Time and Space Efficiency of Codes and Proofs

Joshua Cook’s Thesis Defense

What did I actually do?

More Efficient Verifiers for Interactive Proofs and PCPs

More Verifier Efficient Interactive Protocols For Bounded Space (FSTTCS 2022)

Efficient Interactive Proofs for Non-Deterministic Bounded Space (Random 2023) with Ron Rothblum

Tighter MA/1 Circuit Lower Bounds From Verifier Efficient PCPs for PSPACE (RANDOM 2023) with Dana Moshkovitz

Efficiency of Encoders and Decoders for Error Correcting Codes

Explicit Time and Space Efficient Encoders Exist Only With Random Access (CCC 2024) with Dana Moshkovitz

Time and Space Efficient Deterministic Decoders. 2024 with Dana Moshkovitz

Time and Space Efficient Deterministic List Decoding. 2025 with Dana Moshkovitz

Time-Space vs Cumulative Memory in the Streaming Model. (Manuscript) with John Kallaugher and Niels Kornerup

Focus for proposal

Focus for Defense

The Decoding Problem

What is an Error Correcting code?

Definition:

Function C: $\Sigma^k \rightarrow \Sigma^n$ such that for $x \neq y$, we have

$$Pr_i[C(x)_i \neq C(y)_i] \geq \boldsymbol{\delta}^*.$$

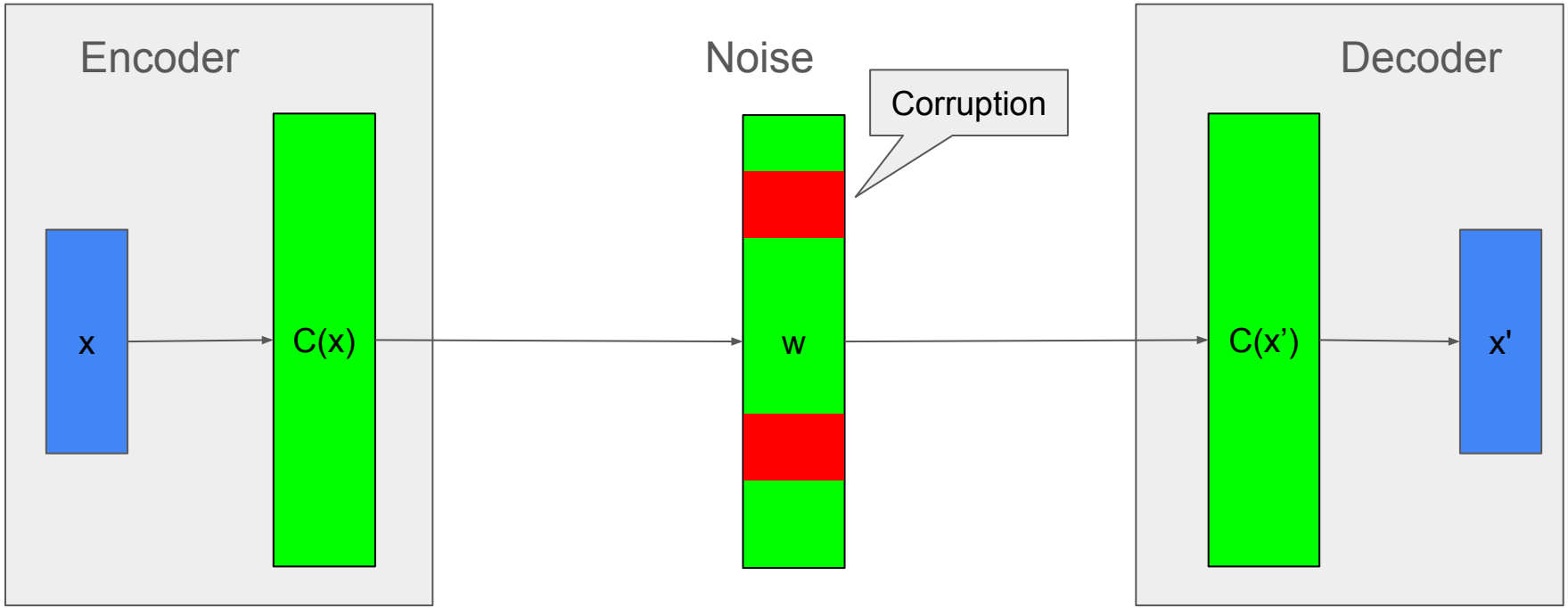
Relative distance $\boldsymbol{\delta}^*$, rate $r = k/n$. Good if $\boldsymbol{\delta}^*$, $r = \boldsymbol{\Omega}(1)$ and $|\Sigma| = O(1)$.

The correcting radius is $\boldsymbol{\delta} = \boldsymbol{\delta}^*/2$.

Many, many applications,

some of which are in the small space regime.

How We Use Error Correcting Codes

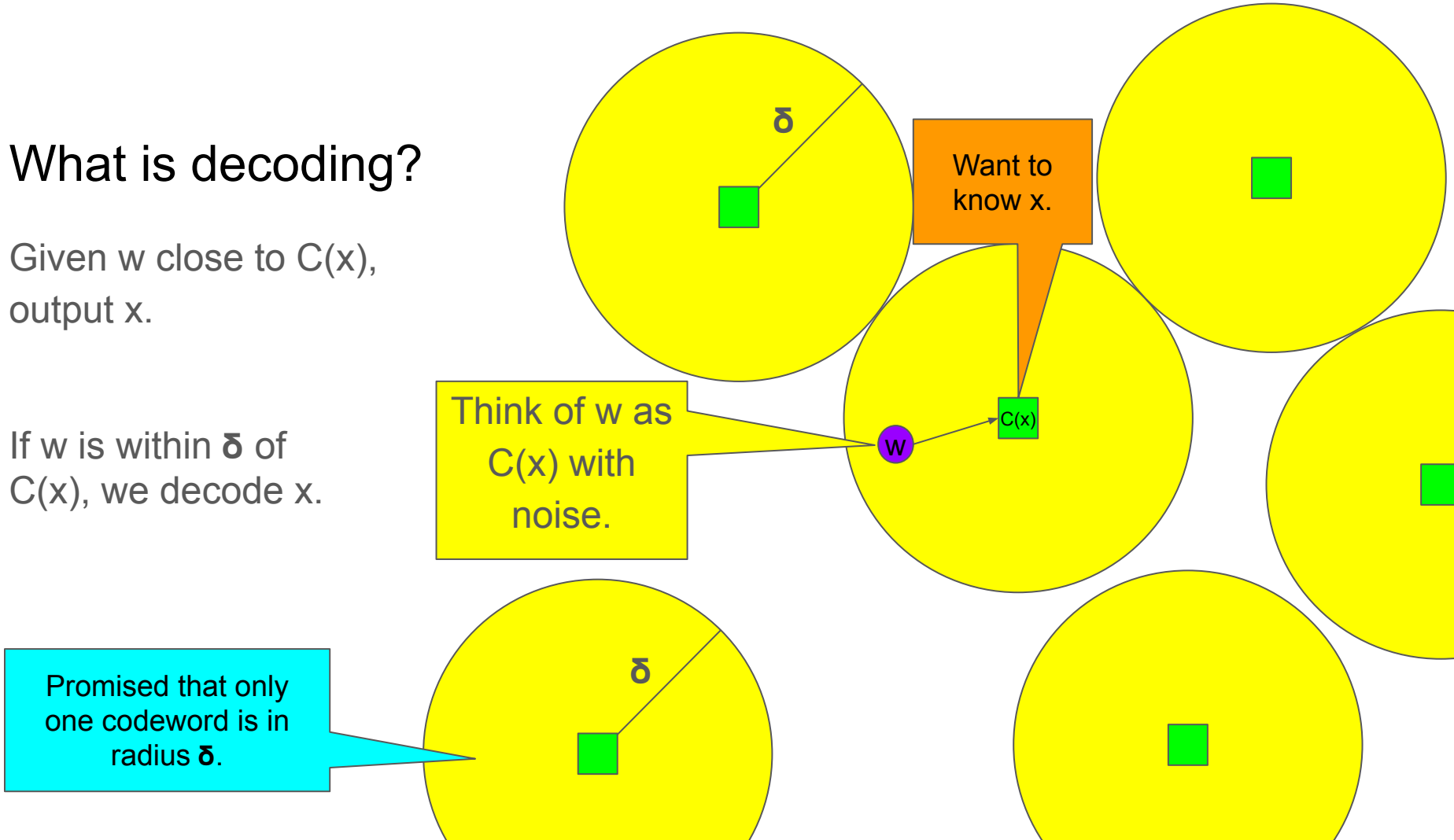


If corruption ≤ **δ**, then x' = x

What is decoding?

Given w close to C(x), output x.

If w is within **δ** of C(x), we decode x.



Examples of Error Correcting Codes

Reed-Solomon codes: codewords univariate low degree polynomials.

Example: for $a = (a_1,a_2,a_3)$, let $f_a(x) = a_1 + a_2x + a_3x^2$. Output truth table of f_a.

Reed-Muller codes: codewords are multivariate low degree polynomials.

Example, let $f_a(x,y) = a_1 + a_2x + a_3x^2 + a_4xy + a_5y + a_6y^2$. Output f_a.

Expander codes: codewords satisfy constraints given by an expander graph.

Many more: Turbo codes, polar codes, tensor codes, hamming code, etc.

Efficient Codes

Want efficient encoding and decoding.

Linear time?

✓ Spielman

Log depth, Linear sized Circuits?

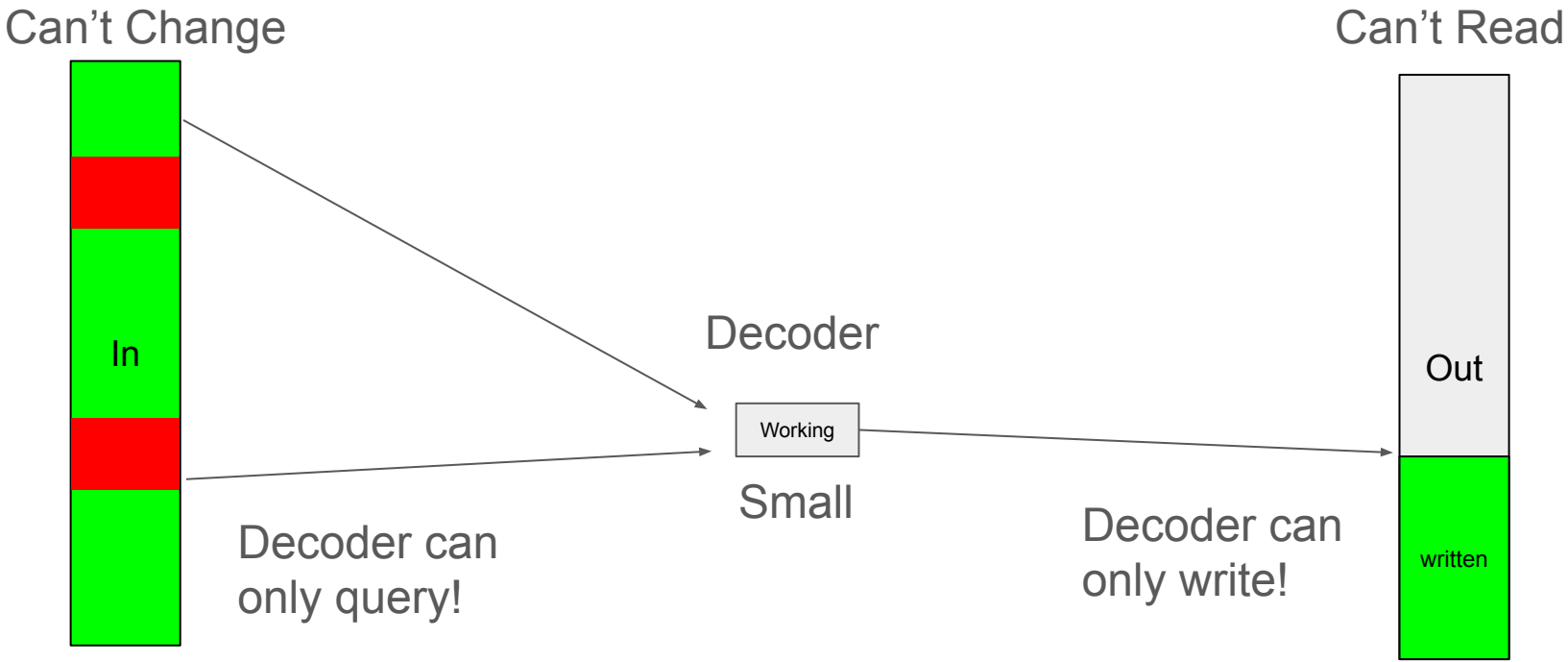
✓ Spielman

Linear time, log space?

Want explicit good codes with deterministic, uniform encoders and decoders.

What is a log space decoder?

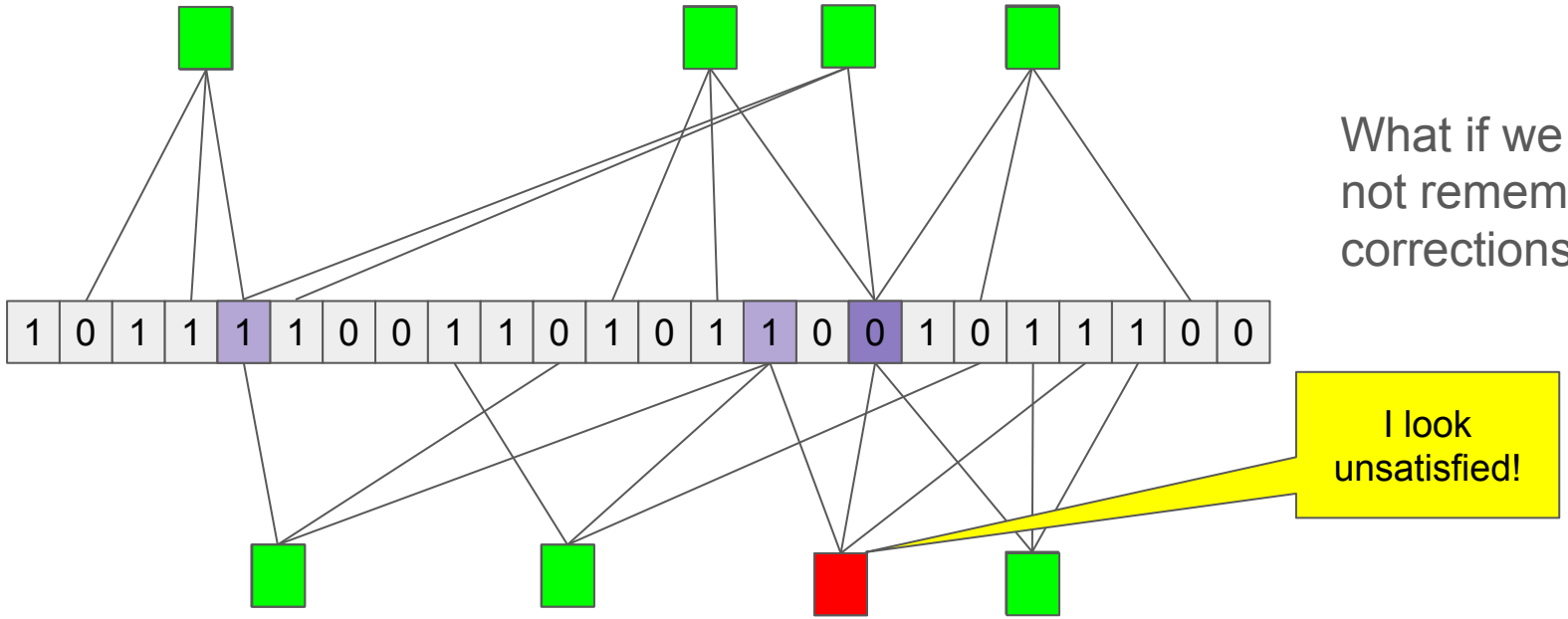
Input is read only. Output is write only. Have only O(log(n)) working space.



Why is Efficient Decoding Hard?

Decoding in Near Linear Time and Small Space

Standard approaches to linear time decoding often require storing partially corrected codeword in memory and making iterative corrections.



Local Decoding and Randomized Decoder

Can we correct a single symbol only looking at a small number of symbols?

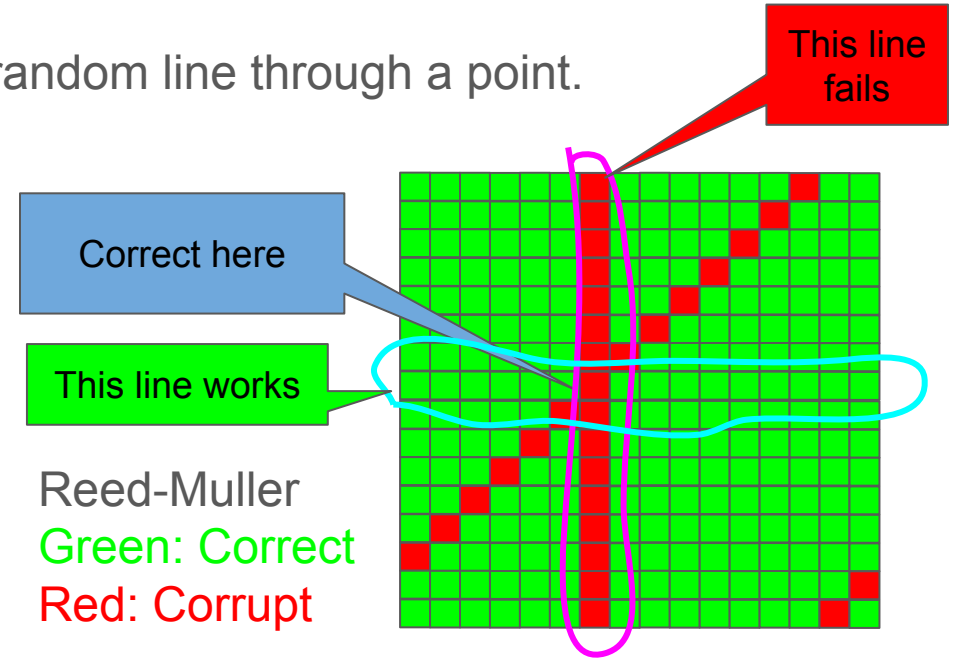
Randomly? Yes.

For Reed Muller, take a random line through a point.

If it rarely hits corruption, local correction succeeds.

Deterministically? No.

Always checks the same few symbols, adversary corrupts those.



Locally Correctable Codes (LCC)

Definition:

An LCC is a code $C: \Sigma^k \to \Sigma^n$ with a randomized algorithm D such that:

For any $w \in \Sigma^n$, $x \in \Sigma^k$ where $\Pr_i[C(x)_i \neq w_i] \leq \boldsymbol{\delta}'$ and any $i \in [n]$ we have

$$\Pr[D(w, i) \neq C(x)_i] \leq \tfrac{1}{3}$$

D is q query if it only makes q queries to w. $\boldsymbol{\delta}'$ is called the correcting radius.

Most codes we consider are systematic, so local correctors give decoders.

Randomized Decoder Continued

Reed-Muller codes are locally correctable!

By repeating O(log(n)) times, the error probability drops below 1/n.

So it is unlikely any symbol is decoded incorrectly.

This gives us a time and space efficient **randomized** decoder.

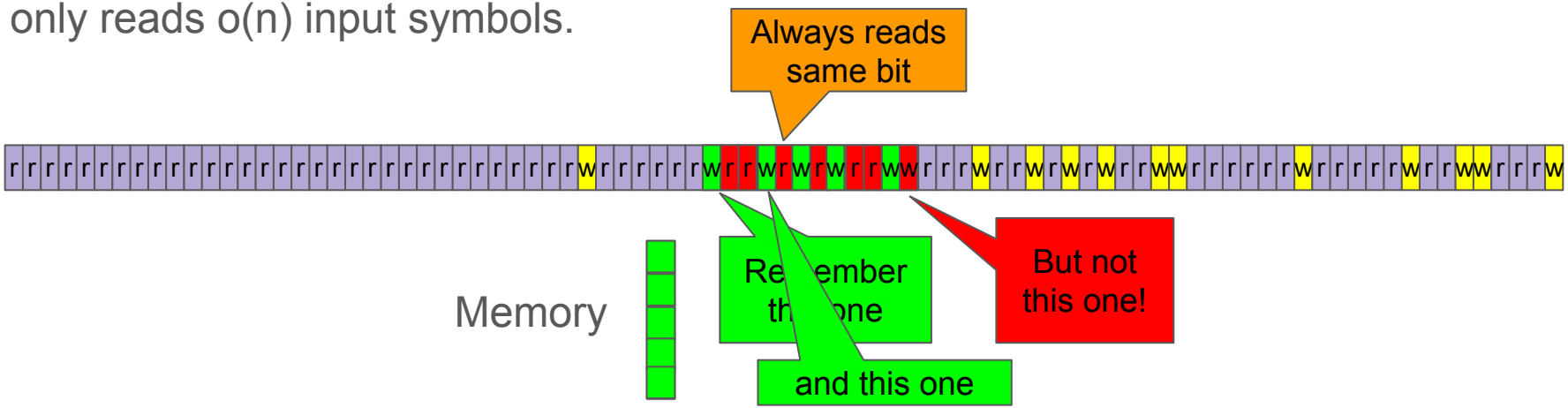
Local correction cannot be **deterministic** (and *always* correct)!

Non Adaptive, Deterministic Decoders Fail

Gronemeier proved that a non-adaptive decoder cannot do this.

Non-adaptive means where the decoder reads and when it writes are independent of the input.

Idea: for space S, wait for an interval where the decoder outputs S+1 symbols and only reads o(n) input symbols.



First Result: Efficient (Non-Uniform) Deterministic Decoders

Theorem:

Good, typical* q query LCCs have **non-uniform**, *deterministic* decoders running in space $O(q \log(n)^{1.5} / \sqrt{\log(q)})$ and time

$$O(n \log(n)\, n^{O(\sqrt{\log(q)} / \sqrt{\log(n)})}).$$

*(typical means systematic, non-adaptive, and with perfect completeness).

For $q = n^{o(1)}$: space is $n^{o(1)}$ and time $n^{1+o(1)}$.

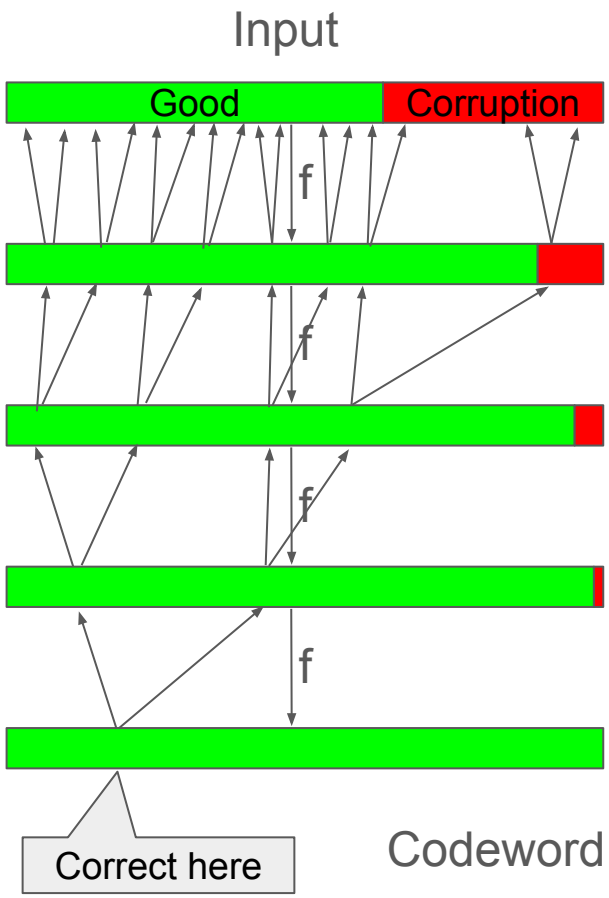
A prior result by Gronemeier proved non-adaptive decoders for good codes required time T and space S such that $ST = \Omega(n^2)$.

A flawed approach

Find a (single) q query f such that

$$\Pr_{i,j}[C(x)_i \neq f_j(w)_i] \leq \eta \Pr_i[C(x)_i \neq w_i]$$

f reduces the fraction of corruptions by η.



How good can a single, deterministic f be?

If f makes q queries, and we can corrupt **δ** fraction of symbols.

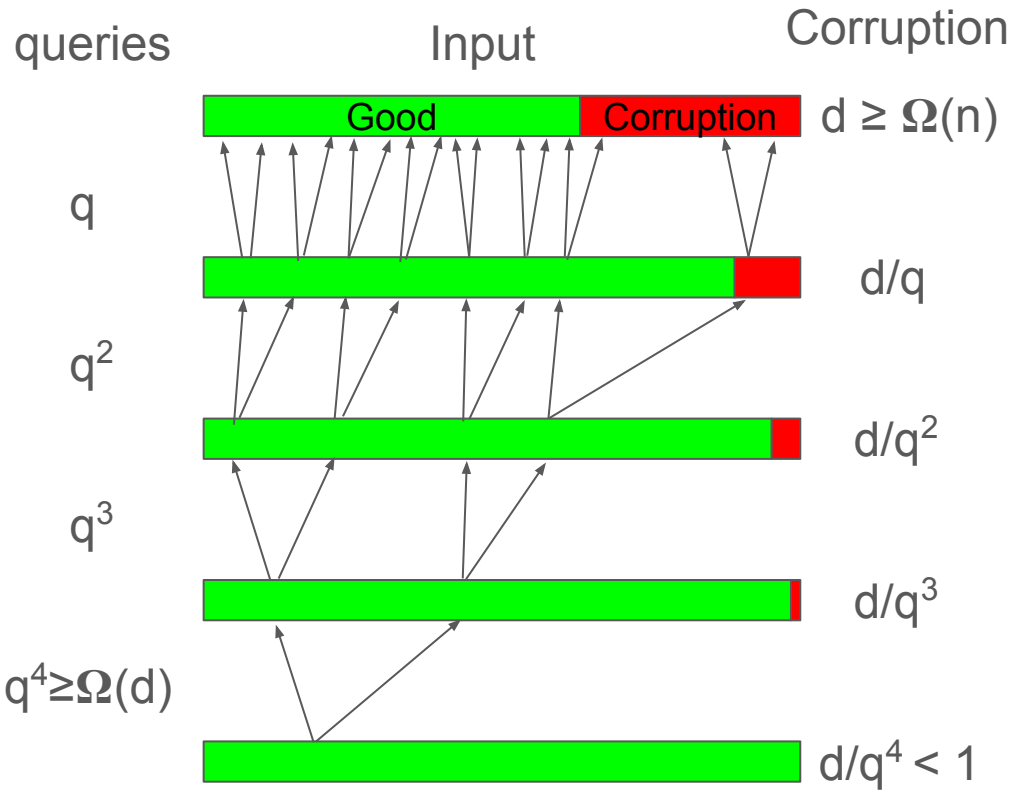
For deterministic f, only get η = 1/q

Why doesn’t it work?

Deterministic, q query f only reduces **δ** corruptions to **δ**/q.

To get zero errors, we would need **δ**n queries (per symbol)!

This gives a total time of **δ**n^2.



Our Efficient Decoder

Fix: More than one function

What about m different q query functions $f_1, \ldots, f_m$?

Now the O(**δ**/q) failures are distributed among m functions.

Less than O(**δ** / m) on average.

Don't have to pay for m in the recursion, so can make m ≫ q!

Definition:

$f_1, \ldots, f_m$ is a below **δ**, factor η improving set if for any w and C(x) with $\Pr_i[C(x)_i \neq w_i] \leq \boldsymbol{\delta}$ we have

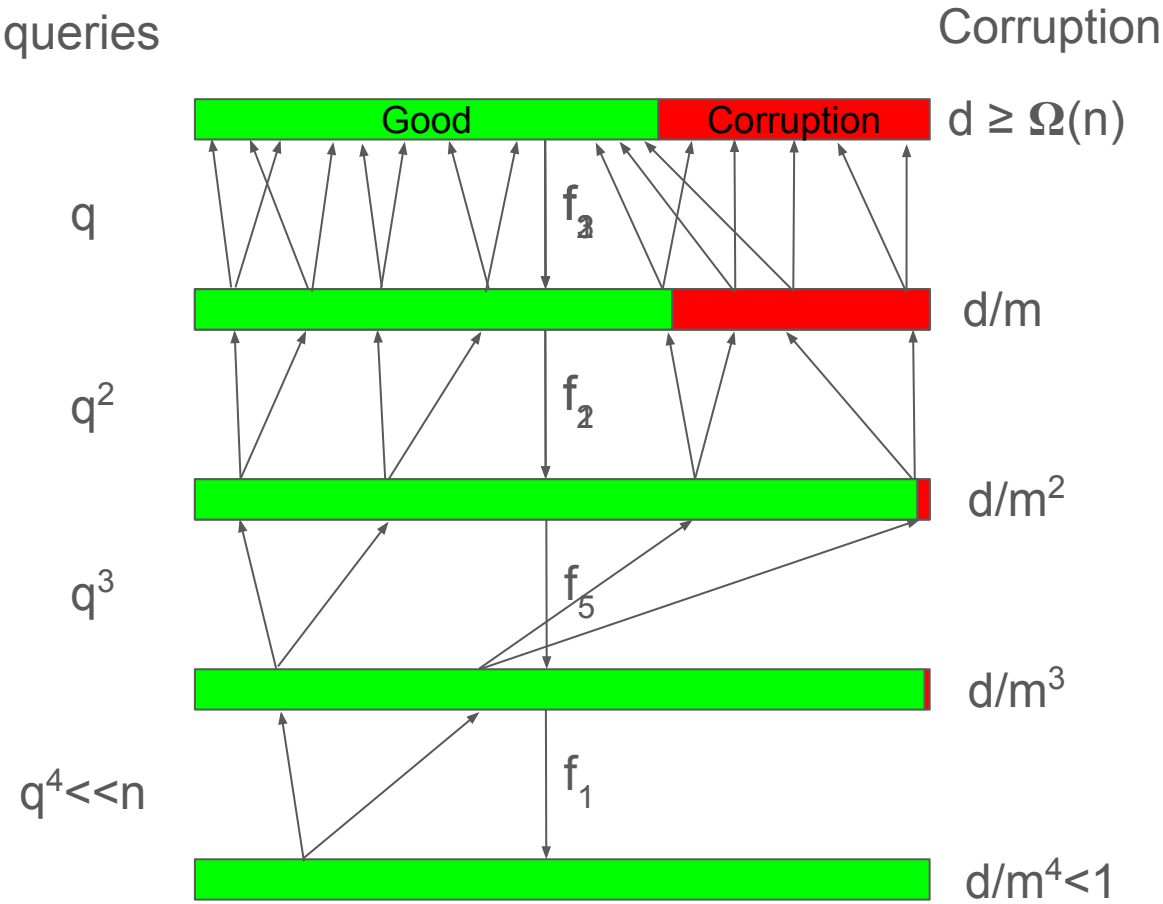
$$\Pr_{i,j}[C(x)_i \neq f_j(w)_i] \leq \eta \Pr_i[C(x)_i \neq w_i]$$

Example

Try all $f_1, .., f_m$.

Keep going until error is zero.

Still recursive, but fewer levels if q << m.



Selecting f_j from Improving Set

Ideally for every i compare $C(x)_i$ to $f_j(w)_i$ to see how good f_j is.

Don’t have access to C(x), but

In expectation a random $f_k(w)$ is close to C(x).

Choose the f_j such that $f_j(w)$ agrees with the most $f_k(w)$ at the most indexes.

Runtime of Algorithm

Selecting a function takes $O(q\, m^2\, n)$ queries to the level before it.

A query to level L takes q^L queries.

Final decoder only requires space q L and time

$$O(n\, m^2 q^L).$$

As long as $\eta \ll 1/q$, q^L will be small (if $\eta = q^{-a}$ then $q^L = n^{1/a}$).

Can We Find such an Improving Set?

Yes, (for typical LCC).

For any q query LCC and η, there is a q query improving set with size

$$m = O(\log(n)^2/\eta).$$

If $q = n^{o(1)}$, then setting η appropriately gives space $n^{o(1)}$ time:

$$n^{1+o(1)}.$$

Uniform Decoding for Reed-Muller

Second Result: Efficient **Uniform** Decoders

Theorem:

There is a good code with a **uniform**, deterministic decoder running in space $n^{o(1)}$ and time $n^{1+o(1)}$.

The code is based on Lifted Reed-Solomon codes.

Also applies to the special case of Reed-Muller codes.

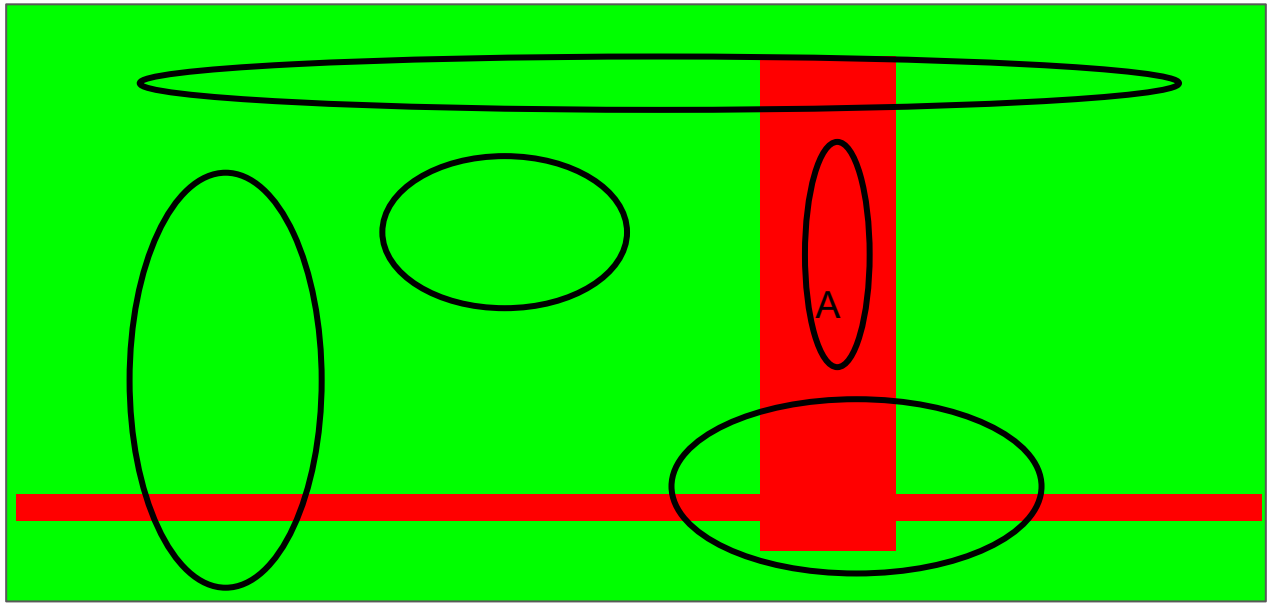
Samplers

Family of subsets of N, $\mathscr{S}$.

We say $\mathscr{S}$ is a sampler if:

For all sets A (let $\mu = {}^{|A|}/_{|N|}$).

The probability $S \in \mathscr{S}$ oversamples A is low.



Definition: We say $\mathscr{S} = (S_1, S_2, \ldots S_k)$ where each $S_i \subseteq N$ is a sampler for N if for some accuracy error $\varepsilon > 0$ and strong confidence error δ, for all $A \subseteq N$, and $\mu = |A|/|N|$ we have

$$\Pr_i[|S_i \cap A|/|S_i| \geq \mu + \varepsilon] \leq \delta\mu.$$

Curve Samplers

Need samplers with special structure to allow decoding.

- Lines (Line samplers)
- Subspaces (Space Samplers).
- Curves (Curve Samplers).

Prior curve samplers by Ta-Shma and Umans (and later by Guo) exist, but they:

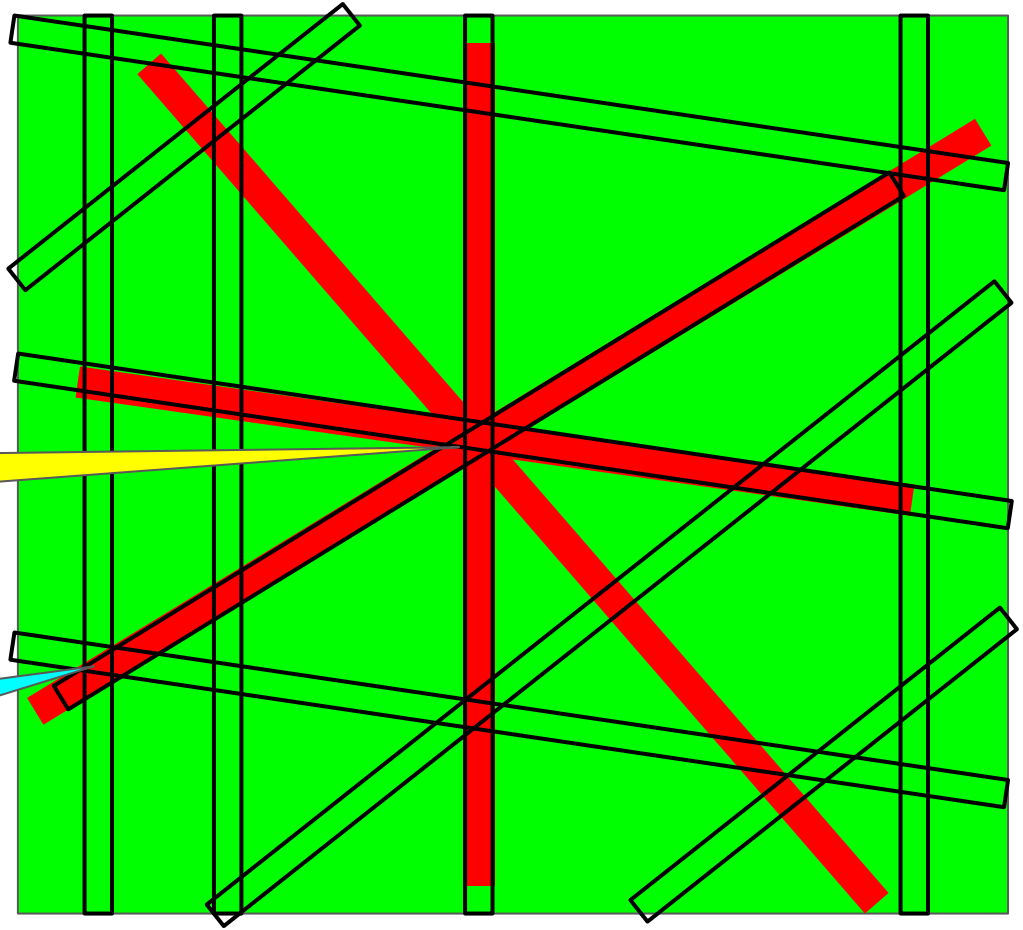
- Had too many samples, more than n^4, while we need $n^{1+o(1)}$.
- Only proved a weaker notion of confidence error.

Line Samplers

A sampler whose samples are lines.

Some points still won't be corrected, but few of these

Most points (even corrupted points), most lines through that point don't oversample A.



How good are line samplers?

For q queries, the probability they oversample is about $\eta \cong 1/q$.

Comes from pairwise independence.

NOT GOOD ENOUGH! Need $\eta << 1/q$.

This is the best lines (or subspaces) can do!

Solutions?

- Use curves (works, but gives much worse rate).
- Use several lines through a point (extends to lifted Reed-Solomon codes).

Third Result: New Curve Samplers

Theorem:

For any integers b and d such that b|d and b > 1, and any $\varepsilon > 0$ there is a degree b-curve sampler for $\mathbb{F}^d$ (where $\mathbb{F}$ is the field of order p) of size $|\mathbb{F}|^{d+poly(b)}$ with accuracy error ε and strong confidence error:

$$2b\left(2b/\varepsilon\sqrt{|\mathbb{F}|}\right)^b$$

Number of samples are close to $n = |\mathbb{F}|^d$.

Prior curve samplers had many more samples, and they did not prove strong confidence error.

Sampler Construction

First sample a subspace.

Epsilon biased sets in extension field. Gives a line sampler, which is a subspace sampler over original field.

Sub-sample with curves.

Since subspace is small, use all low degree curves as a sampler.

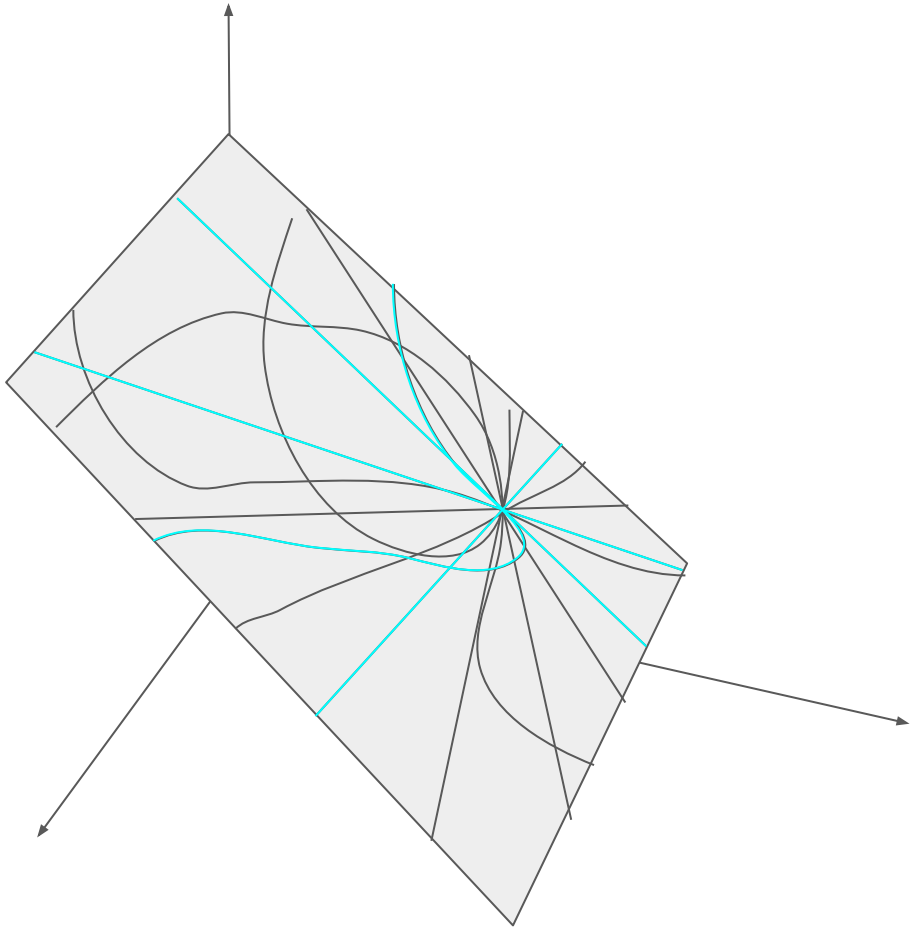
Choose **one** curve.

Do low degree correction on that curve.

Or a sample a few lines through a point in that subspace.

Choose **a few** lines.

Correct on each line and take a majority.



More Details

Extension Fields

Finite fields $\mathbb{F}$ with prime order p is just $\mathbb{Z}_p$.

The extension field, $\mathbb{F}_{p^k}$, has elements of the form

$$f(x) = \sum_i a_i y^i \bmod p(y)$$

Where each $a_i \in \mathbb{F}$, and p(y) is irreducible, degree k.

Formal polynomial. Don’t view as a function, the polynomial IS the element.

Lines Through Extension Fields

Line through $\mathbb{F}'^{d}$ is a function of the form, for some u, v $\in \mathbb{F}'^{d}$:

$$L(x) = u + x\,v$$

If $\mathbb{F}'$ is an extension field of $\mathbb{F}$, then for x $\in \mathbb{F}$:

$$\begin{aligned} L(x) &= u + x\,v \\ &= \textstyle\sum_i u_i y^i \bmod p(y) + (\sum_i x_i y^i \bmod p(y))(\sum_i v_i y^i \bmod p(y)) \\ &= \textstyle\sum_i u_i y^i \bmod p(y) + \sum_{i,j} x_i v_j y^{i+j} \bmod p(y) \end{aligned}$$

None of the x_i are multiplied by each other! Resulting coefficients are linear functions of the coefficients of x!

Line Samplers

So subspace samplers come from line samplers.

- Line samplers through ε-biased sets.
- ε-biased sets (Jalan, Moshkovitz using techniques from Ta-Shma).

Prior curve samplers use extension fields (Ta-Shma and Umans).

- Uses curves, not lines over extension field.
  - Doesn't let us subsample lines (needed for good rate).
  - Not as randomness efficient (requires n^4 samples).

Better Rate with Fewer Queries.

Reed-Muller gives bad rate for few queries.

Use a closely related code called Lifted Reed-Solomon.

- Low degree only when restricted to lines.
- For high degree and low characteristic, more general than Reed-Muller.

Lifted Reed-Solomon with high rate (and few queries) has low distance.

Use similar distance amplification technique as Kopparty, Saraf, and Yekhanin.

List Decoding

List Decoding:
decoding beyond half

w is not within **δ** of a codeword.

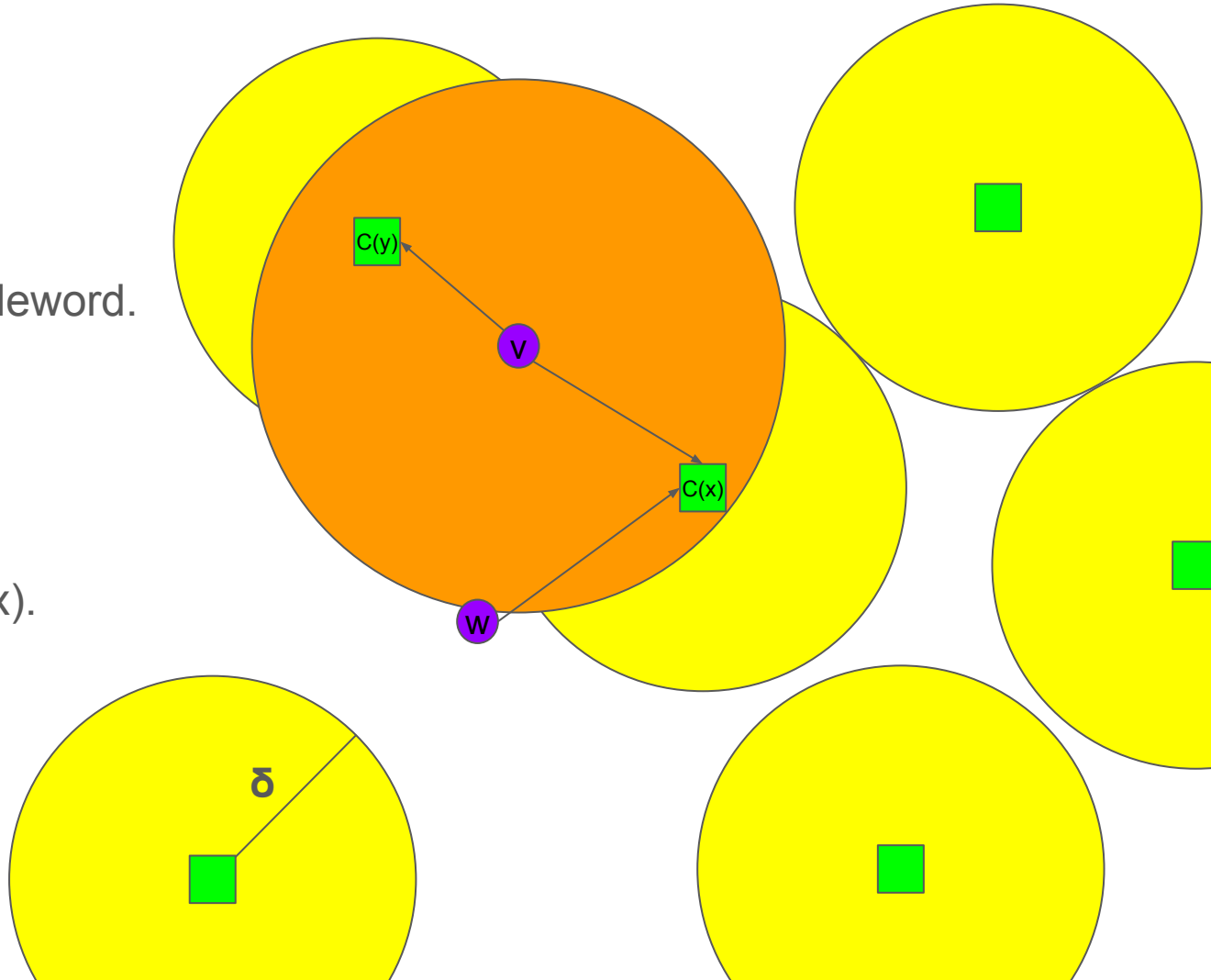
But clearly is from x.

v is within **δ** of a C(y),

But really came from C(x).

V is only close to two codewords.

Let it output both.



List Decoding

Definition:

A list decoding algorithm for a code C: $\Sigma^k \rightarrow \Sigma^n$ is an algorithm $D:\Sigma^n \rightarrow (\Sigma^k)^L$ such that for any $w \in \Sigma^n$, $x \in \Sigma^k$ where $Pr_i[C(x)_i = w_i] \geq \gamma$ for some $j \in [L]$ we have

$$D(w)_j = x.$$

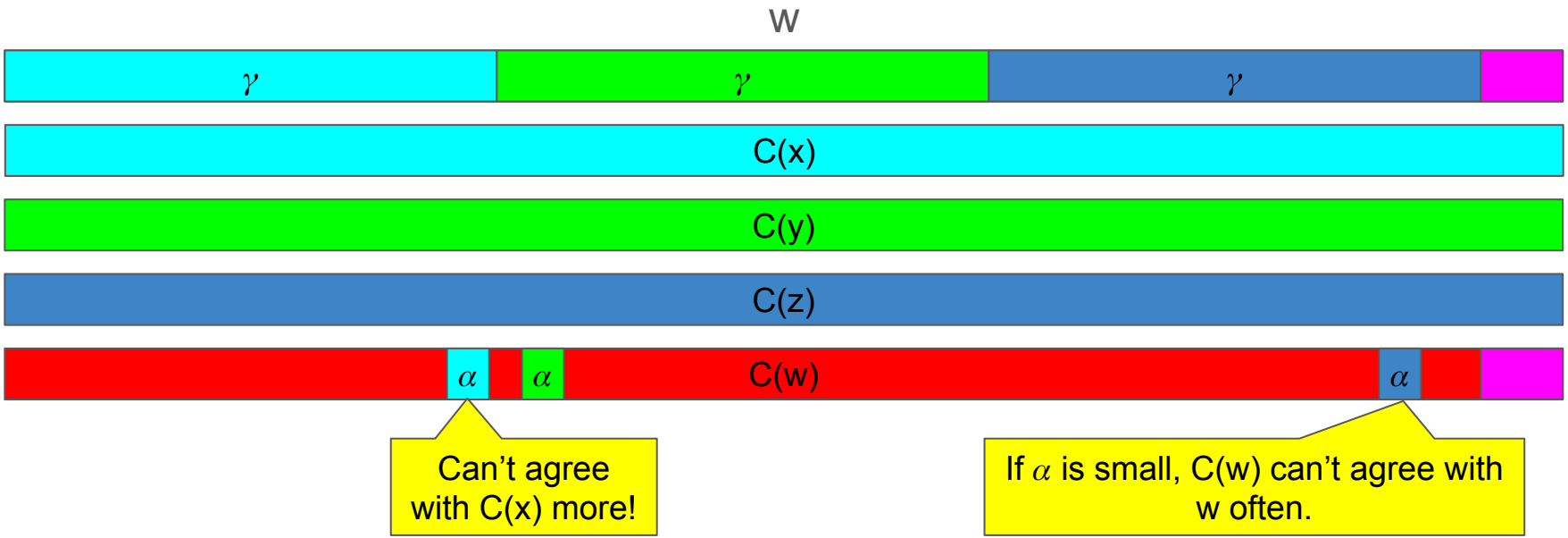
D outputs a list of L codewords.

Any codeword, C(x), that agrees with w on γ fraction of symbols is in the list.

Does List Decoding Help?

Yes for codes with large distance (δ = 1-α close to 1): the Johnson bound.

All words only have ~ 1/γ codewords that agree on γ fraction of points.



Efficient **Uniform** List Decoders

Theorem:

For any constants $\tau > 0$ and $\gamma > 0$, there exists an infinite family of uniform asymptotically good codes that can be list decoded from 1 - γ fraction of errors in time $n^{1+\tau}$ and space n^{τ}.

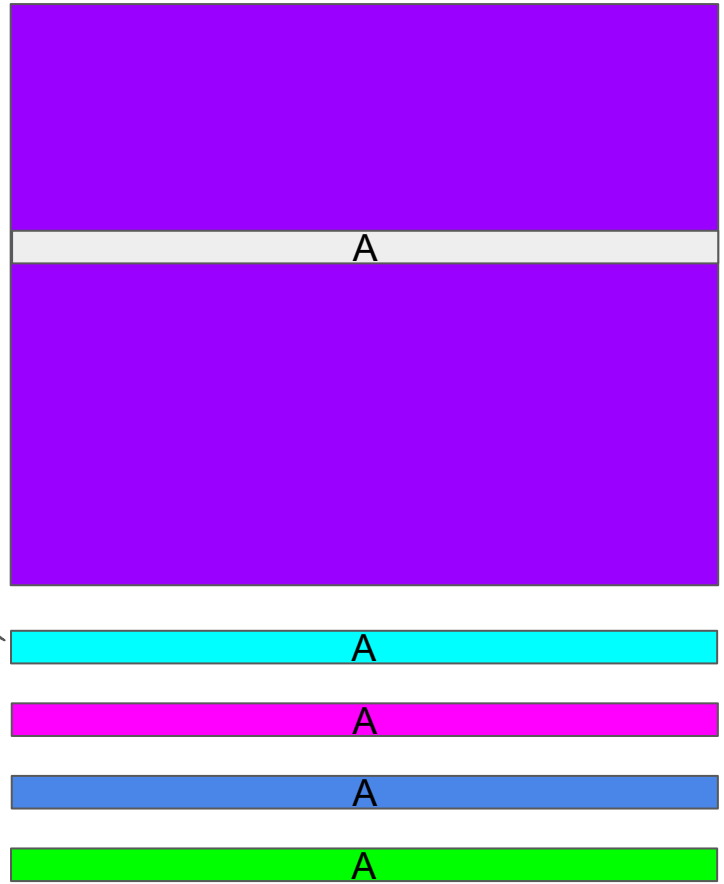
These are Reed-Muller codes.

Great! Use Local List Decoding!

Which one “improves” the word?

All of them!

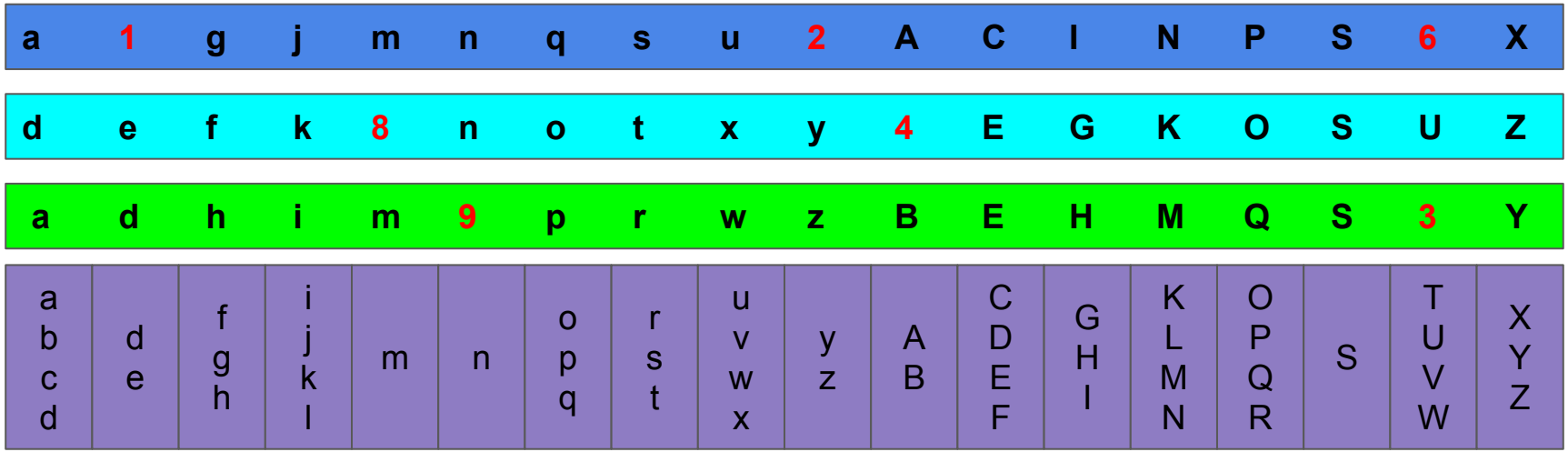
But they come from “different” codewords.



List Recovery

Instead of receiving a single word, receive several candidate symbols per index.

Output all codewords who have many symbols in those lists.



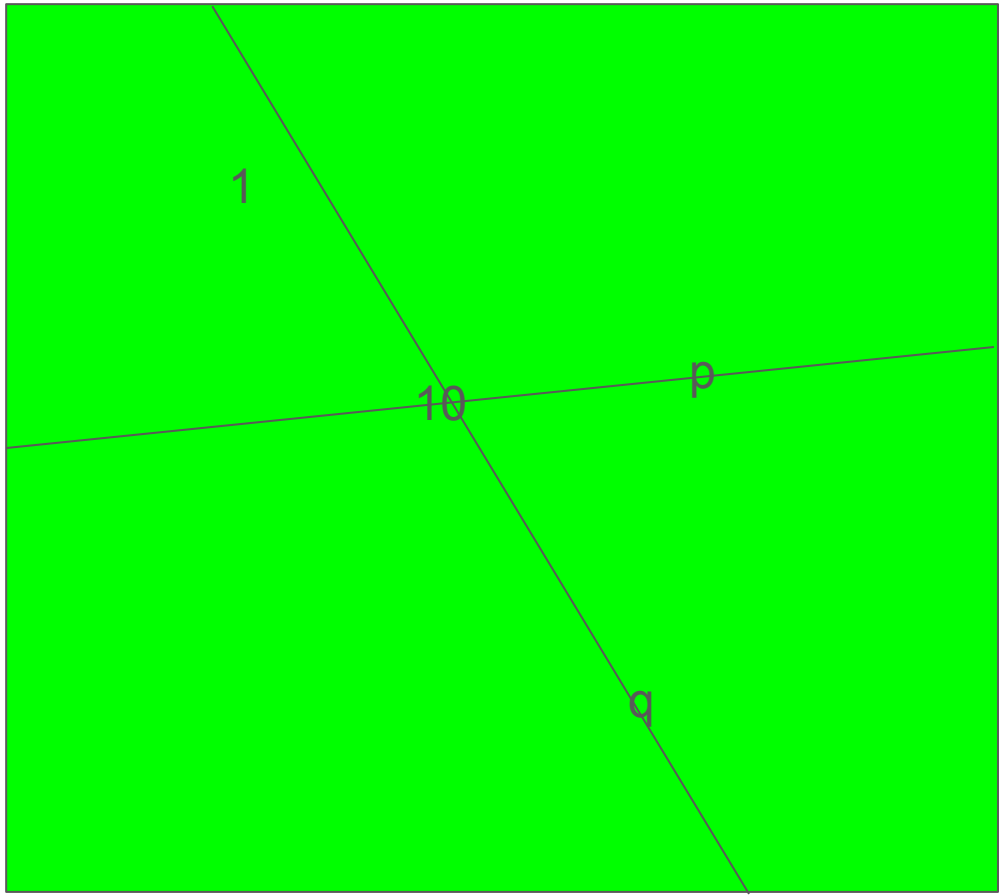
Goal: Make a Zero Error List Recovery Instance

Can’t have many codewords all in short lists.

Need to separate list into specific codewords.

Use a point all codewords differ to decode.

Decode with lines through that point.

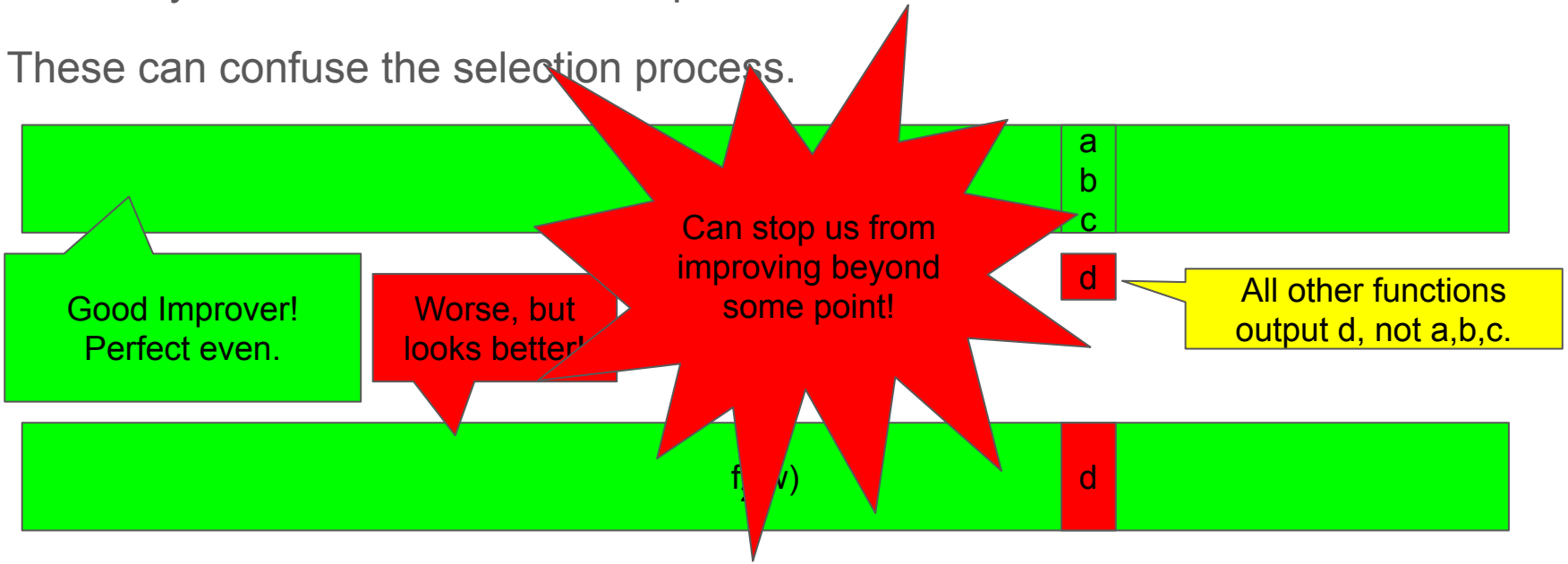


Problem: Choosing an Improver

Before, chose an improver who agreed with other improvers on many places.

Generally allow list decoders to output codewords *not* close to the received word.

These can confuse the selection process.



Two Kinds of Errors

In Codeword:

Symbols in a nearby codeword, missing from lists.

These will be removed with local corrections because of sampling.

Out of Codeword:

Symbols in lists that are not in nearby codewords.

Can cause false negatives.

Sampling can handle these as well…

AFTER the out of codeword errors are small.

Need these to be rare.

Local Testing

We need local corrections that only succeed if input is close to a codeword.

- Local testing in the list decoding (high error) setting.
- Local testing in the low randomness (few samples) setting.

List decoding setting: [RS97; AS97; MR06; BDLN17; MZ23; Har+24]

Low randomness setting: [BSVW03; MR06]

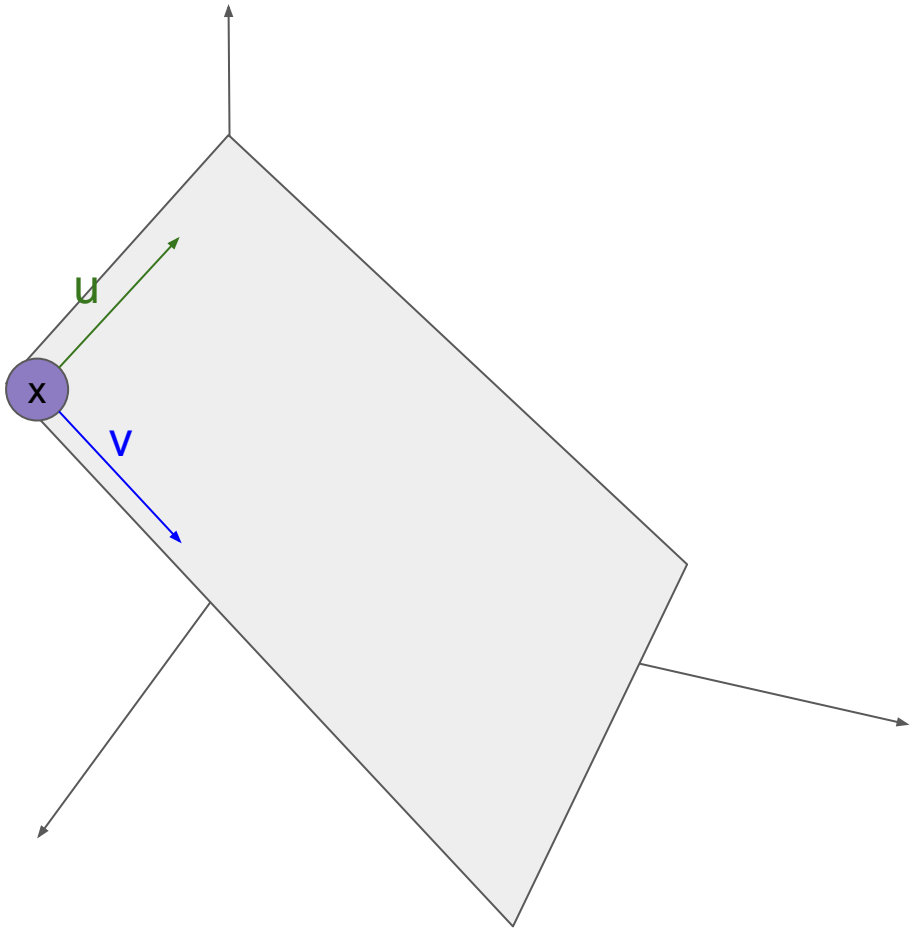
Moshkovitz and Raz give a low degree test that is both.

Randomness Efficient Plane vs Point Test

Choose a completely random point $x \in \mathbb{F}^d$.

Choose two directions from a smaller u, v $\in \mathbb{H}^d$. ($\mathbb{H} \subsetneq \mathbb{F}$)

Check if plane through point in directions is low degree.



Local List Testing (Informal)

Definition:

For some small $\eta > 0$, for all $\alpha > 0$, for $\alpha' = \alpha - \eta$ we have that:

For any assignments of the subspaces to low degree polynomials, g_S

The probability that a random g_S:

agrees with the received word at a random location and

Does not agree with the closest codewords (those within $(1-\alpha')$)

is at most α.

This implies that local correctors have few out-of-codeword errors.

Local Testing (Informal)

Theorem:

For some small $\eta > 0$, for all α, γ, $\epsilon > 0$ such that $\gamma(1-\alpha) > 4\eta$. Then there is a family of functions f such that for any word w:

Completeness: If y is a codeword that agrees with w on $\gamma + \epsilon$ of symbols, then

$$\Pr[y_i \notin f(w)_i] < \eta/\epsilon^2$$

Soundness: If Q is the multi-string of codewords that agree with w on $\gamma(1-\alpha) - 5\eta$ fraction of symbols, then

$$\Pr[f(w)_i \notin Q_i] < 1 - \alpha$$

So we just decode using these planes?

Subspaces (including planes) are not good enough samplers for our decoder…

But only need local testing in first round!

- After out of codeword errors are small, sampling arguments are sufficient.
- Can use prior sets of lines to correct from there.

How do we decide what a good function is?

For the first function (the planes with local testing) we don’t! Brute force, try all!

After that, most correctors output the exact right list most of the time.

Use the same argument as the unique decoding case.

Those Other Results

Tighter MA/1 Circuit Lower Bounds From Verifier Efficient PCPs for PSPACE

Tighter circuit lower bounds for **MA/1** (prior result by Santhanam).

MA/1 is a type of interactive proof where the prover gets a single message.

$$\mathrm{MATIME}[n^{a+o(1)}] \not\subseteq \mathrm{SIZE}[n^a] \text{ (for some } a > 1)$$

Tighter equivalence between **MIP** = **NEXP** (prior by, Babai, Fortnow and Lund).

MIP is a type of interactive proof where there are two provers.

NE has two prover, one round proofs with time $\tilde{O}(n)$ verifiers.

Time-Space vs Cumulative Memory in the Streaming Model.

with John Kallaugher and Niels Kornerupwith John Kallaugher and Niels Kornerup

Space Lower Bounds for Learning from extractors.

Problem: Given seeded extractor Ext which takes as input an n bit source, and a length n source **a**, output a single bit of **a**, $\mathbf{a}_i$, from samples of the form Ext(**a**, s) for uniformly random seeds s.

Result: Any algorithm that outputs $\mathbf{a}_i$ with high probability must either use space $\Omega(n)$ or read $2^{\Omega(n)}$ samples.

Prior results by Raz, Moshkovitz, Garg, Tal (and others)

- Only for extractors that output **1** bit! Ours apply to more general extractors.
- Our lower bounds are weaker, but tight! More output bits help learning.

Cumulative Memory in the Streaming Model, Continued

Used to give a streaming problem with small average case memory usage, O(log(n)), but requires large memory at some point ($\sqrt{n}$).

Why not use prior learning lower bounds?

- Their algorithm only work in the random order model.
  - Ours work for more standard, adversarially ordered streams.
- Prior results had large symbols in the stream (like size $n^{1/4}$).
  - Ours has more standard length O(log(n)) symbols.

Time and Space Efficient Encoding

Bazzi and Mitter proved that linear time encoders for good codes use linear space.

So we can only hope for almost linear time.

(Punctured) Repeat Accumulate: time O(nlog(n)) space O(log(n))

Non-Uniform

Gál, Hansen, Koucký, Pudlák and Viola: time O(n polylog(n)) space O(log(n))

Non-Uniform

Us: time $n^{1+o(1)}$ space $n^{o(1)}$, ***Uniform***

Efficient Interactive Proofs

Time T space S algorithms have interactive proofs with:

- Verifiers that run in time $\tilde{O}(n + S\log(T))$.
- Verifiers that run in space $\tilde{O}(S)$.
- Provers that run in time $2^{O(S)}$.

Even holds for nondeterministic algorithms (with up to log(T) alternations).

Similar results also hold for randomized algorithms.

Open Problems

Open Problems - About Codes

1. Find a single code that is encodable and decodable in space $n^{o(1)}$, time $n^{1+o(1)}$.
2. Give List Decoders with better constants. Ours are like 10^{20}.
3. Give time/space efficient uniform decoders for more codes (like multiplicity codes).
    a. Our uniform decoders are only for lifted Reed-Solomon.
    b. Our technique for multiplicity codes only gives non-uniform decoders.
4. Give List Decoders for codes with constant rate when achieving space $n^{o(1)}$, time $n^{1+o(1)}$.

Open Problems - About Interactive Proofs

1. For any time T space S algorithms, give an interactive proof whose:
   a. Verifier runs in time poly(S)
   b. Prover runs in time poly(T)

   (This is solved when T = poly(S) or log(T) = O(S), but not in general)

2. Give PCPs whose prover runs in (Complexity Preserving PCPs
   a. Time almost T and space almost S.
   b. Reason to think is impossible (Based on encoding lower bounds)
      i. Just give prover time times space less than S T (of original)

Thanks To My Collaborators and Other Students

Co-Authors

- Dana Moshkovitz
- Ron Rothblum
- Niels Kornerup
- John Kallaugher

And all the other faculty and students I have had many great conversations with.

Extra Technical Details

Why not multiplicity?

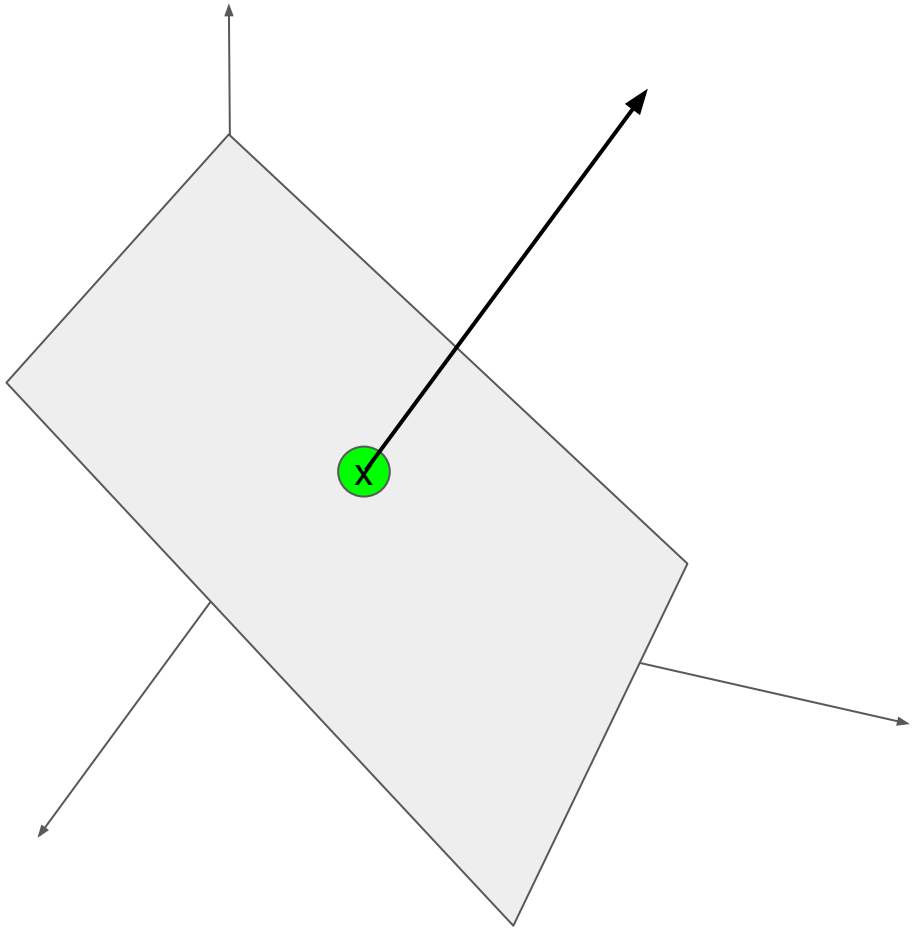
Multiplicity codes based on low degree polynomial, p.

Symbols not just p(x), but also contains derivatives of p at x.

For instance, contains all the first order derivatives.

Recovering it requires derivatives in all directions.

Can’t get directional derivatives outside that plane!



Why not Multiplicity codes?

Multiplicity codes need psuedorandom lines, like ours.

But multiplicity needs lines in directions that spans the space.

Otherwise some directional derivatives cannot be recovered.

Our first sampling step restricts us to a subspace:

Can’t get derivatives outside that subspace.

Similar sampler *may* work, but the samples need more structure than just being lines (a single line is not sufficient for correcting even a single symbol).

However, our techniques give a non-uniform decoder.

Reed Solomon Decoding

To decode Reed-Muller we decode along lines.

Reed-Muller codes are codes that are low degree polynomials when restricted to lines.

A line decodes correctly if it samples less than (p-d)/2 corruptions.

Codeword is degree d polynomial in field size p