Algebraic Algorithms and Coding Theory

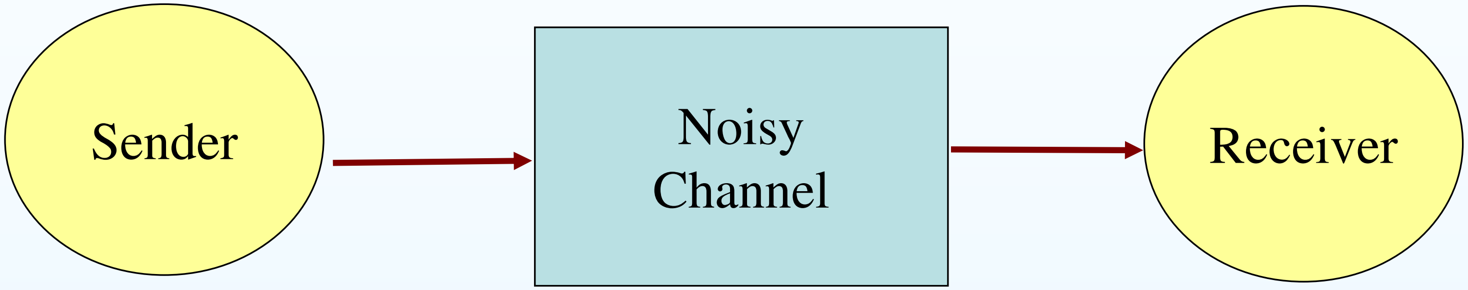
Madhu Sudan

— A Survey —

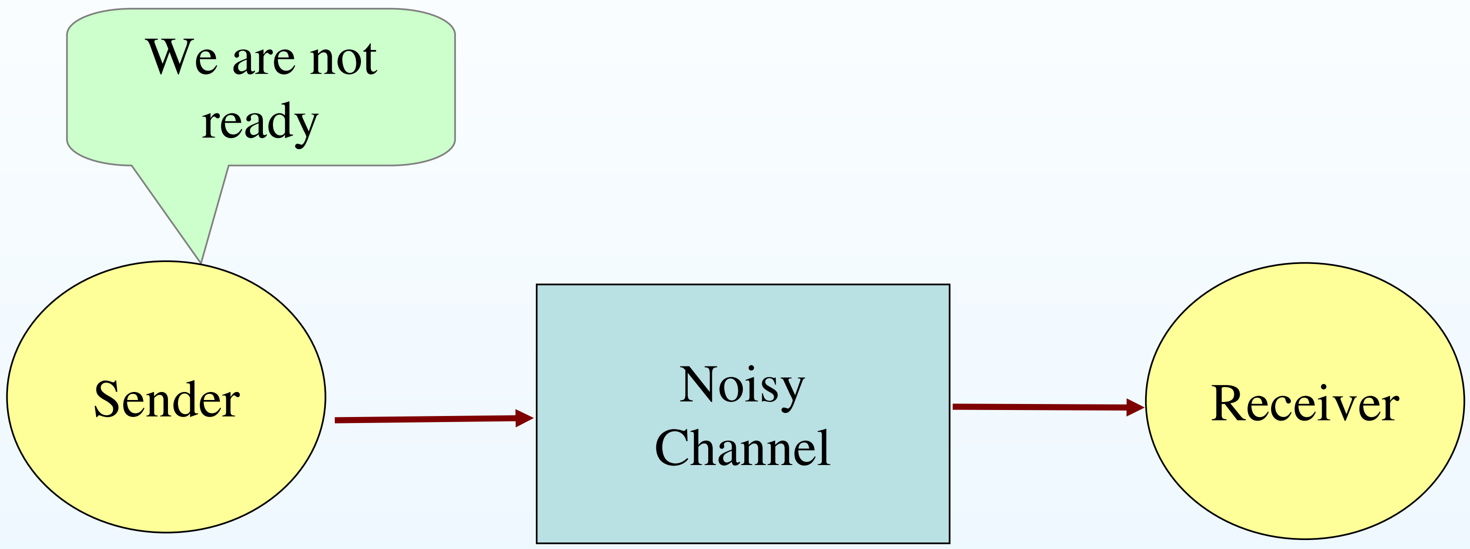
MIT CSAIL

Part 1: Introduction to Coding Theory

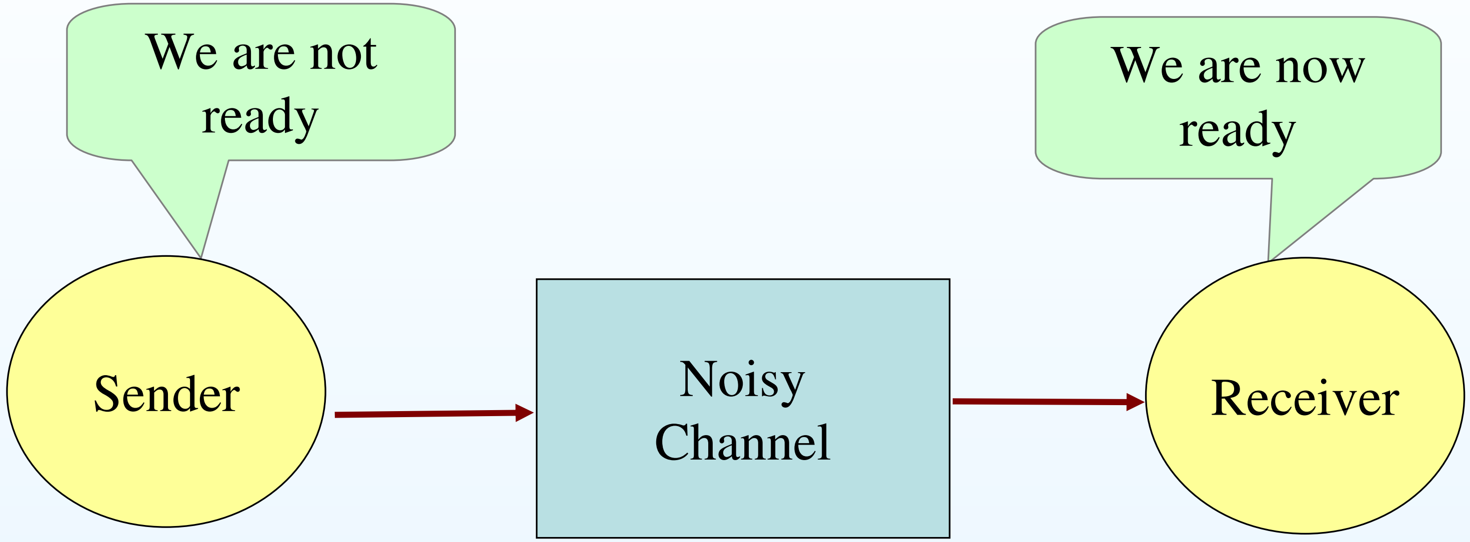
The Problem of Information Transmission



The Problem of Information Transmission



The Problem of Information Transmission



- When information is digital, reliability is critical.
- Need to understand errors, and correct them.

Shannon (1948)

- Model noise by probability distribution.
- Example: Binary symmetric channel (BSC)
  - Parameter $p \in [0, \frac{1}{2}]$.
  - Channel transmits bits.
  - With probability $1 - p$ bit transmitted faithfully, and with probability p bit flipped (independent of all other events).

Shannon's architecture

- Sender encodes k bits into n bits.
- Transmits n bit string on channel.
- Receiver decodes n bits into k bits.
- Rate of channel usage = k/n.

Shannon's theorem

- Every channel (in broad class) has a capacity s.t., transmitting at Rate below capacity is feasible and above capacity is infeasible.
- Example: Binary symmetric channel (p) has capacity $1 - H(p)$, where $H(p)$ is the binary entropy function.
  - $p = 0$ implies capacity $= 1$.
  - $p = \frac{1}{2}$ implies capacity $= 0$.
  - $p < \frac{1}{2}$ implies capacity > 0.
- Example: q-ary symmetric channel (p): On input $\sigma \in \mathbb{F}_q$ receiver receives (independently) σ', where
  - $\sigma' = \sigma$ w.p. $1 - p$.
  - σ' uniform over $\mathbb{F}_q - \{\sigma\}$ w.p. p.

  Capacity positive if $p < 1 - 1/q$.

Constructive versions

- Shannon's theory was non-constructive. Decoding takes exponential time.
- [Elias '55] gave polytime algorithms to achieve positive rate on every channel of positive capacity.
- [Forney '66] achieved any rate $<$ capacity with polynomial time algorithms (and exponentially small error).
- Modern results (following [Spielman '96]) lead to linear time algorithms.

Hamming (1950)

- Modelled errors adversarially.
- Focussed on image of encoding function (the “Code”).
- Introduced metric (Hamming distance) on range of encoding function. $d(x, y) = \#$ coordinates such that $x_i \neq y_i$.
- Noticed that for adversarial error (and guaranteed error recovery), distance of Code is important.

$$\Delta(C) = \min_{x,y \in C} \{d(x, y)\}.$$

- Code of distance d corrects $(d-1)/2$ errors.

Contrast between Shannon & Hamming

Contrast between Shannon & Hamming

[Sha48] : $\boxed{\mathcal{C}}$ probabilistic.

E.g., flips each bit independently w.p. p.

✔ Tightly analyzed for many cases e.g., q-SC(p).

✗ Channel may be too weak to capture some scenarios.

✗ Need very accurate channel model.

Contrast between Shannon & Hamming

[Sha48] : $\boxed{\mathcal{C}}$ probabilistic.

✔ Corrects many errors. ✗ Channel restricted.

Contrast between Shannon & Hamming

[Sha48] : $\boxed{\mathcal{C}}$ probabilistic.

✔ Corrects many errors. ✗ Channel restricted.

[Ham50] : $\boxed{\mathcal{C}}$ flips bits *adversarially*

✔ Safer model, “good” codes known

✗ Too pessimistic: Can only decode if $p < 1/2$ for any alphabet.

Contrast between Shannon & Hamming

[Sha48] : $\boxed{\mathcal{C}}$ probabilistic.

✔ Corrects many errors. ✗ Channel restricted.

[Ham50] : $\boxed{\mathcal{C}}$ flips bits *adversarially*

✗ Fewer errors. ✔ More general errors.

Contrast between Shannon & Hamming

[Sha48] : $\boxed{\mathcal{C}}$ probabilistic.

✔ Corrects many errors. ✗ Channel restricted.

[Ham50] : $\boxed{\mathcal{C}}$ flips bits *adversarially*

✗ Fewer errors. ✔ More general errors.

- Which model is correct? Depends on application.
  - Crudely: Small $q \Rightarrow$ Shannon. Large $q \Rightarrow$ Hamming.
- Recent work: New Models of error-correction + algorithms.
  - List-decoding: Relaxed notion of decoding.

Contrast between Shannon & Hamming

[Sha48] : $\boxed{\mathcal{C}}$ probabilistic.

✔ Corrects many errors. ✗ Channel restricted.

[Ham50] : $\boxed{\mathcal{C}}$ flips bits *adversarially*

✗ Fewer errors. ✔ More general errors.

- Which model is correct? Depends on application.
  - Crudely: Small $q \Rightarrow$ Shannon. Large $q \Rightarrow$ Hamming.
- Recent work: New Models of error-correction + algorithms.
  - List-decoding: Relaxed notion of decoding.

    ✔ More errors ✔ Strong (enough) errors.

Summary of origins

- Two seminal works:
  - [Shannon]: A Mathematical Theory of Communication.
  - [Hamming]: Error-detecting and error-correcting codes.
- Both went way beyond the immediate motivations and examined far-reaching subjects. (Shannon more so than Hamming?)
- Fundamental questions:
  - [Shannon]: Find capacity of various channels explicitly. Find efficient encoding and decoding functions.
  - [Hamming]: Given q find best tradeoff between Rate of code and (fractional) distance.

Development

Great progress over last sixty years. Some sample results:

- 1950-1960: First families of codes. Algebraic coding theory.
  - Reed-Muller Codes.
  - Reed-Solomon Codes.
  - BCH Codes.
- 1960-1970: Algorithmic focus intensifies.
  - Peterson. Berlekamp-Massey.
  - Gallager - LDPC codes.
  - Forney - Concatenated codes.
- 1970-1980: Deep theories.
  - Linear Programming bound.
  - Lovasz on Shannon Capacity.
  - Justesen's codes.

Development (contd.)

- 1980-1990: Algebraic-Geometry codes. (started in mid 70's by Goppa). Better than random!
- 1990-today: Algorithms:
  - Linear time decoding.
  - Approaching Shannon capacity in practice.
  - List-decoding: Best of Hamming+Shannon worlds.
- Today: Focus on algebraic, algorithmic, aspects.

Part 2: Algebraic Error-Correcting Codes

Motivation: [Singleton] Bound

- Suppose $C \subseteq \mathbb{F}_q^n$ has q^k codewords. How large can its distance be?

Motivation: [Singleton] Bound

- Suppose $C \subseteq \mathbb{F}_q^n$ has q^k codewords. How large can its distance be?
- Bound: $\Delta(C) \leq n - k + 1$.

Motivation: [Singleton] Bound

- Suppose $C \subseteq \mathbb{F}_q^n$ has q^k codewords. How large can its distance be?
- Bound: $\Delta(C) \leq n - k + 1$.
- Proof:
  - Project code to first $k - 1$ coordinates.
  - By Pigeonhole Principle, two codewords collide.
  - These two codewords thus disagree in at most $n - k + 1$ coordinates.

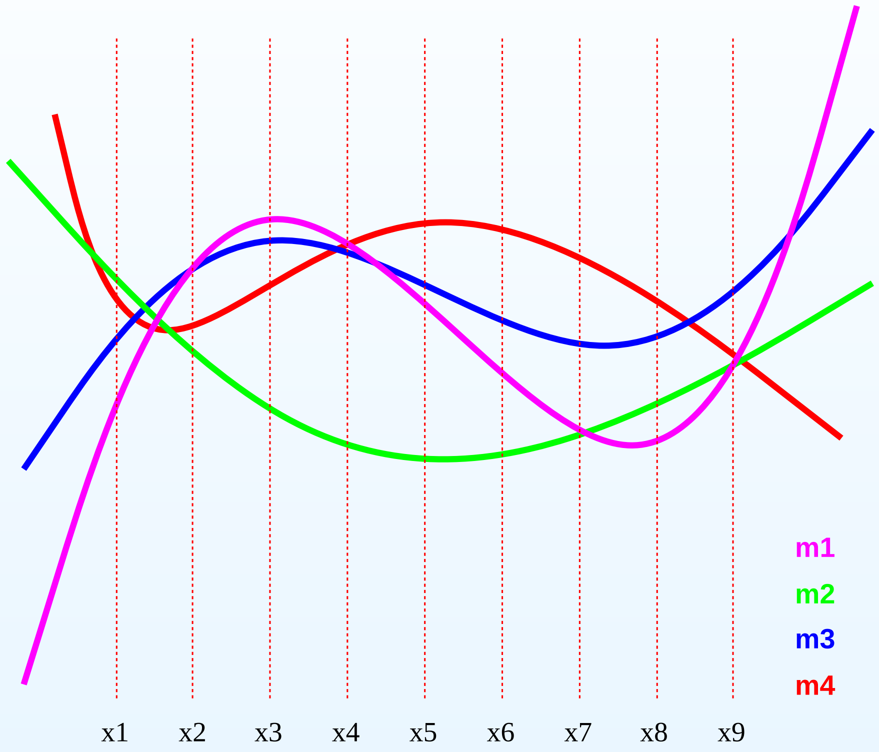
Motivation: [Singleton] Bound

- Suppose $C \subseteq \mathbb{F}_q^n$ has q^k codewords. How large can its distance be?
- Bound: $\Delta(C) \leq n - k + 1$.
- Proof:
  - Project code to first $k - 1$ coordinates.
  - By Pigeonhole Principle, two codewords collide.
  - These two codewords thus disagree in at most $n - k + 1$ coordinates.
- Surely we can do better?

Motivation: [Singleton] Bound

- Suppose $C \subseteq \mathbb{F}_q^n$ has q^k codewords. How large can its distance be?
- Bound: $\Delta(C) \leq n - k + 1$.
- Proof:
  - Project code to first $k - 1$ coordinates.
  - By Pigeonhole Principle, two codewords collide.
  - These two codewords thus disagree in at most $n - k + 1$ coordinates.
- Surely we can do better?
- Actually - No! [Reed-Solomon] Codes match this bound!

Reed-Solomon Codes



- Messages $\equiv$ Polynomial.
- Encoding $\equiv$ Evaluation at $x_1, \ldots, x_n$.
- $n >$ Degree: Injective
- $n \gg$ Degree: Redundant

Reed-Solomon Codes (formally)

- Let $\mathbb{F}_q$ be a finite field.
- Code specified by $k, n, \alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$.
- Message: $\langle c_0, \ldots, c_k \rangle \in \mathbb{F}_q^{k+1}$ coefficients of degree k polynomial $p(x) = c_0 + c_1 x + \cdots c_k x^k$.
- Encoding: $p \mapsto \langle p(\alpha_1), \ldots, p(\alpha_n) \rangle$. ($k+1$ letters to n letters.)
- Degree k poly has at most k roots $\Leftrightarrow$ Distance $d = n - k$.
- These are the Reed-Solomon codes.
  Match [Singleton] bound!
  Commonly used (CDs, DVDs etc.).

Algebraic Error-Correcting Codes

- Broad class of codes. Include
  - Reed-Muller codes.
  - (Dual)-BCH codes.
  - Algebraic-Geometry (AG) codes (or Goppa codes).

Algebraic Error-Correcting Codes

- Broad class of codes.

Algebraic Error-Correcting Codes

- Broad class of codes.

- Unifying theme:
  - Message space = Collection of algebraic functions.
  - Encoding = Evaluation over (carefully chosen) subset of vector space over field.

Algebraic Error-Correcting Codes

- Broad class of codes.
- Unifying theme:
  - Message space = Collection of algebraic functions.
  - Encoding = Evaluation over (carefully chosen) subset of vector space over field.
- Distance analysis: Varies.
  - Reed-Solomon/Reed-Muller: # roots of low-degree polynomials.
  - BCH: $x^p + y^p = (x+y)^p$.
  - Dual-BCH; Weil Bounds.
  - AG: Reimann-Roch, Drinfeld-Vladuts bound, Weil bound etc.

Algebraic Error-Correcting Codes

- Broad class of codes.
- Unifying theme:
  - Message space = Collection of algebraic functions.
  - Encoding = Evaluation over (carefully chosen) subset of vector space over field.
- Distance analysis: Varies.

Algebraic Error-Correcting Codes

- Broad class of codes.
- Unifying theme:
  - Message space = Collection of algebraic functions.
  - Encoding = Evaluation over (carefully chosen) subset of vector space over field.
- Distance analysis: Varies.
- Stunning combinatorial implications:
  - Often better than probabilistic method.
  - Give asymptotic performance that is unmatched by other combinatorial techniques.
  - Often very specific to fields.

Part 3: Algebraic Algorithms & Coding

A Brief History

Algorithmic issues in Coding

- Most basic problem: Decoding Problem for Codes
  - Fix $C \subseteq \mathbb{F}_q^n$.
  - Transmit $c \in C$. Receive $y \in \mathbb{F}_q^n$ such that $\Delta(c, y) \leq e$.
  - Receiver's (algorithmic) problem: Given y, compute c (if it is uniquely determined).

Algorithmic issues in Coding

- Most basic problem: Decoding Problem for Codes

Algorithmic issues in Coding

- Most basic problem: Decoding Problem for Codes

- Looks like any other NP-search problem.
  - Enumerate error locations?
  - Enumerate codewords?

Algorithmic issues in Coding

- Most basic problem: Decoding Problem for Codes

- Looks like any other NP-search problem.

Algorithmic issues in Coding

- Most basic problem: Decoding Problem for Codes

- Looks like any other NP-search problem.

- For carefully designed codes, can beat brute force search!
  In case of algebraic codes, this often uses non-trivial algebraic algorithms.
  - Classical Reed-Solomon decoding: Interpolation.
  - Decoding of AG codes: Grobner basis algorithms.
  - Modern Reed-Solomon decoding: Factorization of bivariate polynomials.
  - Number-theoretic analogs: Factorization over rationals + lattice algorithms.
  - AG codes: Factorization over other rings.

Algebraic Algorithms inspired by Coding

- Fewer examples, but they do exist!
- [Berlekamp]'s algorithm for factoring over finite fields motivated by need to decode faster.
- Recent development: [Umans], [Kedlaya-Umans] give nearly linear-time algorithm for modular polynomial composition, inspired by some “list-decoding” successes. Leads to $O(n^{1.5})$ time algorithm for factorization of polynomials over finite fields.

Part 4: Decoding from High-Error

The List-Decoding Problem

Error-correction capability vs. Rate

- Let $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$ be the “best” code of rate R, i.e., $R = k/n$.

Error-correction capability vs. Rate

- Let $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$ be the “best” code of rate R, i.e., $R = k/n$.
- How many errors can it help correct?
  - $m \longrightarrow E(m) \xrightarrow{\text{Channel}} y \to \text{Receiver}$ s.t. $\Delta(E(m), y) \leq e$.
  - For what e is m efficiently computable?
  - For what e is m uniquely determined (by E and y)?

Error-correction capability vs. Rate

- Let $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$ be the "best" code of rate R, i.e., $R = k/n$.
- How many errors can it help correct?
  - $m \longrightarrow E(m) \xrightarrow{\text{Channel}} y \to \text{Receiver}$ s.t. $\Delta(E(m), y) \leq e$.
  - For what e is m efficiently computable?
  - For what e is m uniquely determined (by E and y)?
- Depends on model of error!
  - [Shannon]: Errors random $\Rightarrow p = e/n \to 1 - R$.
  - [Hamming]: Errors adversarial $\Rightarrow p = e/n \to (1 - R)/2$.
    (Adversary picks codewords that disagree in $1 - R$ fraction of coordinates and lets y be halfway between them!).

Error-correction capability vs. Rate

- Let $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$ be the "best" code of rate R, i.e., $R = k/n$.
- How many errors can it help correct?
  - $m \longrightarrow E(m) \xrightarrow{\text{Channel}} y \to \text{Receiver}$ s.t. $\Delta(E(m), y) \leq e$.
  - For what e is m efficiently computable?
  - For what e is m uniquely determined (by E and y)?
- Depends on model of error!
  - [Shannon]: Errors random $\Rightarrow p = e/n \to 1 - R$.
  - [Hamming]: Errors adversarial $\Rightarrow p = e/n \to (1 - R)/2$.
- [Elias] Also depends on notion of error-correction!
  - Requirement that m be uniquely determined is too restrictive.
  - In most (practical and theoretical) cases, suffices to narrow m down to a small (poly-sized) list.

Relaxed Decoding : List Decoding

$$\mathbf{m} \longrightarrow \boxed{S} \xrightarrow{E(m)} \boxed{C} \xrightarrow{y} \boxed{LD} \xrightarrow[\exists i\,:\,\mathbf{z}_i = \mathbf{m}]{\mathbf{z}_1, \ldots, \mathbf{z}_L}$$

- List decoder $\boxed{LD}$ outputs a short *list* of all possible messages.

Relaxed Decoding : List Decoding

$$\mathbf{m} \longrightarrow \boxed{S} \xrightarrow{E(m)} \boxed{C} \xrightarrow{y} \boxed{LD} \xrightarrow[\exists i\,:\,\mathbf{z}_i = \mathbf{m}]{\mathbf{z}_1, \ldots, \mathbf{z}_L}$$

- List decoder $\boxed{LD}$ outputs a short *list* of all possible messages.
- Notion due to [Elias57, Wozencraft58].

Relaxed Decoding : List Decoding

$$\mathbf{m} \longrightarrow \boxed{S} \xrightarrow{E(m)} \boxed{C} \xrightarrow{y} \boxed{LD} \xrightarrow[\exists i \,:\, \mathbf{z}_i = \mathbf{m}]{\mathbf{z}_1, \ldots, \mathbf{z}_L}$$

- List decoder $\boxed{LD}$ outputs a short *list* of all possible messages.
- Notion due to [Elias57, Wozencraft58].
- [Zyablov-Pinsker70s] Exist codes ($\boxed{S}$, $\boxed{LD}$) correcting $(1 - R - \epsilon)$-fraction errors with rate R (if $q \geq q(1/\epsilon)$.

Relaxed Decoding : List Decoding

$$\mathbf{m} \to \boxed{S} \xrightarrow{E(m)} \boxed{C} \xrightarrow{y} \boxed{LD} \xrightarrow[\exists i : \mathbf{z}_i = \mathbf{m}]{\mathbf{z}_1, \ldots, \mathbf{z}_L}$$

- List decoder $\boxed{LD}$ outputs a short *list* of all possible messages.
- Notion due to [Elias57, Wozencraft58].
- [Zyablov-Pinsker70s] Exist codes ($\boxed{S}$, $\boxed{LD}$) correcting $(1 - R - \epsilon)$-fraction errors with rate R (if $q \geq q(1/\epsilon)$.
- Matches Shannon! Adversarial Error! But non-constructive!

Relaxed Decoding : List Decoding

$$\mathbf{m} \longrightarrow \boxed{S} \xrightarrow{E(m)} \boxed{C} \xrightarrow{\;y\;} \boxed{LD} \xrightarrow[\exists i \,:\, \mathbf{z}_i = \mathbf{m}]{\mathbf{z}_1, \ldots, \mathbf{z}_L}$$

- List decoder $\boxed{LD}$ outputs a short *list* of all possible messages.
- Notion due to [Elias57, Wozencraft58].
- [Zyablov-Pinsker70s] Exist codes ($\boxed{S}$, $\boxed{LD}$) correcting $(1 - R - \epsilon)$-fraction errors with rate R (if $q \geq q(1/\epsilon)$.
- Matches Shannon! Adversarial Error! But non-constructive!
- Questions:
  - Can we find such codes?
  - Can we decode them?
  - Do Reed-Solomon codes have the desired property?

List-decodability of Reed-Solomon Codes

- A general result: Code of distance $(1-\tau)\cdot n$ is always combinatorially-list-decodable from $(1-\sqrt{\tau})\cdot n$ errors. [Johnson] Bound.
  - If $\tau \to 0$, fraction of errors approaches 100%.
- Implication for Reed-Solomon codes:
  - For any function $f : \mathbb{F}_q \to \mathbb{F}_q$ there are at most $\ell \leq q^2$ polynomials $p_1, \ldots, p_\ell$ of degree $k = R \cdot n$ that agree with f on $\sqrt{R} \cdot n$ points.
  - Open: What about $(R+\epsilon)n$ agreement?
- Algorithmic issues:
  - Find $p_1, \ldots, p_\ell$ efficiently?
  - Find better code? that can decode from $(R+\epsilon)n$ agreement?

Rest of the talk

- List-decoding of Reed-Solomon codes
  - Rate R codes upto $\sqrt{2R}$-fraction agreement.
  - Rate R codes upto $\sqrt{R}$-fraction agreement.
- List-decoding of Folded Reed-Solomon codes
  - Rate R codes upto $R + \epsilon$-fraction agreement.
- Makes essential use of algebraic algorithms!

Part 5:
List-Decoding of Reed-Solomon Codes

Reed-Solomon Decoding

Restatement of the problem:

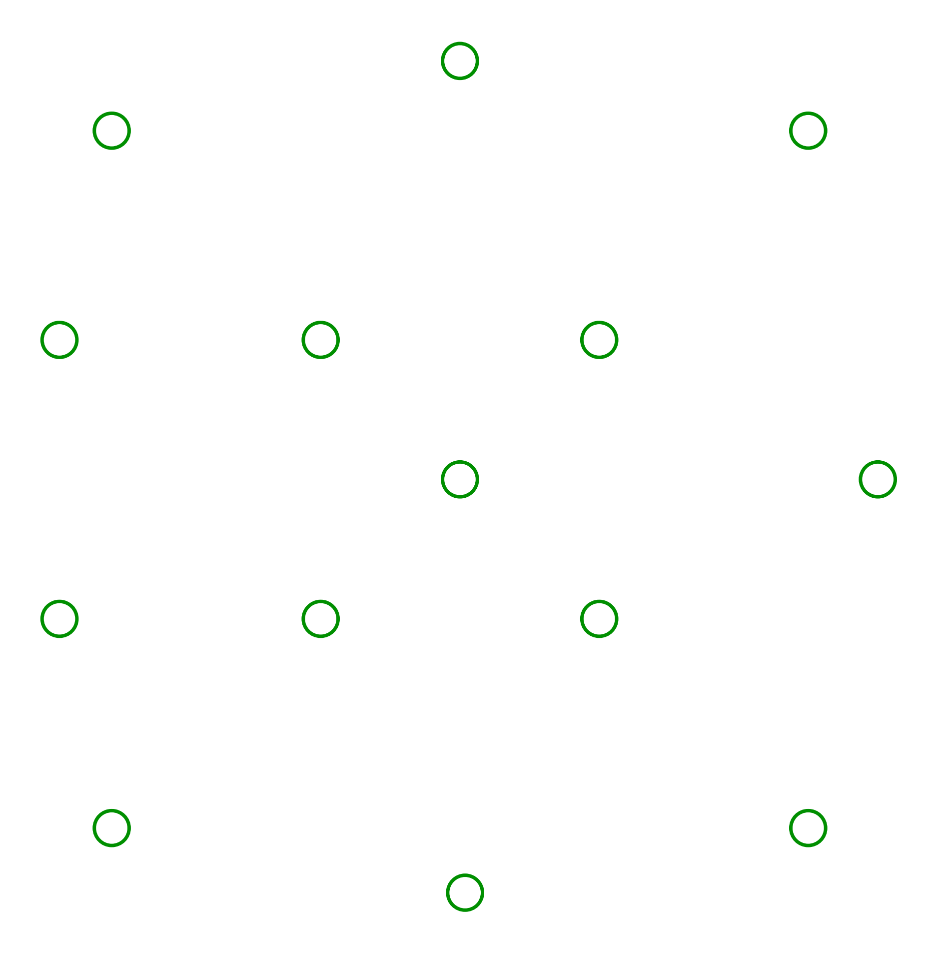
- Input: n points $(\alpha_i, y_i) \in \mathbb{F}_q^2$; agreement parameter t
- Output: **All** degree k polynomials $p(x)$ s.t. $p(\alpha_i) = y_i$ for at least t values of i.

We use $k = 1$ for illustration.

- i.e. want *all* "lines" ($y - ax - b = 0$) that pass through $\geq t$ out of n points.

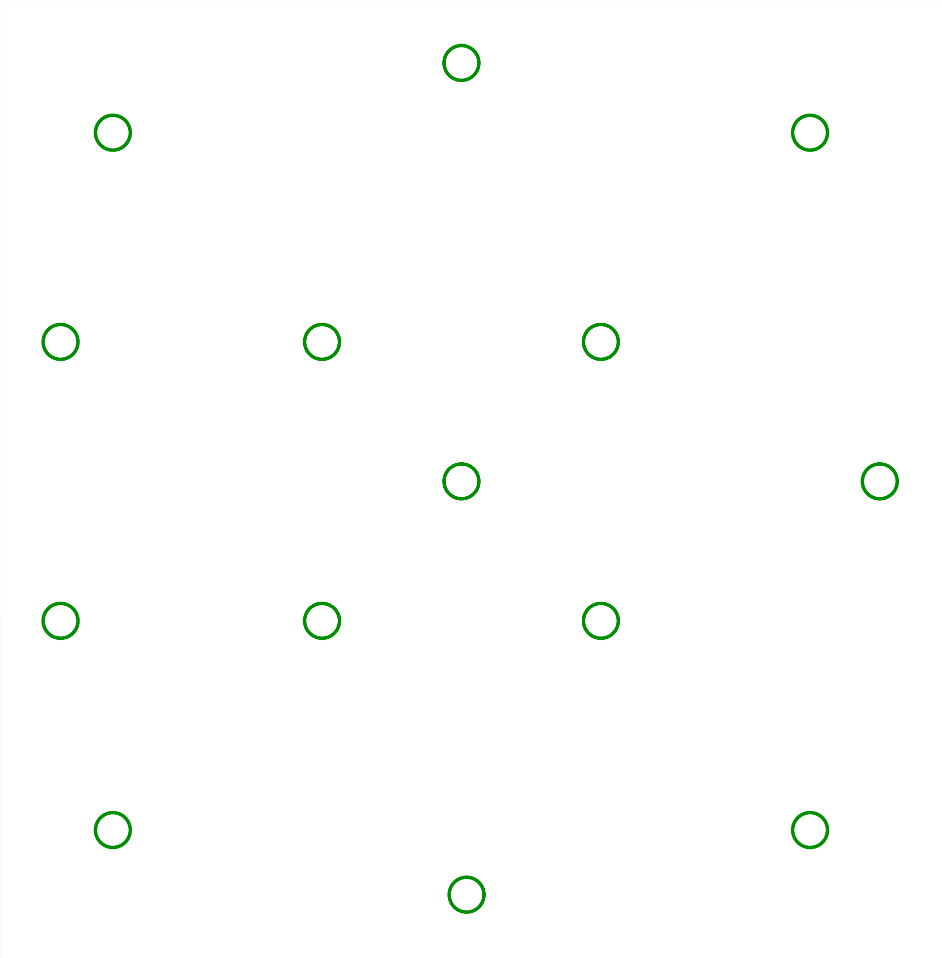
Algorithm Description [S. '96]

$n = \mathbf{14}$ points; Want all *lines* through at least **5** points.



Algorithm Description [S. '96]

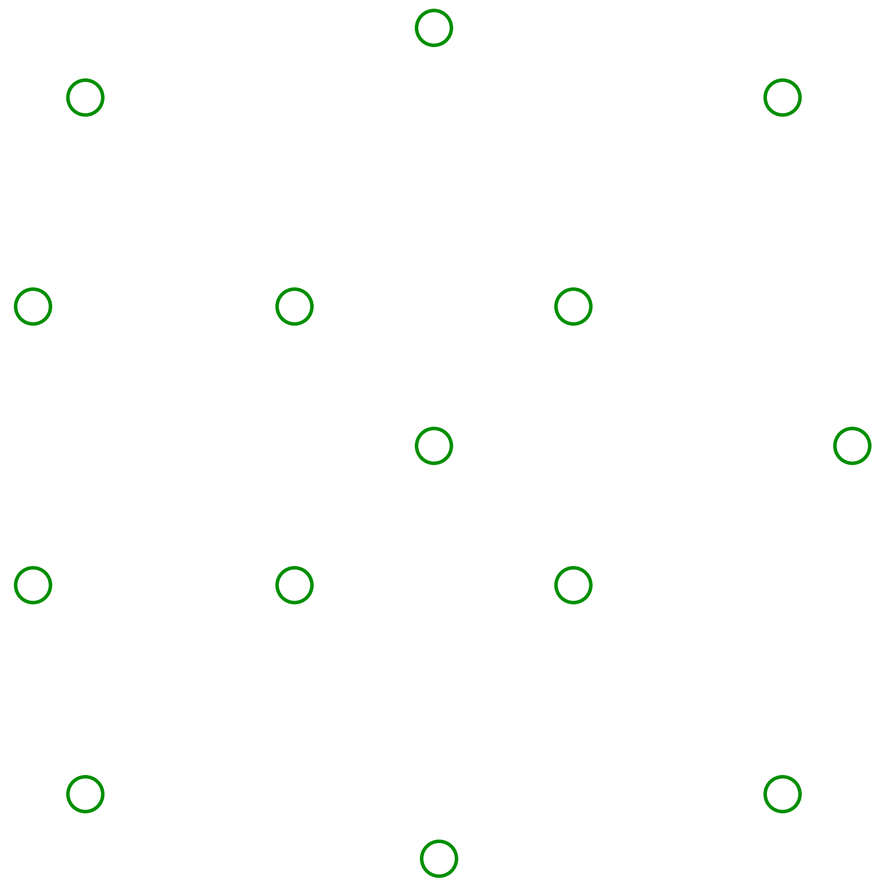
$n = \mathbf{14}$ points; Want all *lines* through at least **5** points.



Find deg. 4 poly. $Q(x, y) \not\equiv 0$
s.t. $Q(\alpha_i, y_i) = 0$ for all points.

Algorithm Description [S. '96]

$n = \mathbf{14}$ points; Want all *lines* through at least **5** points.



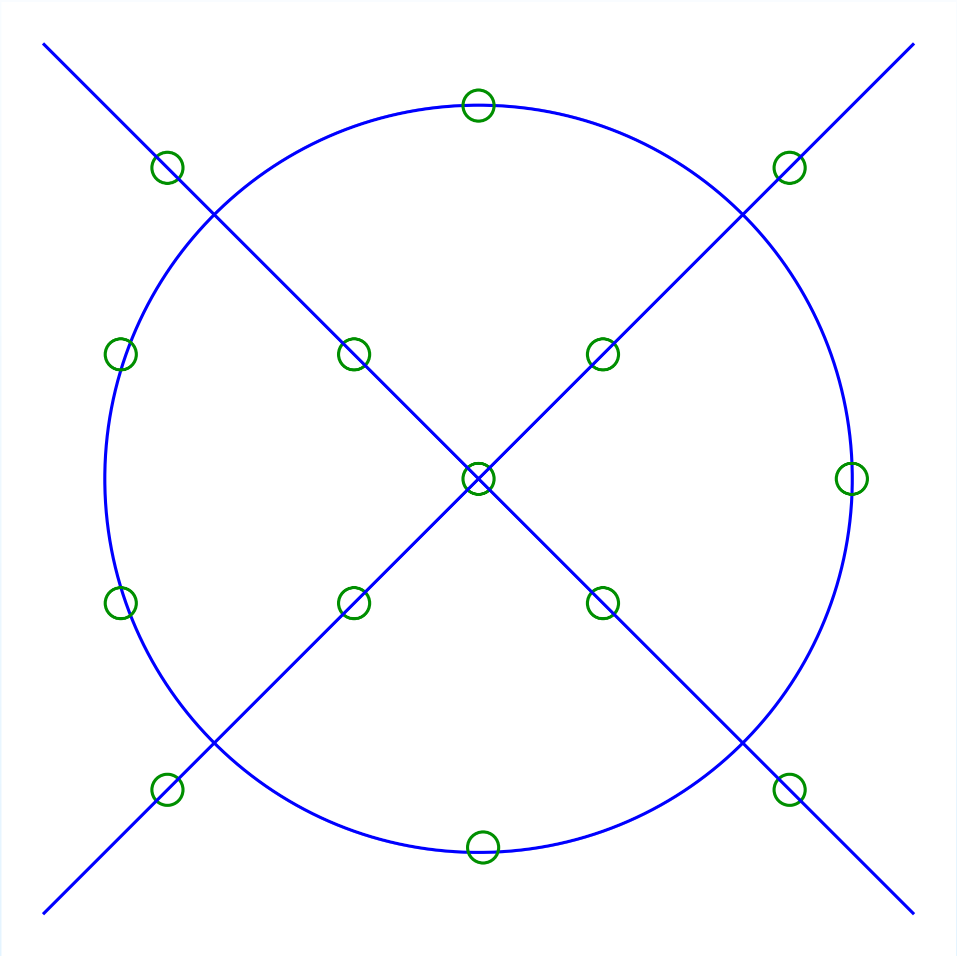
Find deg. 4 poly. $Q(x,y) \not\equiv 0$
s.t. $Q(\alpha_i, y_i) = 0$ for all points.

$Q(x,y) = y^4 - x^4 - y^2 + x^2$

Let us plot all zeroes of Q ...

Algorithm Description [S. '96]

$n = \mathbf{14}$ points; Want all *lines* through at least **5** points.



Find deg. 4 poly. $Q(x, y) \not\equiv 0$
s.t. $Q(\alpha_i, y_i) = 0$ for all points.

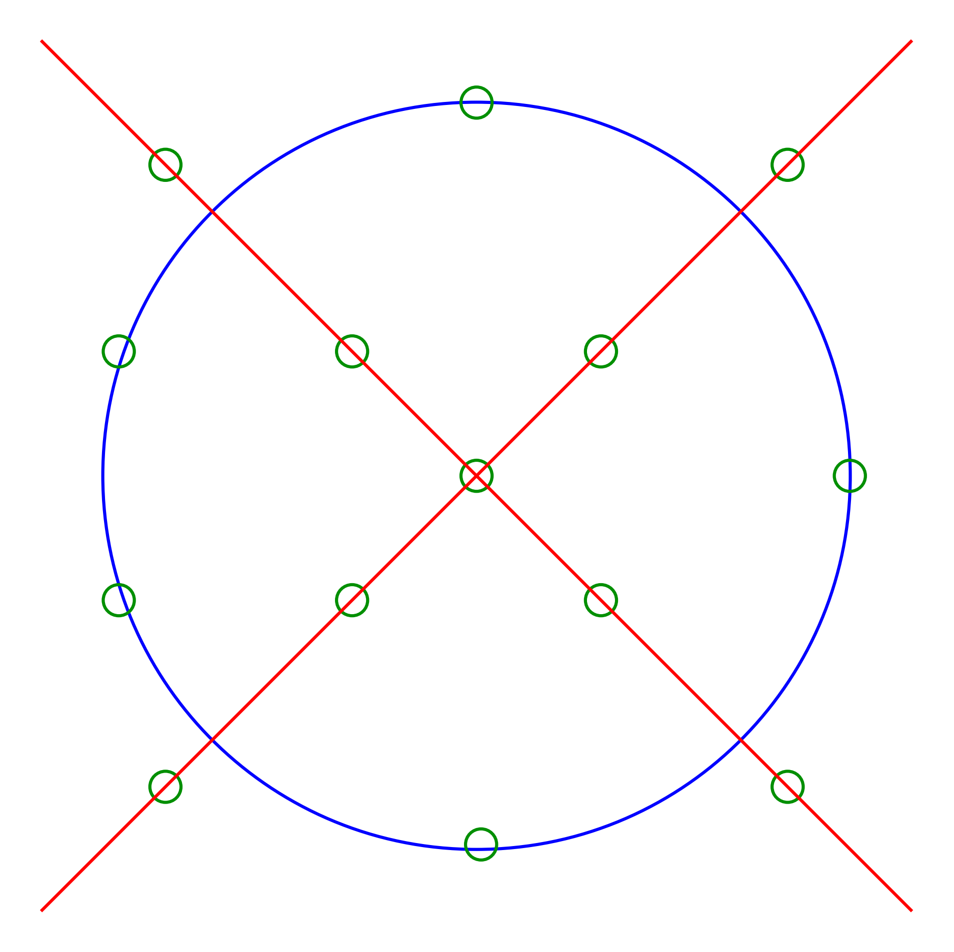
$Q(x, y) = y^4 - x^4 - y^2 + x^2$

Let us plot all zeroes of Q ...

Both relevant lines emerge !

Algorithm Description [S. '96]

$n = \mathbf{14}$ points; Want all *lines* through at least **5** points.



Find deg. 4 poly. $Q(x, y) \not\equiv 0$
s.t. $Q(\alpha_i, y_i) = 0$ for all points.

$Q(x, y) = y^4 - x^4 - y^2 + x^2$

Let us plot all zeroes of Q ...

Both relevant lines emerge !

Formally, $Q(x, y)$ factors as:
$(x^2 + y^2 - 1)(y + x)(y - x)$.

What Happened?

1. Why did degree 4 curve exist?
   - Counting argument: degree 4 gives enough degrees of freedom to pass through any 14 points.
2. Why did all the relevant lines emerge/factor out?
   - Line ℓ intersects a deg. 4 curve Q in 5 points $\implies \ell$ is a factor of Q

Generally

Lemma 1: $\exists Q$ with $\deg_x(Q), \deg_y(Q) \leq D = \sqrt{n}$ passing thru any n points.

Lemma 2: If Q with $\deg_x(Q), \deg_y(Q) \leq D$ intersects $y - p(x)$ with $\deg(p) \leq d$ intersect in more that $(D+1)d$ points, then $y - p(x)$ divides Q.

Efficient algorithm?

1. Can find Q by solving system of linear equations

Efficient algorithm?

1. Can find Q by solving system of linear equations
2. Fast algorithms for factorization of bivariate polynomials exist ('83-'85) [Kaltofen, Chistov & Grigoriev, Lenstra, von zur Gathen & Kaltofen]

Efficient algorithm?

1. Can find Q by solving system of linear equations
2. Fast algorithms for factorization of bivariate polynomials exist ('83-'85) [Kaltofen, Chistov & Grigoriev, Lenstra, von zur Gathen & Kaltofen]

- Immediate application:

Efficient algorithm?

1. Can find Q by solving system of linear equations
2. Fast algorithms for factorization of bivariate polynomials exist ('83-'85) [Kaltofen, Chistov & Grigoriev, Lenstra, von zur Gathen & Kaltofen]

- Immediate application:

  Theorem: Can list-decode Reed-Solomon code from $n-(k+1)\sqrt{n}$ errors.

Efficient algorithm?

1. Can find Q by solving system of linear equations
2. Fast algorithms for factorization of bivariate polynomials exist ('83-'85) [Kaltofen, Chistov & Grigoriev, Lenstra, von zur Gathen & Kaltofen]

- Immediate application:

  Theorem: Can list-decode Reed-Solomon code from $n-(k+1)\sqrt{n}$ errors.

- With some fine-tuning of parameters:

Efficient algorithm?

1. Can find Q by solving system of linear equations
2. Fast algorithms for factorization of bivariate polynomials exist ('83-'85) [Kaltofen, Chistov & Grigoriev, Lenstra, von zur Gathen & Kaltofen]

- Immediate application:

  Theorem: Can list-decode Reed-Solomon code from $n-(k+1)\sqrt{n}$ errors.

- With some fine-tuning of parameters:

  Theorem: [S. '96] Can list-decode Reed-Solomon code from $1-\sqrt{2R}$-fraction errors.

Efficient algorithm?

1. Can find Q by solving system of linear equations
2. Fast algorithms for factorization of bivariate polynomials exist ('83-'85) [Kaltofen, Chistov & Grigoriev, Lenstra, von zur Gathen & Kaltofen]

- Immediate application:

  Theorem: Can list-decode Reed-Solomon code from $n-(k+1)\sqrt{n}$ errors.

- With some fine-tuning of parameters:

  Theorem: [S. '96] Can list-decode Reed-Solomon code from $1-\sqrt{2R}$-fraction errors.

- Does not meet combinatorial bounds though!

Efficient algorithm?

1. Can find Q by solving system of linear equations
2. Fast algorithms for factorization of bivariate polynomials exist ('83-'85) [Kaltofen, Chistov & Grigoriev, Lenstra, von zur Gathen & Kaltofen]

- Immediate application:

  Theorem: Can list-decode Reed-Solomon code from $n-(k+1)\sqrt{n}$ errors.

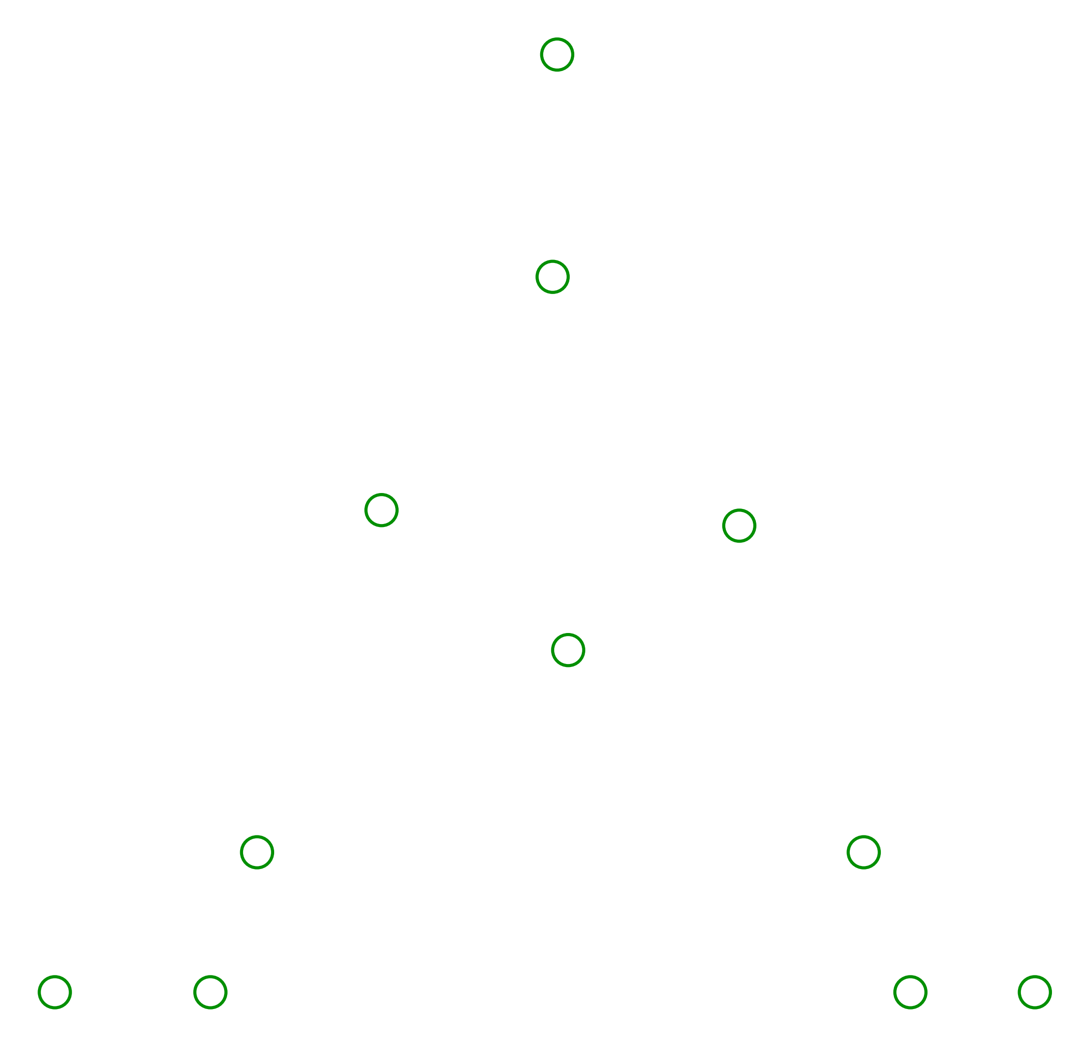
- With some fine-tuning of parameters:

  Theorem: [S. '96] Can list-decode Reed-Solomon code from $1-\sqrt{2R}$-fraction errors.

- Does not meet combinatorial bounds though!

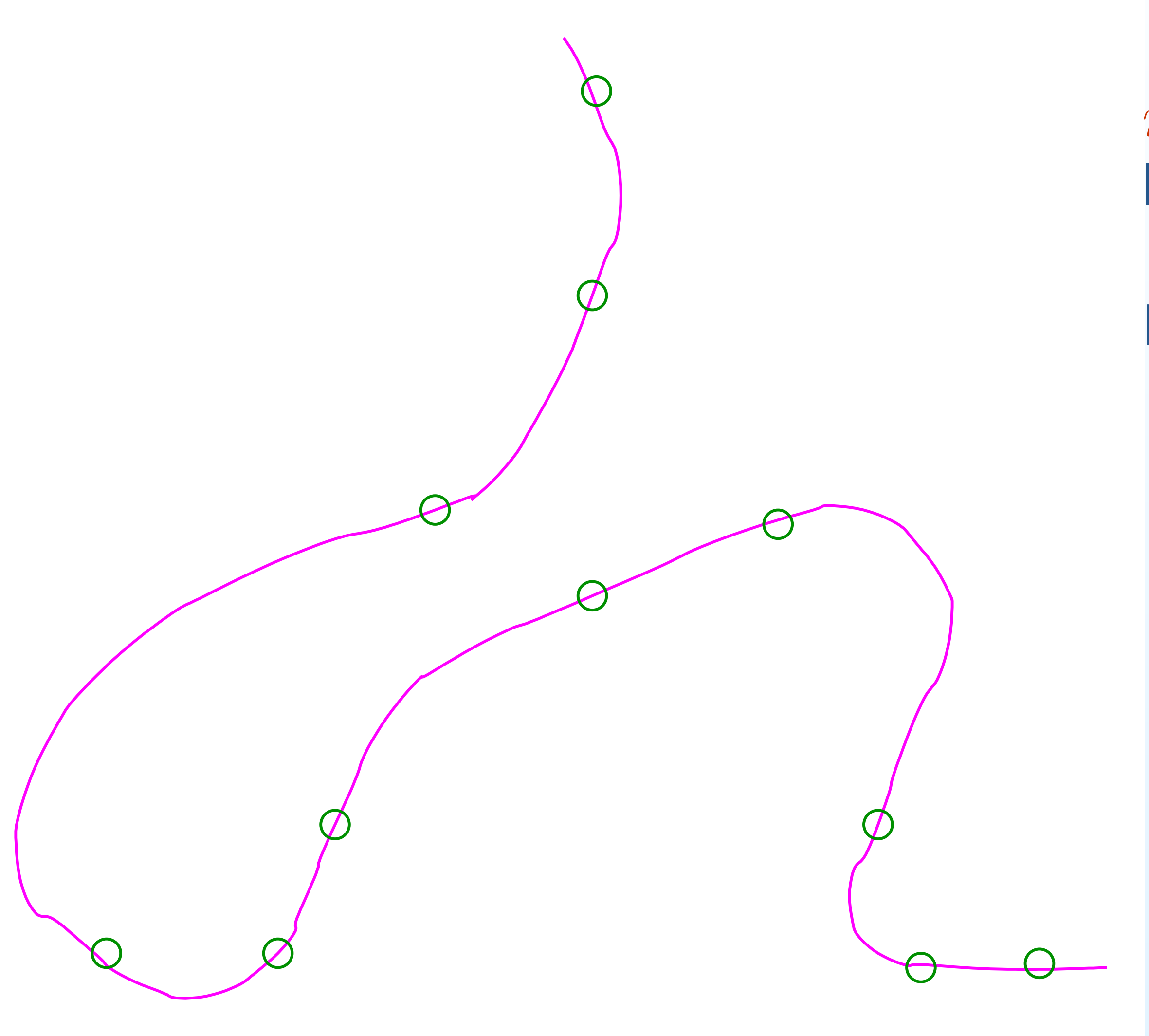
Part 6: Improved RS List-Decoding

Going Further: Example 2 [Guruswami+S. '98]



$n = 11$ points; Want all lines through ≥ 4 pts.

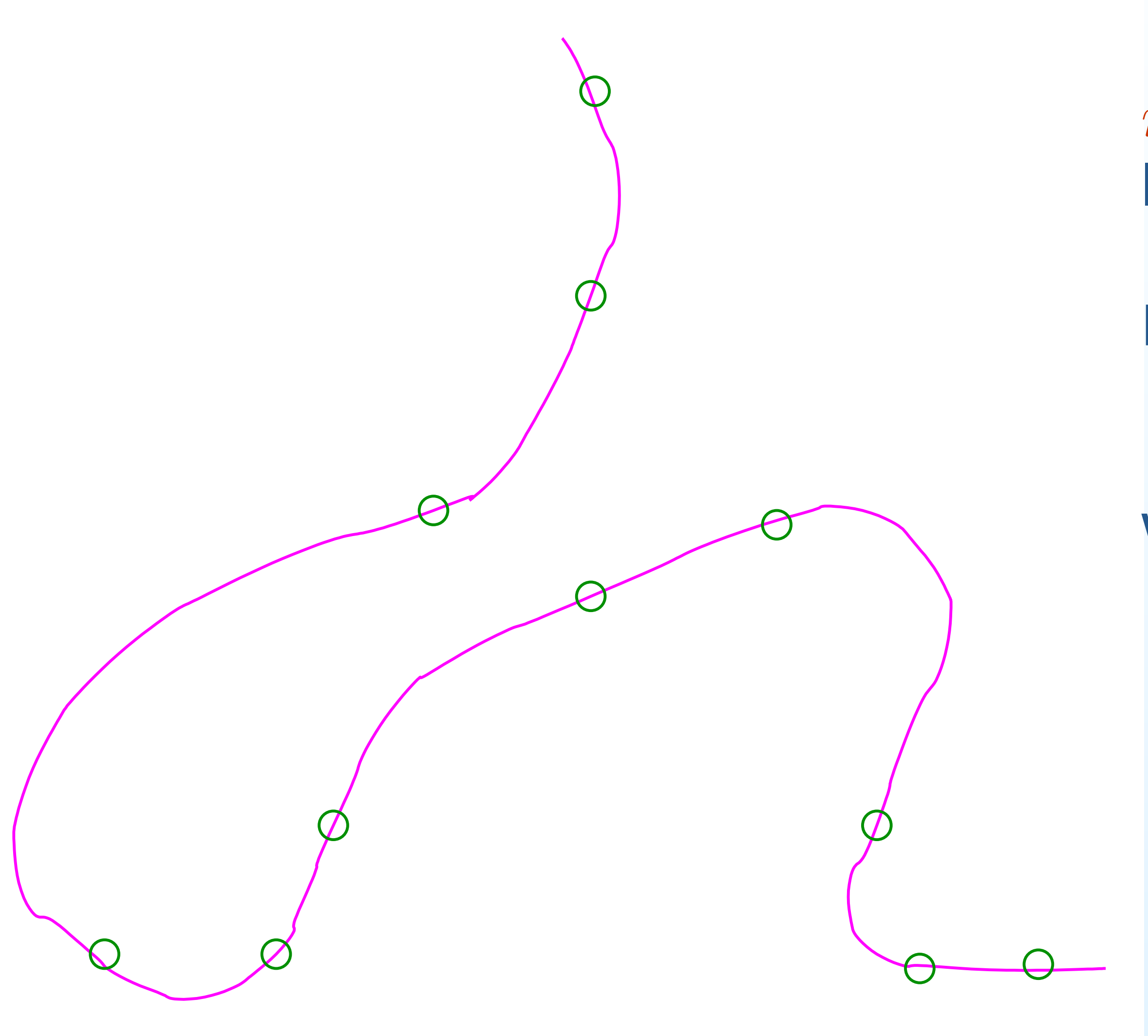
Going Further: Example 2 [Guruswami+S. '98]



$n = 11$ points; Want all lines through ≥ 4 pts.

Fitting degree 4 curve Q as earlier doesn't work.

Going Further: Example 2 [Guruswami+S. '98]

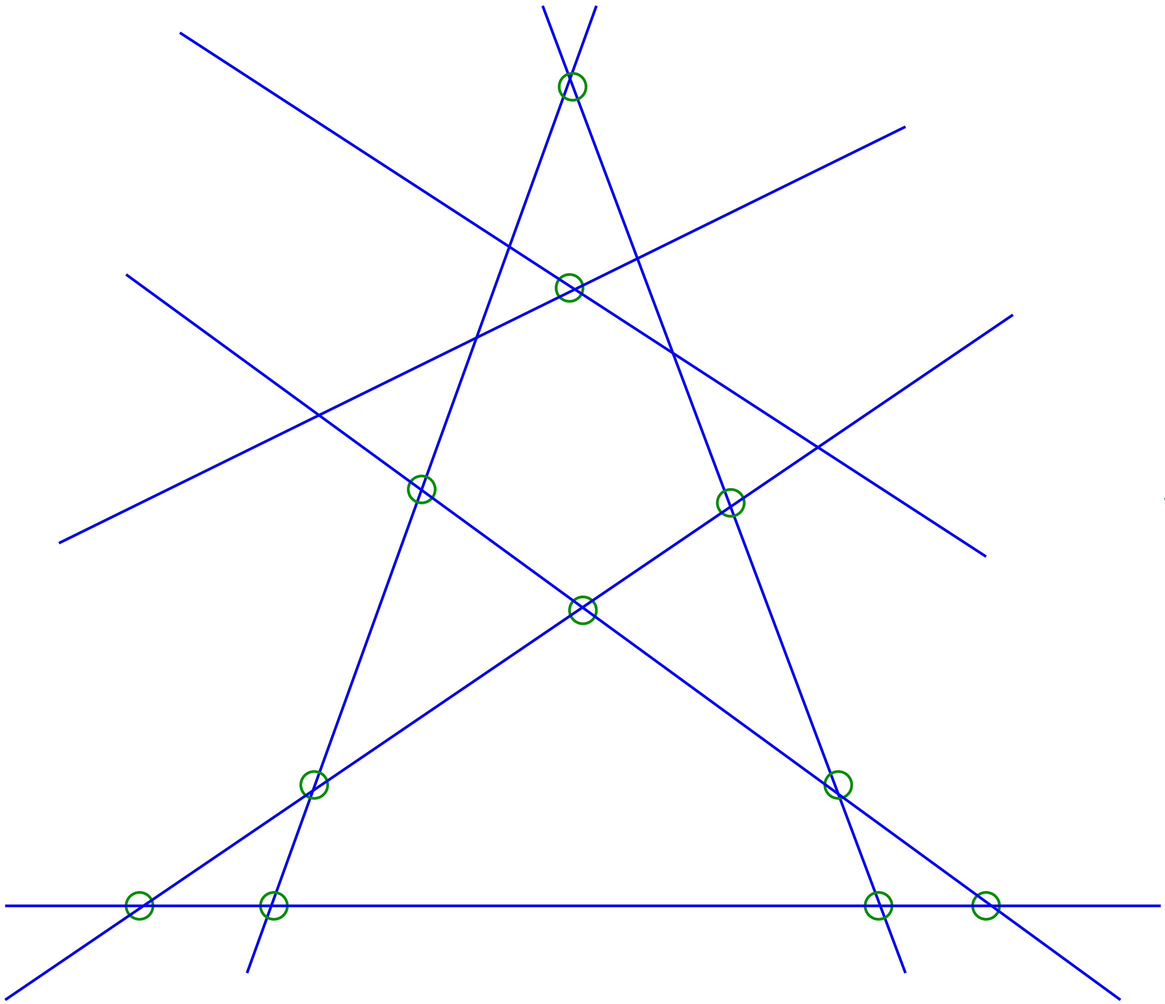


$n = 11$ points; Want <u>all</u> lines through ≥ 4 pts.

Fitting degree 4 curve Q as earlier doesn't work.

Why?

Going Further: Example 2 [Guruswami+S. '98]



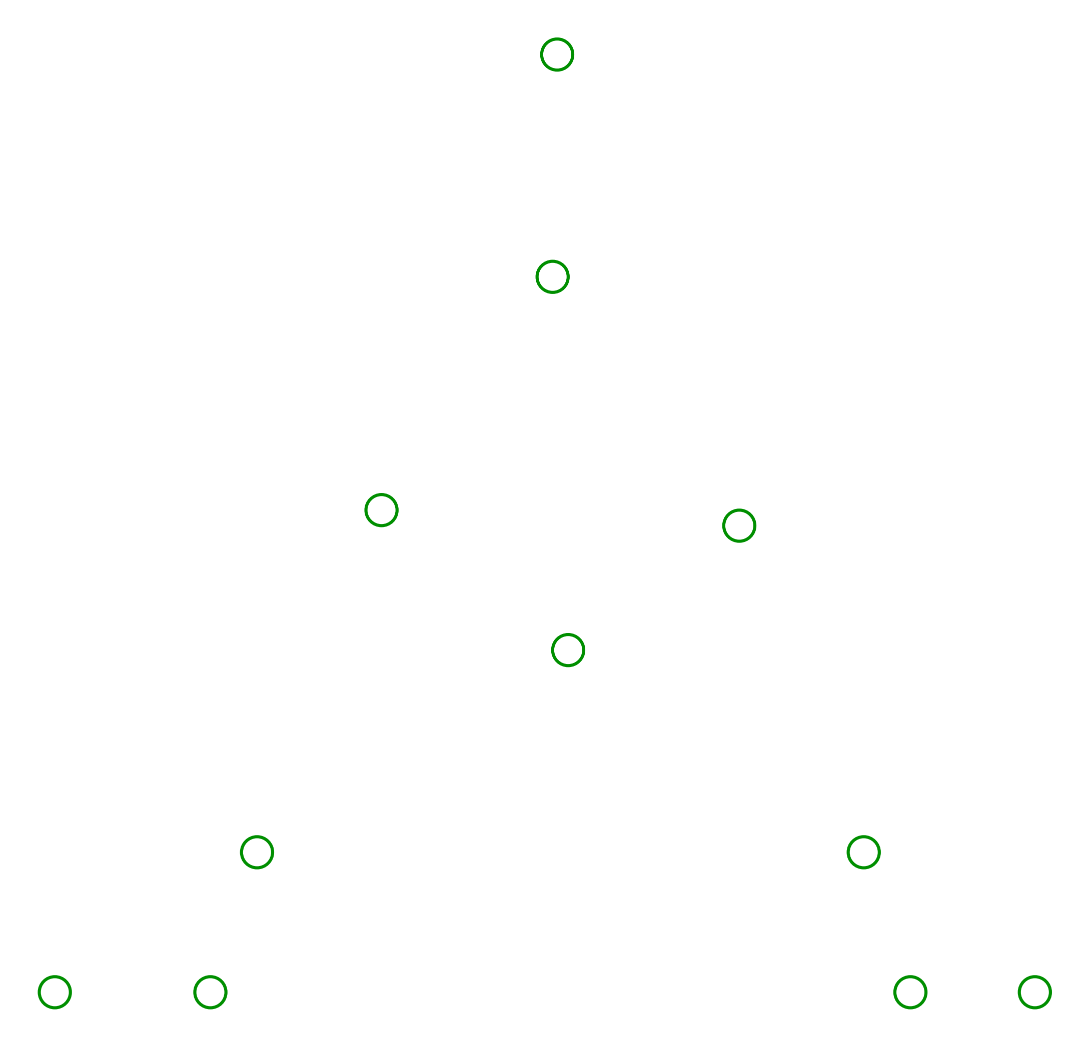
$n = 11$ points; Want <u>all</u> lines through ≥ 4 pts.

Fitting degree 4 curve Q as earlier doesn't work.

Why?

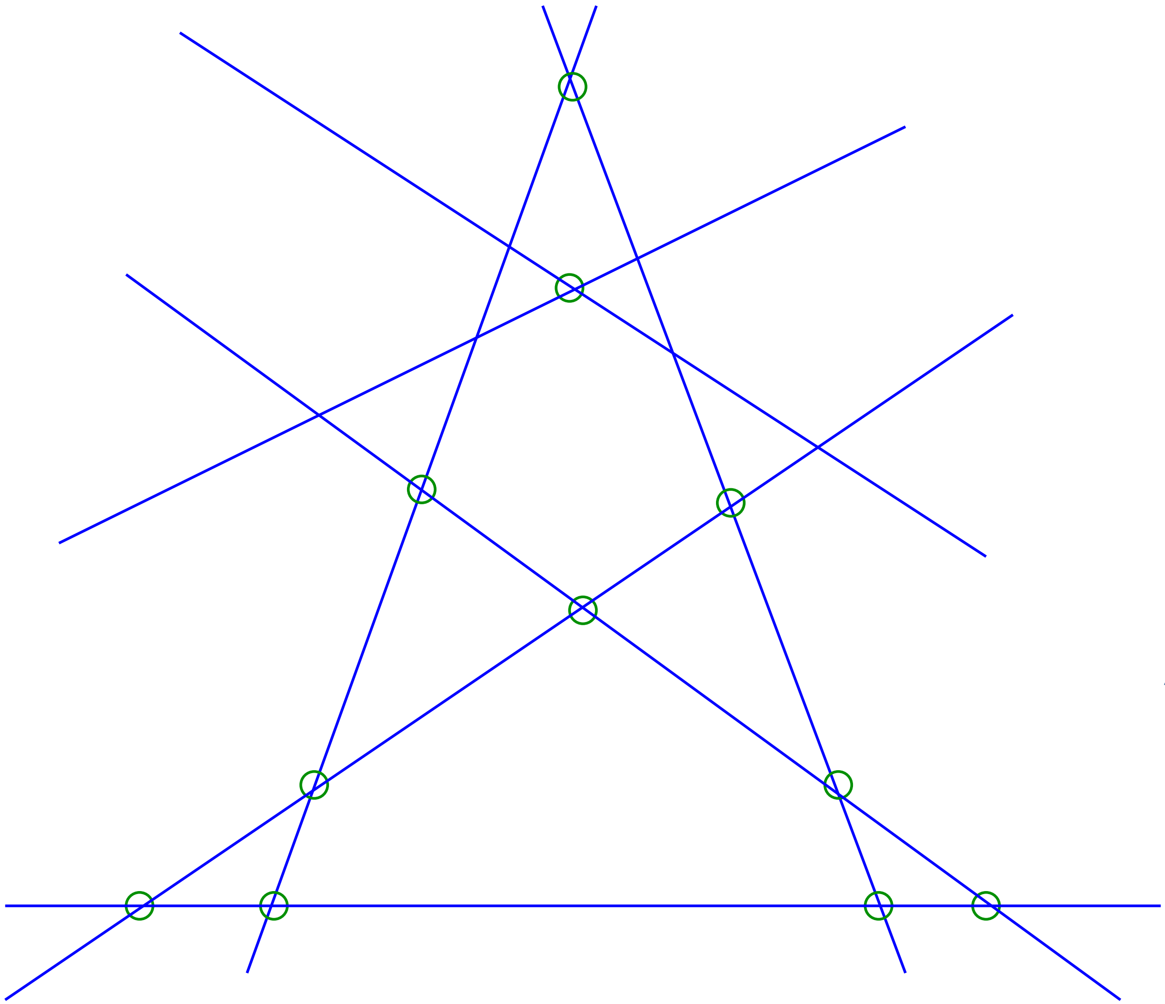
Correct answer has 5 lines. Degree 4 curve can't have 5 factors!

Going Further: Example 2 [Guruswami+S. '98]



$n = 11$ points; Want <u>all</u>
lines through ≥ 4 pts.
Fit degree 7 poly. $Q(x, y)$
passing through each
point <u>twice</u>.
$Q(x, y) = \cdots$
(margin too small)
Plot all zeroes ...

Going Further: Example 2 [Guruswami+S. '98]



$n = 11$ points; Want <u>all</u> lines through ≥ 4 pts.

Fit degree 7 poly. $Q(x, y)$ passing through each point <u>twice</u>.

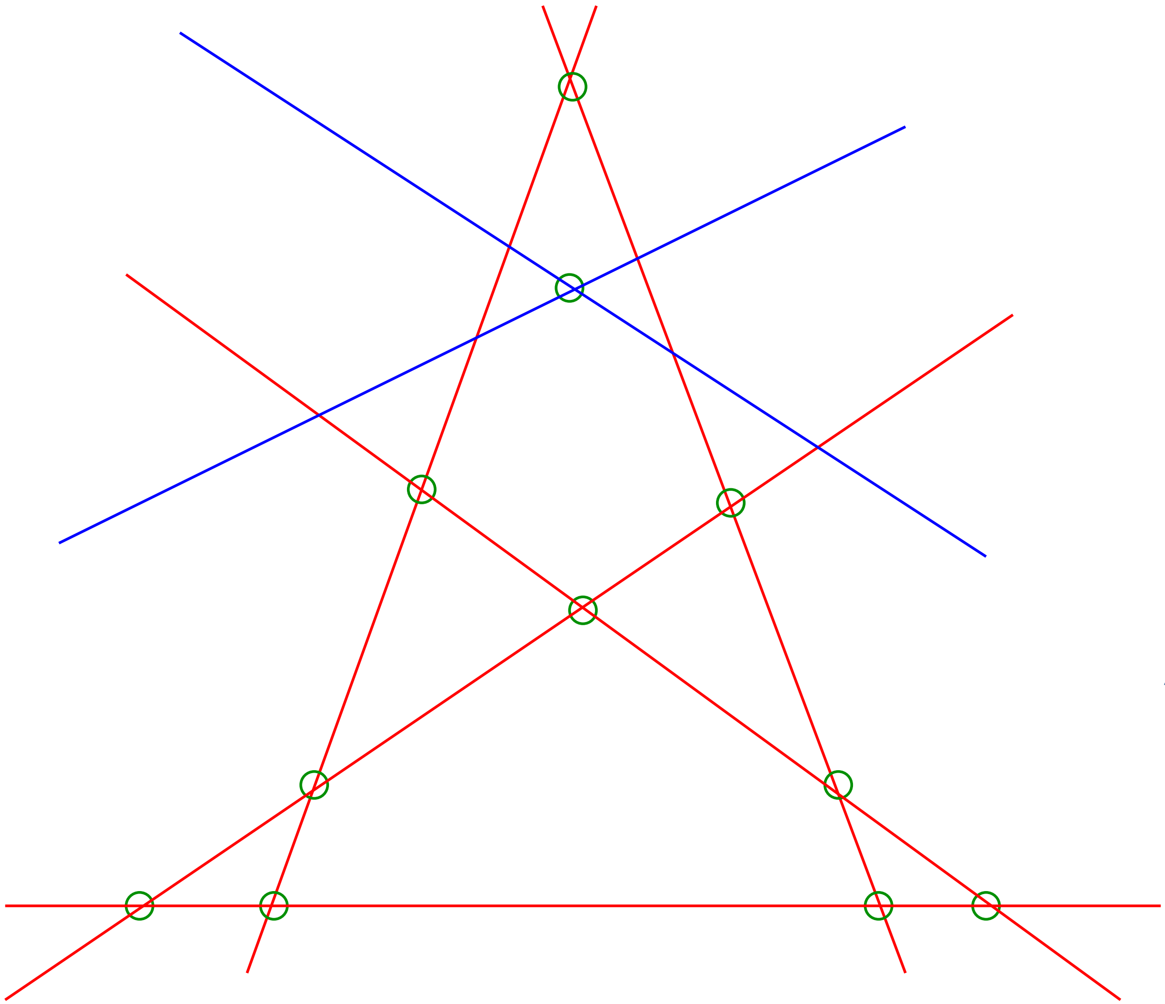
$Q(x, y) = \cdots$

(margin too small)

Plot all zeroes ...

All relevant lines emerge!

Going Further: Example 2 [Guruswami+S. '98]



$n = 11$ points; Want all lines through ≥ 4 pts.

Fit degree 7 poly. $Q(x, y)$ passing through each point twice.

$Q(x, y) = \cdots$

(margin too small)

Plot all zeroes ...

All relevant lines emerge!

Where was the gain?

- Requiring Q to pass through each point twice, effectively doubles the # intersections between Q and line.
  - So # intersections is now 8.
- On the other hand # constraints goes up from 11 to 33. Forces degree used to go upto 7 (from m4).
- But now # intersections is less than degree!

Can pass through each point twice with less than twice the degree!

- Letting intersection multiplicity go to ∞ gives decoding algorithm for upto $1 - \sqrt{R}$ errors.

Part 7: Rate-Optimal List-Decoding

Folded Reed-Solomon Codes and Decoding

A recent breakthrough

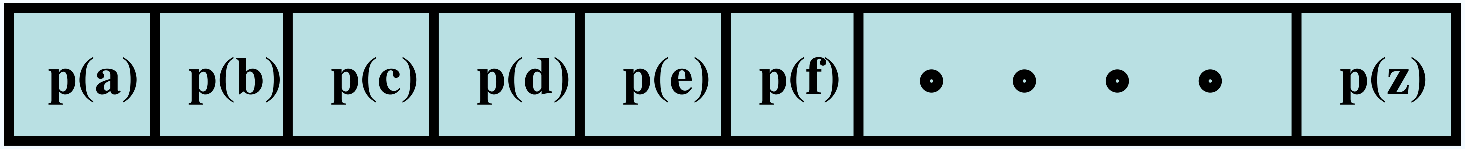
- State of the art in 2005:
  - Codes of positive rate with error correction rate close to upper limit.
  - But Rate only positive, not optimal.
  - E.g., Say disk has 5% “byte" error rate.
    - Rate with unique decoding = 90%.
    - Rate promised by list decoding = 95%.
    - Rate achieved algorithmically = 90.25%.

A recent breakthrough

- State of the art in 2005:
  - Codes of positive rate with error correction rate close to upper limit.
  - But Rate only positive, not optimal.
- Breakthrough: [ParvareshVardy 05, GuruswamiRudra 06].
- Codes of rate R correcting $1 - R - \epsilon$ fraction errors over alphabet of size $f(\epsilon)$ (for every $\epsilon > 0, 0 < R < 1$.)
- Key Ingredient: “Folded Reed-Solomon Codes” + Clever “Concatenation”.
- Analysis complicated (series of accidental discoveries).
- Yields optimal results over large alphabets.

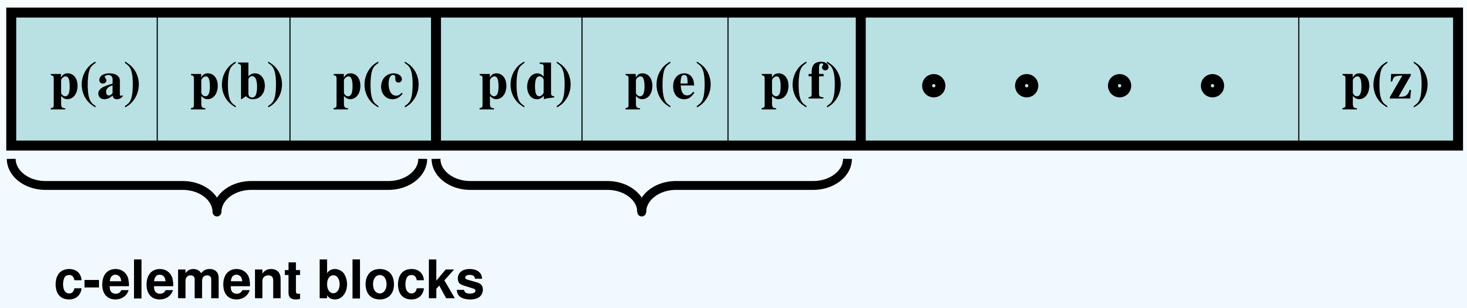
Folded Reed-Solomon Codes [GR06]

- Message: Univariate degree k polynomial $p \in \mathbb{F}_q[x]$.
- Encoding:



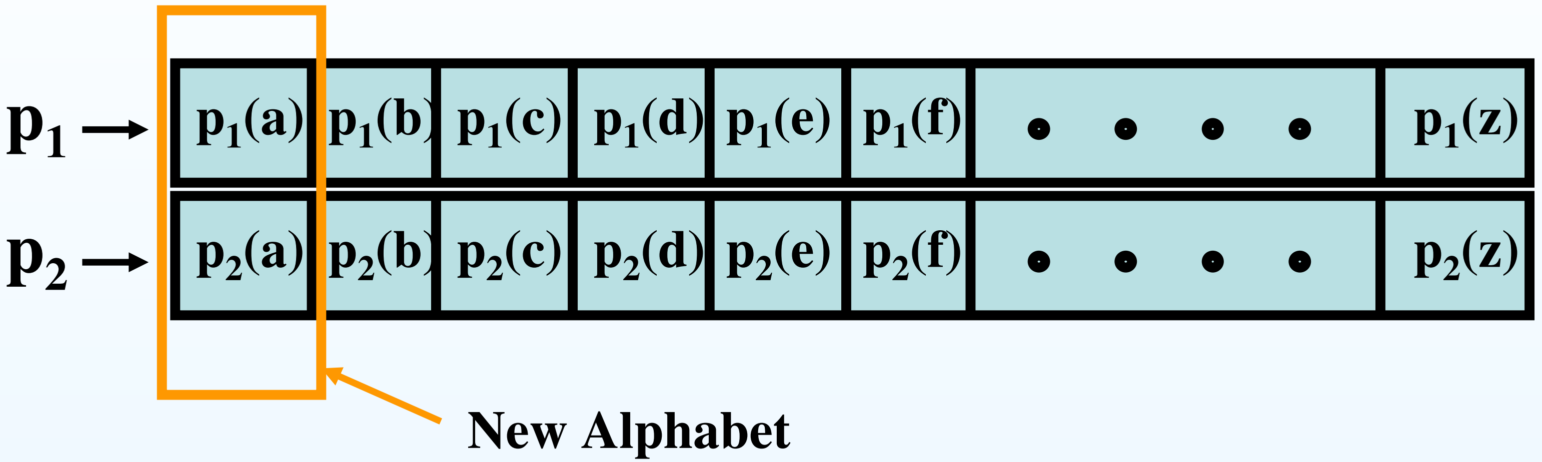
Folded Reed-Solomon Codes [GR06]

- Message: Univariate degree k polynomial $p \in \mathbb{F}_q[x]$.
- Encoding:



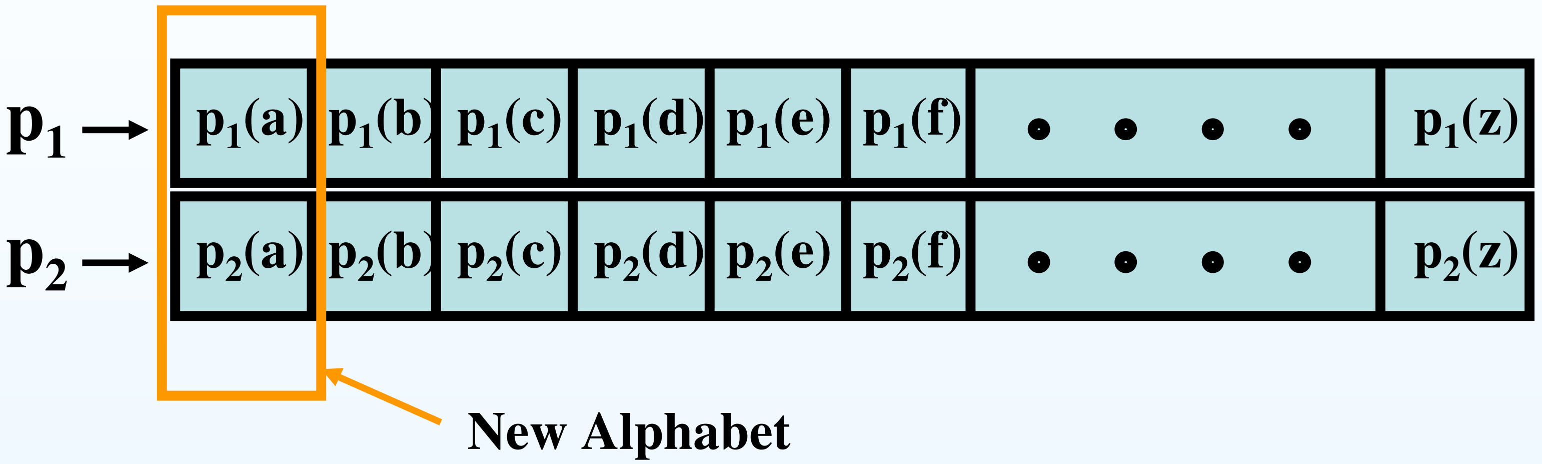
- $a = \omega, b = \omega^2, c = \omega^3 \ldots$.
- Defines Code mapping $\Sigma^{k/c} \to \Sigma^{n/c}$ for $\Sigma = \mathbb{F}_q^c$.
- Does the “blocking” really alter the code?
- Surprisingly . . . YES!

Interleaved Reed-Solomon Codes



- Introduced by [Kiayias-Yung] (accidentally)!
- Alphabet = $\mathbb{F}_q^2$.
- Message = $(p_1, p_2), p_i \in \mathbb{F}_q[x]$ of deg. $\leq k$.
- Encoding = $\langle (p_1(\alpha_i), p_2(alpha_i)) \rangle_i$.
- Rate, Distance, same as RS.

Interleaved Reed-Solomon Codes

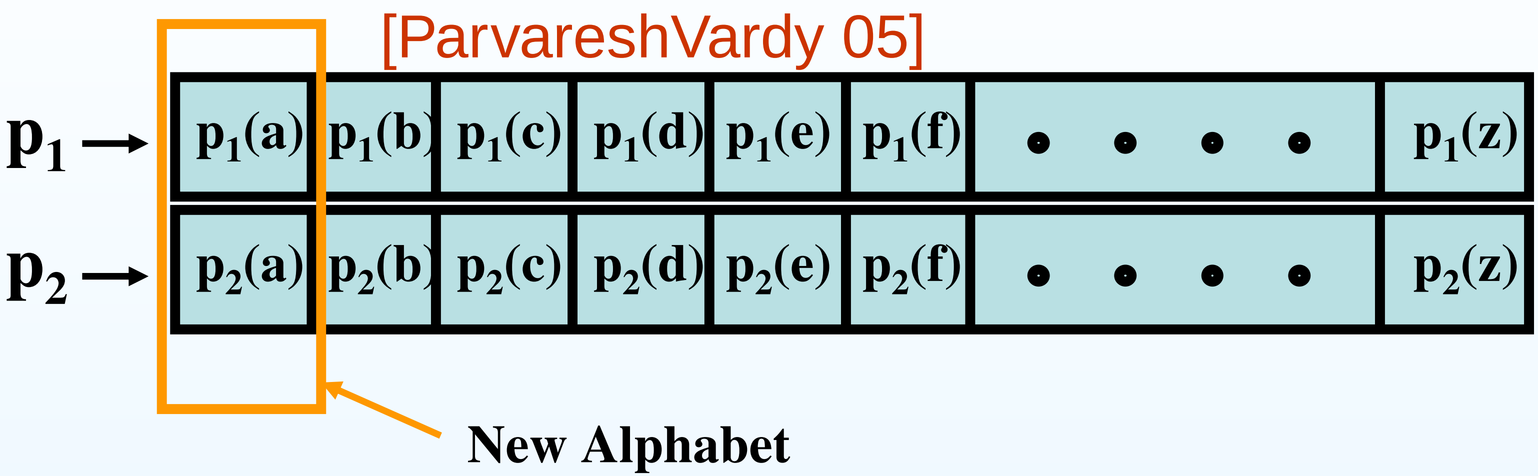


- Decoding [Coppersmith Sudan 03]:
- Looks like trivariate polynomial search.
- Find $Q(x, y, z)$ s.t. $Q(\alpha_i, \beta_i, \gamma_i) = 0$ (of high multiplicity) for every $i \in [n]$.
- Degree of $Q \sim n^{1/3}$.

✔ Roughly corrects $1 - R^{2/3}$ random errors.

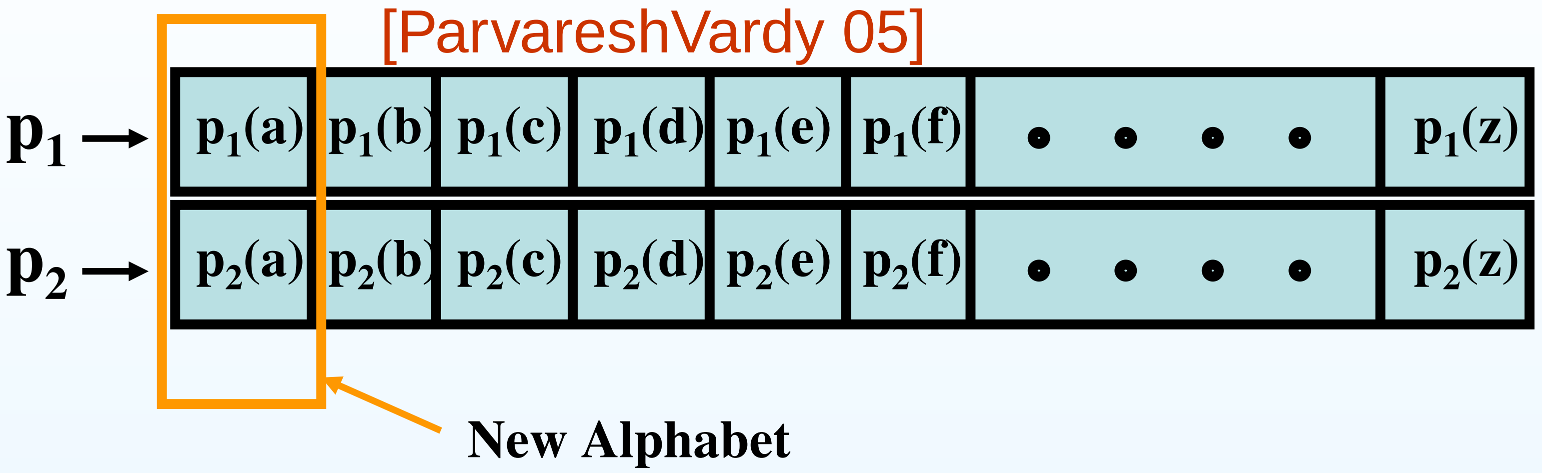
✗ Only *random* errors.

Interleaved Reed-Solomon Codes



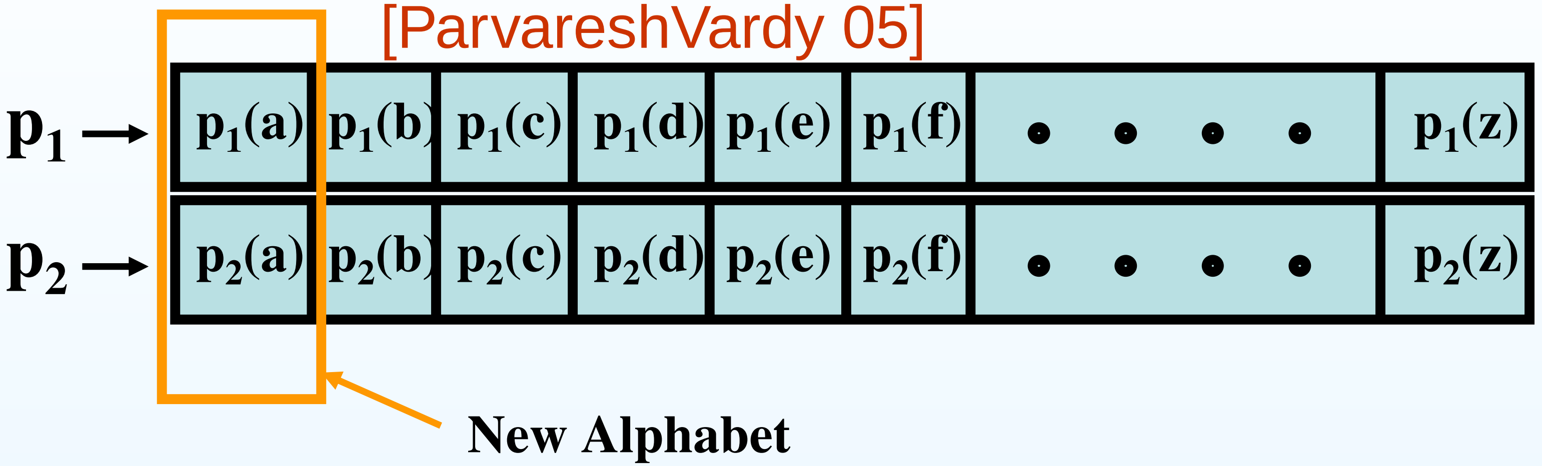
- Problem with naive “interleaving”/decoding:
  - Only information in Q is that it lies in the ideal $(y - p_1(x), z - p_2(x))$.
  - So Q gives a curve in $F_q[x] \times \mathbb{F}_q[x]$ that passes (p_1, p_2).
  - Certainly this is hopelessly little info about (p_1, p_2) to pin them down!
- Hopeless?

Related Interleaved Reed-Solomon Codes



- Since Q only give *one* curve through (p_1, p_2), lets force them to lie on a different curve by design!
- Message: $p_1 \in \mathbb{F}_q[x]$
- Encoding: Compute $p_2 = p_1^D \pmod{h(x)}$ and use interleaved encoding of (p_1, p_2).

Related Interleaved Reed-Solomon Codes



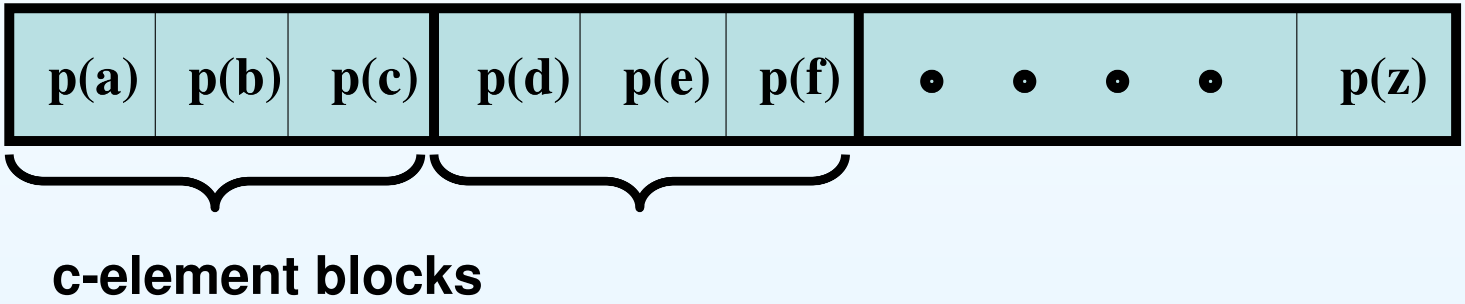
- $p_1 \longrightarrow (p_1, p_1^D (\mathrm{mod}\ h(x))) \longrightarrow$ Interleaved encoding.
- Decoding:
  - Effectively working in $\mathbb{F}_q(x)(\mathrm{mod}\ h(x))$.
  - $Q_x(y, z)$ and $z = y^D$ give enough information to pin down p_1, p_2.

✔ Decoding now really works. ✗Rate halves!

- Get codes of rate R decodable from $1 - (2R)^{2/3}$ error.

Folded Reed-Solomon Codes

- If we pick $h(x) = x^{q-1} - \omega$ and $D = q$, then $p_1(x)^D = p_1(x^D) = p_1(\omega x)$.
- c-folded RS code has rate k/n but manages to convey $(c-1)/c$ fraction of PV code of rate $k/(2n)$.



- Gives code of rate R correcting $1 - ((c/c-1)R)^{2/3})$-fraction errors.
- Letting $2 \to \infty$ and $c \to \infty$, corrects $1 - R - \epsilon$ fraction errors.

Conclusions

- Algebra plays a fundamental role in the combinatorics, algorithmics, and practice of Error-correction.
- Algebraic algorithms solve some very non-trivial search problems!
- Lead to first codes correcting maximal fraction of errors $(1 - R)$ of any given rate R, over large alphabet.
- Major open problem: Build binary codes of rate $1 - H(p)$ list-decodable from p fraction errors.
    - Will algebra over finite fields play a role?

Thank You !!