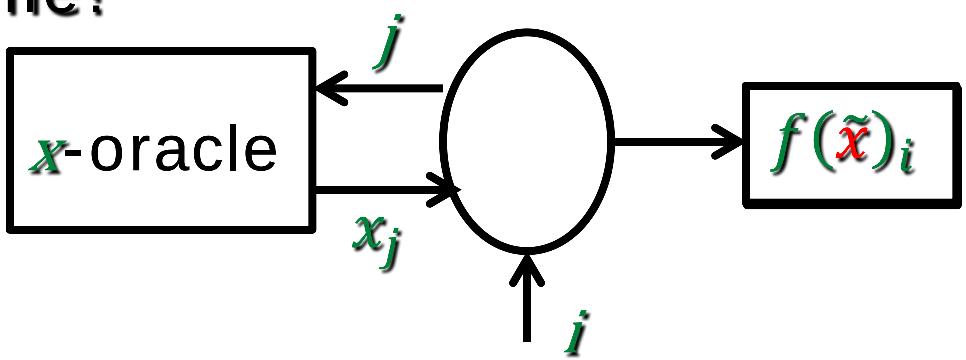
- For $f:\{0,1\}^a \to \{0,1\}^b$, can $f(x)$ be computed in $o(a,b)$ time?



- Answer 1: No
- Answer 2: Yes, provided:
  - Output represented implicitly
  - Input given as oracle

Sublinear Algorithmics

- For $f:\{0,1\}^a \rightarrow \{0,1\}^b$, can $f(x)$ be computed in $o(a,b)$ time?



- Answer 2: Yes, provided:
  - Output represented implicitly
  - Input given as oracle
  - Correctness guaranteed on approx. to input.

Sub-linear time algorithms

- Initiated in late eighties in context of
  - Program checking [BlumKannan,BlumLubyRubinfeld]
  - Interactive Proofs/PCPs [BabaiFortnowLund]
- Now successful in many more contexts
  - Property testing/Graph-theoretic algorithms
  - Sorting/Searching
  - Statistics/Entropy computations
  - (High-dim.) Computational geometry
- Many initial results are coding-theoretic!

Sub-linear time algorithms & Coding

- Encoding: Not reasonable to expect in sub-linear time.
- Testing? Decoding? – Can be done in sublinear time.
  - In fact many initial results do so!
- Codes that admit efficient …
  - … testing: Locally Testable Codes (LTCs)
  - … decoding: Locally Decodable Codes (LDCs).

Rest of this talk

- Definitions of LDCs
- Some background/Basic Construction
- Recent constructions of LDCs.
  - [Kopparty-Saraf-Yekhanin '11]

Definition

Locally Decodable Code (LDC)

- Code C with encoder E is (ℓ,δ)-LDC if there exists a (sublinear-time) decoding algorithm D on
  - Input: $i \in [k]$ and Oracle for $w: [n] \to \Sigma$, s.t. $\exists m \in \Sigma^k$ s.t. $\delta(w, E(m)) \leq \delta$,
  - Outputs: m_i w.p. at least $2/3$
  - Locality: makes only ℓ queries to w.
- History:
  - Some implied LDCs from 1950s [Reed,Muller].
  - Construction + Implied definitions [Babai,Fortnow,Lund,Szegedy'90].
  - Explicit definitions [S.,Trevisan,Vadhan; Katz-Trevisan]

Motivations

Motivations to study LTCs

- Intimately related to concept of Probabilistically Checkable Proofs (PCPs):
  - Format for writing mathematical proofs that can be checked by few local probes.
    - (Key ingredient in many hardness of approximation results.)
  - Current state of art:
    - State of the art PCP/LTCs [BenSassonS,Dinur]
      - Parameters: k bits to $k \cdot \mathrm{poly} \log k$ bits.
      - Locality: O(1) queries.

Motivations to study LDCs

- **Hard-core predicates:** Hard Boolean functions from general hard functions.
- **Hardness amplification:** Functions that are hard to compute on random inputs, from worst-case hard functions.
- **Private Information Retrieval:** Distributed information storage method which allows user to query information privately.

Why the negativity?

- Why are local codes leading only to negative results? (inapprox, hard predicates, hard-on-average functions, privacy schemes …)
- What about the obvious positive possibility: on storage devices etc.?
  - Rate is too weak:
    - best known with sublinear decoding
      - Rate .5 for locality $\sqrt{n}$
      - Rate $\epsilon^{\frac{1}{\epsilon}}$ for locality n^{ϵ}.
    - Provable lower bounds: $n^{\left\{1+\frac{1}{\ell}\right\}}$ [KatzTrevisan]
    - Practical settings: Rate .8, .9 etc.

Basic Constructions

Self-correctible codes

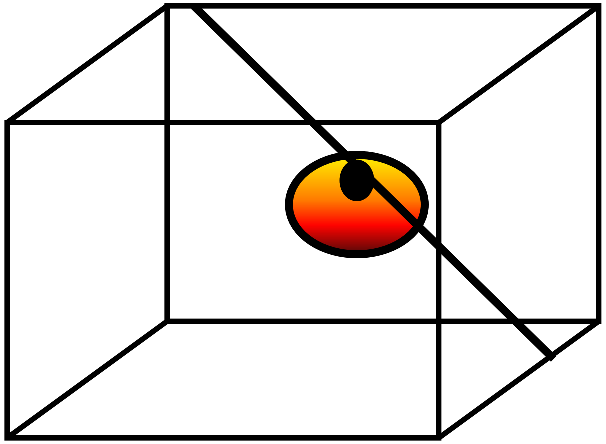
- Will ask for (slightly?) stronger object:
  - Every letter of codeword locally recoverable.
    - (as opposed to message)

- Why?
  - Simpler concept (depends only on code, not encoding function).
  - Implies existence of encoding function that leads to LDC.

Codes from Multivariate Polynomials.

- Parameters: F_q, m, d
- Message space: m-variate polynomials of degree at most d over F_q
- Encoding: Evaluations over $(F_q)^m$
- Resulting code parameters: $\binom{n}{k}$
  - $n = q^m$
  - $k = \binom{m+d}{d}$
  - Distance $\geq 1 - \frac{d}{q}$ (Can also use $d > q$, with care)

Local decoding/Self-correction



$(F_q)^m$

- Codeword = Function P on cube.
- Rec'd word = Function f on cube = P with errors
- Correction problem: Recover codeword at point
- Self-correction alg:
  - Look at f on line
  - Recover P on line (classical decoding)
- Locality = $q = n^{1/m}$

Sample Parameters

- Best locality:
  - $d = 1,\ m = k,\ q\ =\ 2$:
    - $n = 2^k$; 2-locally decodable code ("Hadamard code") for ¼-fraction errors.
- Weakest (sublinear) locality:
  - $m = 2,\ \ q = \frac{d}{1-2\delta}$
  - $k = \binom{d+2}{2} \approx \frac{d^2}{2}$; $n = q^2 \approx \frac{2k}{(1-2\delta)^2}$;
  - Locality $= \sqrt{n}$, correcting δ fraction errors
- In general: locality n^{ϵ} at rate $\epsilon^{\frac{1}{\epsilon}}$ with $m = 1/\epsilon$

The Rate < ½ barrier

- To get $\Omega(1)$ distance, need $d < q$.
- To get non-trivial locality: $m \geq 2$.
- Implies $k < \binom{q+2}{2} \approx q^2/2$, and $n = q\text{^}2$.
- Rate at most ½.

The new breakthrough

- Multiplicity Codes [KSY '11]
- Theorem:
  - For every $\alpha, \beta > 0$, $\exists\, \delta > 0$ s.t. $\forall\, n$
    there exist codes C_n with
    - $\text{Rate}(C_n) \geq 1 - \alpha$
    - C_n is n^β-locally-decodable from δ errors.
- Rate arbitrarily close to 1
  - (not expected – at least not by me).
- Locality arbitrarily small power of n.
- Even concrete parameters are interesting.

Multiplicity Codes

Basic Idea:

- Extend Multivariate Polynomial codes:
  - Encoding also includes evaluations of "partial derivatives".
  - Cons: Now encoding is even more redundant, so we lose rate?
  - Pros: But we can use higher degrees.
    - E.g.: Fraction of points where are all zero is at most $\deg(f)/2q$.
      - (f_x denotes "partial derivative" wrt x)

Hasse derivatives & Multiplicities

- For every $i = (i_1, \ldots, i_m)$, there exists a notion of i th partial derivative of $P(x_1, \ldots, x_m)$, denoted $P^{(i)}$
- Order of $i = (i_1, \ldots, i_m)$ is $\sum_j i_j$
- $\mathrm{Mult}\big(P, (a_1, \ldots, a_m)\big) \triangleq$ largest s s.t. all derivatives of P of order $< s$ vanish at $(a_1, \ldots, a_m)$
- Multiplicity Schwartz-Zippel Lemma:

$$\mathrm{E}_{a_1,\ldots,a_m}\left[\,\mathrm{Mult}\big(P, (a_1, \ldots, a_m)\big)\right] \leq \frac{\deg(P)}{q}$$

Multiplicity Codes Example-1

- Parameters: $m = 2,\ d = (1-2\delta)q,\ s = 2$.
- Alphabet = (F_q)^3
- Message = m-variate polynomials of degree d over F_q
- $\text{Encoding}(P) = \left\{\left(P(a,b), P_x(a,b), P_y(a,b)\right)\right\}_{(a,b)}$
- Code parameters: $n = q^2\,; k \approx \frac{1}{3}\cdot\frac{d^2}{2}\,;$
  - $\text{Rate}(C) \approx \frac{2}{3}$ (as $\delta \to 0$);
  - Locality = ? Hopefully: $O(q) = O(\sqrt{n})$.
  - If so, sublinear locality at rate $> \frac{1}{2}$!

Locality – I (no errors)

- Reconstructing $P(a,b)$ from $f = P$.
  - Idea: Still decode along lines.
  - Pick line ℓ thru $(a,b) : \ell = \{(\alpha t + a, \beta t + b)\}_t$.
  - Define $g(t) = P(\alpha t + a, \beta t + b)$.
  - $\deg(g) \leq d$;
  - have correct value of $g(t), \forall t \in F_q - \{0\}$.
  - Insufficient, since $d > q$.
  - $g'(t) \triangleq$ derivative of g wrt t can be obtained from P_x and P_y (specifically, $g' = \alpha P_x + \beta P_y$)
  - Now have enough info to interpolate $g(t)$ and so can get $g(0)$

Locality – I (with errors)

- Reconstructing $P(a,b)$ from $f \approx P$.
  - Idea: Still decode along lines.
  - Pick line ℓ thru $(a,b) : \ell = \{(\alpha t + a, \beta t + b)\}_t$.
  - Define $g(t) = P(\alpha t + a, \beta t + b)$.
  - $\deg(g) \leq d$;
  - have correct value of $g(t)$, for most $t \in F_q - \{0\}$.
  - Insufficient, since $d > q$.
  - $g'(t) \triangleq$ derivative of g wrt t can be obtained from P_x and P_y (specifically, $g' = \alpha P_x + \beta P_y$)
  - Now have enough info to decode $g(t)$ and so can get $g(0)$

Locality - II

- Not done! also need to recover $P_x(a,b)$ and $P_y(a,b)$.
- Idea 1: P_x is just another polynomial of degree d – can recover locally?
  - No! Don't have P_{xx}, P_{xy}, etc. which would be needed.
- Actual solution:
  - Using $\ell = (\alpha t + a, \beta t + b)_t$,
    can recover $\alpha P_x + \beta P_y$.
  - Pick another random line and get $\alpha_2 P_x + \beta_2 P_y$.
  - Can recover P_x and P_y from the above.
- Conclude: Decodable with $O(\sqrt{n})$ queries.

Improving Rate, Locality

- Increase # variables to reduce locality to $n^{\frac{1}{m}}$
- Next, increase multiplicities s (and degree) to get rate up to $1 - \alpha$!
- Naively, fraction of errors corrected $\rightarrow \Omega(\frac{\delta}{s^m})$, where $\delta = \frac{\alpha\beta}{8}$.
- Running time $O(s^m n^3)$.
- More sophisticated idea $\rightarrow \frac{3}{5} \cdot \delta$

$\boxtimes$

Derivatives?

- Classical derivatives no good over finite fields
  - 2^{nd} derivative of every poly. zero over F_{2^k}
- Hasse derivatives:
  - Univ. poly P: a root of mult. s
    - $(x-a)^s$ divides $P(x)$
      $\Leftrightarrow \quad x^s$ divides $P(x+a)$
      $\Leftrightarrow \quad P^{(i)}(a) = 0,\ \forall\, i < s$ where $P(x+z) = \sum_i P^{(i)}(x) \cdot z^i$
    - $P^{(i)}(x)$ is the Hasse derivative of P(x).
  - Multiv. Poly? Just extend above notationally!
    - $z = (z_1, \ldots, z_m), i = (i_1, \ldots, z_m), \quad z^i \triangleq \prod_j z_j^{i_j}$

Concluding thoughts

- Techniques:
  - Derivatives are not locally computable!
  - More multiplicities
  - More non-linear codes?
- Theory?
  - Can we prove these codes are locally testable?
  - Can we get PCPs with such parameters?
- Practice?
  - No more rate barrier to using locally decodable codes! When will we see these on USB sticks?

Thank You!