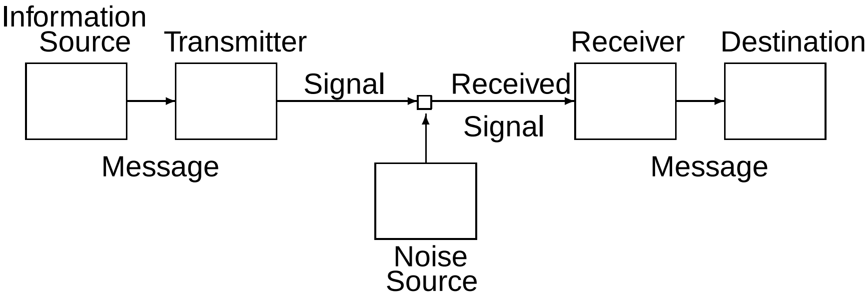
Decoding Algorithms for Reed-Solomon Codes

Annie Cervin

Department of Mathematics and Computer Science
College of the Holy Cross

Advisor: John Little

Introduced by Claude Shannon in 1948, coding theory tries to eliminate errors in transmitted messages.



Coding theory involves the study of both encoding and decoding.

- The encoding algorithm incorporates redundancy into a message.

Coding theory involves the study of both encoding and decoding.

- The encoding algorithm incorporates redundancy into a message.
- The message is transmitted.

Coding theory involves the study of both encoding and decoding.

- The encoding algorithm incorporates redundancy into a message.
- The message is transmitted.
- The decoding algorithm analyzes the received word and uses the redundancy to find the possibilities for the original message.

Block Codes

- Uncoded messages are divided into words with fixed length k.

Block Codes

- Uncoded messages are divided into words with fixed length k.
- The words are made from an alphabet of q symbols.

Block Codes

- Uncoded messages are divided into words with fixed length k.
- The words are made from an alphabet of q symbols.
- Each alphabet of q symbols corresponds to a finite field $\mathbb{F}_q$.

Block Codes

- Uncoded messages are divided into words with fixed length k.
- The words are made from an alphabet of q symbols.
- Each alphabet of q symbols corresponds to a finite field $\mathbb{F}_q$.
- The possible words in a message can be thought of as k-tuples of elements of $\mathbb{F}_q$. The collection of words is identified as $\mathbb{F}_q^k$.

Block Codes

- Uncoded messages are divided into words with fixed length k.
- The words are made from an alphabet of q symbols.
- Each alphabet of q symbols corresponds to a finite field $\mathbb{F}_q$.
- The possible words in a message can be thought of as k-tuples of elements of $\mathbb{F}_q$. The collection of words is identified as $\mathbb{F}_q^k$.
- Pick an integer $n > k$.

Block Codes

- Uncoded messages are divided into words with fixed length k.
- The words are made from an alphabet of q symbols.
- Each alphabet of q symbols corresponds to a finite field $\mathbb{F}_q$.
- The possible words in a message can be thought of as k-tuples of elements of $\mathbb{F}_q$. The collection of words is identified as $\mathbb{F}_q^k$.
- Pick an integer $n > k$.
- Each message will consist of blocks of n-tuples.

Encoding and Decoding Operations

- The encoding operation can be described as $E : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^n$.

Encoding and Decoding Operations

- The encoding operation can be described as $E : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^n$.
- The decoding operation is $D : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^k \cup \{fail\}$.

Encoding and Decoding Operations

- The encoding operation can be described as $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$.
- The decoding operation is $D : \mathbb{F}_q^n \to \mathbb{F}_q^k \cup \{fail\}$.
- The set of all codewords is $C = Im(E)$.

Encoding and Decoding Operations

- The encoding operation can be described as $E : \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n$.
- The decoding operation is $D : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^k \cup \{fail\}$.
- The set of all codewords is $C = Im(E)$.
- When a codeword x is transmitted with an error, x is replaced by $v = x + e$.

Encoding and Decoding Operations

- The encoding operation can be described as $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$.
- The decoding operation is $D : \mathbb{F}_q^n \to \mathbb{F}_q^k \cup \{fail\}$.
- The set of all codewords is $C = Im(E)$.
- When a codeword x is transmitted with an error, x is replaced by $v = x + e$.
- The error vector is $e \in \mathbb{F}_q^n$ and v is the received word.

Hamming Distance

Definition

The Hamming weight of a word u, written as $\mathrm{wt}(u)$, is the number of non-zero entries in u. The Hamming distance between two words u and v, written as $d(u, v)$, is the number of entries in which they differ.

- Working over the finite field $\mathbb{F}_7$, let u=(3, 1, 3, 5, 0, 6, 5) and $v = (3, 1, 2, 4, 0, 2, 0)$.

Hamming Distance

Definition

The Hamming weight of a word u, written as $\mathrm{wt}(u)$*, is the number of non-zero entries in u. The Hamming distance between two words u and v, written as* $d(u, v)$*, is the number of entries in which they differ.*

- Working over the finite field $\mathbb{F}_7$, let u=(3, 1, 3, 5, 0, 6, 5) and $v = (3, 1, 2, 4, 0, 2, 0)$.
- $u - v = (0, 0, 1, 1, 0, 4, 5)$.

Hamming Distance

Definition

The Hamming weight of a word u, written as $\mathrm{wt}(u)$*, is the number of non-zero entries in u. The Hamming distance between two words u and v, written as* $d(u, v)$*, is the number of entries in which they differ.*

- Working over the finite field $\mathbb{F}_7$, let u=(3, 1, 3, 5, 0, 6, 5) and $v = (3, 1, 2, 4, 0, 2, 0)$.
- $u - v = (0, 0, 1, 1, 0, 4, 5)$.
- Then $\mathrm{wt}(u - v)$ is 4 and the Hamming distance between u and v is 4.

Definition

The minimum distance d of a code C is the smallest Hamming distance between distinct codewords of C.

- Let C have the following codewords: $(0,0,0),(1,1,0),(0,1,1),(1,0,1)$.

Definition

The minimum distance d of a code C is the smallest Hamming distance between distinct codewords of C.

- Let C have the following codewords: $(0,0,0),(1,1,0),(0,1,1),(1,0,1)$.
- Each differs from the other in at least two places.

Definition

The minimum distance d of a code C is the smallest Hamming distance between distinct codewords of C.

- Let C have the following codewords: $(0,0,0),(1,1,0),(0,1,1),(1,0,1)$.
- Each differs from the other in at least two places.
- Thus, the minimum distance of C is 2.

Detection and Correction of Errors

Theorem

Let C be a code. Then errors of weight $\leq \delta$ in the received words can be detected if and only if the minimum distance $d \geq \delta + 1$.

Theorem

Errors of weight $\leq \delta$ can be corrected by nearest neighbor decoding if $d \geq 2\delta + 1$.

Introduction

- Presented in a paper by Irving Reed and Gustave Solomon in 1960.

Introduction

- Presented in a paper by Irving Reed and Gustave Solomon in 1960.
- They are used in Compact Disc players and space communication.

Introduction

- Presented in a paper by Irving Reed and Gustave Solomon in 1960.
- They are used in Compact Disc players and space communication.
- Based on finite fields or Galois Fields.

Introduction

- Presented in a paper by Irving Reed and Gustave Solomon in 1960.
- They are used in Compact Disc players and space communication.
- Based on finite fields or Galois Fields.
- Reed-Solomon codes are linear codes.

Definition

A linear code of length n over the field $\mathbb{F}_q$ is a vector subspace of $\mathbb{F}_q^n$.

The codes achieve the Singleton bound over a fixed finite field.

Theorem

The Singleton bound requires that for any code $C \subset \mathbb{F}_q^n$ with q^k codewords and minimum distance d,

$$k \leq n - d + 1.$$

Creating Reed-Solomon Codes

- We fix $n = q - 1$, an integer $k \leq q$, and all polynomials with degree $\leq k - 1$ over $\mathbb{F}_q$.

Creating Reed-Solomon Codes

- We fix $n = q - 1$, an integer $k \leq q$, and all polynomials with degree $\leq k - 1$ over $\mathbb{F}_q$.
- Each codeword is made by evaluating one of these polynomials with coefficients in $\mathbb{F}_q$ at the nonzero elements of $\mathbb{F}_q$.

Creating Reed-Solomon Codes

- We fix $n = q - 1$, an integer $k \leq q$, and all polynomials with degree $\leq k - 1$ over $\mathbb{F}_q$.
- Each codeword is made by evaluating one of these polynomials with coefficients in $\mathbb{F}_q$ at the nonzero elements of $\mathbb{F}_q$.
- The nonzero elements can be written in terms of a primitive element for $\mathbb{F}_q$, α, and are $1, \alpha, \ldots, \alpha^{q-2}$.

Creating Reed-Solomon Codes

- We fix $n = q - 1$, an integer $k \leq q$, and all polynomials with degree $\leq k - 1$ over $\mathbb{F}_q$.
- Each codeword is made by evaluating one of these polynomials with coefficients in $\mathbb{F}_q$ at the nonzero elements of $\mathbb{F}_q$.
- The nonzero elements can be written in terms of a primitive element for $\mathbb{F}_q$, α, and are $1, \alpha, \ldots, \alpha^{q-2}$.
- Let L_k be the $\mathbb{F}_q$ vector space of polynomials of degree $< k$ with coefficients in $\mathbb{F}_q$.

Creating Reed-Solomon Codes

- We fix $n = q - 1$, an integer $k \leq q$, and all polynomials with degree $\leq k - 1$ over $\mathbb{F}_q$.
- Each codeword is made by evaluating one of these polynomials with coefficients in $\mathbb{F}_q$ at the nonzero elements of $\mathbb{F}_q$.
- The nonzero elements can be written in terms of a primitive element for $\mathbb{F}_q$, α, and are $1, \alpha, \ldots, \alpha^{q-2}$.
- Let L_k be the $\mathbb{F}_q$ vector space of polynomials of degree $< k$ with coefficients in $\mathbb{F}_q$.
- The linear evaluation mapping can be written as:

$$\begin{array}{rcl} \omega : L_k & \rightarrow & \mathbb{F}_q^{q-1} \\ f & \mapsto & (f(1), f(\alpha), \ldots, f(\alpha^{q-2})). \end{array}$$

Creating Reed-Solomon Codes

- We fix $n = q - 1$, an integer $k \leq q$, and all polynomials with degree $\leq k - 1$ over $\mathbb{F}_q$.
- Each codeword is made by evaluating one of these polynomials with coefficients in $\mathbb{F}_q$ at the nonzero elements of $\mathbb{F}_q$.
- The nonzero elements can be written in terms of a primitive element for $\mathbb{F}_q$, α, and are $1, \alpha, \ldots, \alpha^{q-2}$.
- Let L_k be the $\mathbb{F}_q$ vector space of polynomials of degree $< k$ with coefficients in $\mathbb{F}_q$.
- The linear evaluation mapping can be written as:

$$\begin{aligned} \omega : L_k &\to \mathbb{F}_q^{q-1} \\ f &\mapsto (f(1), f(\alpha), \ldots, f(\alpha^{q-2})). \end{aligned}$$

- $\mathrm{Im}(L_k)$ is denoted $RS(k, q)$.

Minimum Distance

- The minimum distance of $RS(k, q)$ is $d = q - k$.

Minimum Distance

- The minimum distance of $RS(k, q)$ is $d = q - k$.
- The Singleton bound is achieved for $RS(k, q)$ because $k = n - d + 1 = (q - 1) - (q - k) + 1 = k$.

Minimum Distance

- The minimum distance of $RS(k,q)$ is $d = q - k$.
- The Singleton bound is achieved for $RS(k,q)$ because $k = n - d + 1 = (q-1) - (q-k) + 1 = k$.
- Every $RS(k,q)$ achieves the largest possible code minimum distance for this specific block length n.

Minimum Distance

- The minimum distance of $RS(k, q)$ is $d = q - k$.
- The Singleton bound is achieved for $RS(k, q)$ because $k = n - d + 1 = (q - 1) - (q - k) + 1 = k$.
- Every $RS(k, q)$ achieves the largest possible code minimum distance for this specific block length n.
- $RS(k, q)$ can correct codes up to τ where $\tau = \lfloor (n - k)/2 \rfloor$.

The codewords themselves can then be used to produce polynomials such as $(c_1, c_2, \ldots, c_n) \rightarrow c_1 + c_2 t + c_3 t^2 + \ldots + c_n t^{n-1}$.

Theorem

The Reed-Solomon code $RS(k, q)$ is a cyclic code over $\mathbb{F}_q$. It is generated by $g(t) = (t - \alpha)(t - \alpha^2) \ldots (t - \alpha^{2\tau})$. Its minimum distance is $d = q - k = 2\tau + 1$.

Division Algorithm

The most common method uses division to achieve encoding.

- Take $c = (c_1, \ldots, c_k)$ and create $m(t) = c_k t^{q-2} + \ldots + c_1 t^{q-k-1}$.

Division Algorithm

The most common method uses division to achieve encoding.

- Take $c = (c_1, \ldots, c_k)$ and create $m(t) = c_k t^{q-2} + \ldots + c_1 t^{q-k-1}$.
- Divide $g(t) = (t - \alpha)(t - \alpha^2) \ldots (t - \alpha^{q-k-1})$ into $m(t)$ using the division algorithm. Thus, $m(t) = q(t) \cdot g(t) + r(t)$.

Division Algorithm

The most common method uses division to achieve encoding.

- Take $c = (c_1, \ldots, c_k)$ and create $m(t) = c_k t^{q-2} + \ldots + c_1 t^{q-k-1}$.
- Divide $g(t) = (t - \alpha)(t - \alpha^2) \ldots (t - \alpha^{q-k-1})$ into $m(t)$ using the division algorithm. Thus, $m(t) = q(t) \cdot g(t) + r(t)$.
- Form $f(t) = q(t) \cdot g(t) = m(t) - r(t)$.

Division Algorithm

The most common method uses division to achieve encoding.

- Take $c = (c_1, \ldots, c_k)$ and create $m(t) = c_k t^{q-2} + \ldots + c_1 t^{q-k-1}$.
- Divide $g(t) = (t-\alpha)(t-\alpha^2)\ldots(t-\alpha^{q-k-1})$ into $m(t)$ using the division algorithm. Thus, $m(t) = q(t) \cdot g(t) + r(t)$.
- Form $f(t) = q(t) \cdot g(t) = m(t) - r(t)$.
- $f(t)$ is a codeword because it is a multiple of the generator polynomial $g(t)$. Transmit $f(t)$.

Introduction

- Unique decoding algorithms are constructed to return only one codeword from the received word.

Introduction

- Unique decoding algorithms are constructed to return only one codeword from the received word.
- Let a code C have minimum distance $d \geq 2\delta + 1$ and the weight of the error introduced by the channel be $\mathrm{wt}(e) \leq \delta$.

Introduction

- Unique decoding algorithms are constructed to return only one codeword from the received word.
- Let a code C have minimum distance $d \geq 2\delta + 1$ and the weight of the error introduced by the channel be $\mathrm{wt}(e) \leq \delta$.
- If an error occurs, the nearest neighbor will be the original word.

Introduction

- Unique decoding algorithms are constructed to return only one codeword from the received word.
- Let a code C have minimum distance $d \geq 2\delta + 1$ and the weight of the error introduced by the channel be $\mathrm{wt}(e) \leq \delta$.
- If an error occurs, the nearest neighbor will be the original word.
- If, however, the error has $\mathrm{wt}(e) > \delta$, a *fail* message will be returned.

Introduction

- Unique decoding algorithms are constructed to return only one codeword from the received word.
- Let a code C have minimum distance $d \geq 2\delta + 1$ and the weight of the error introduced by the channel be $\mathrm{wt}(e) \leq \delta$.
- If an error occurs, the nearest neighbor will be the original word.
- If, however, the error has $\mathrm{wt}(e) > \delta$, a *fail* message will be returned.
- There exists a unique decoding algorithm based on the Extended Euclidean Algorithm for the greatest common divisor and the combination of polynomials that gives you the greatest common divisor.

The Basics

- Introduced by Peter Elias in the 1950s.

The Basics

- Introduced by Peter Elias in the 1950s.
- Accept lists of size $\leq L$ with a decoding radius T, the decoder will return at most L codewords which are at most a distance T from the received word.

The Basics

- Introduced by Peter Elias in the 1950s.
- Accept lists of size $\leq L$ with a decoding radius T, the decoder will return at most L codewords which are at most a distance T from the received word.
- We focus on Sudan-Guruswami's work from the late 1990s.

Polynomials in Two Variables

- The ring of polynomials in x, y with coefficients in a field K is denoted as $K[x, y]$.

Polynomials in Two Variables

- The ring of polynomials in x, y with coefficients in a field K is denoted as $K[x, y]$.
- $I = \langle x, y \rangle$ is a nonprincipal ideal in $K[x, y]$.

Polynomials in Two Variables

- The ring of polynomials in x, y with coefficients in a field K is denoted as $K[x, y]$.
- $I = \langle x, y \rangle$ is a nonprincipal ideal in $K[x, y]$.
- A monomial order $>$ in $K[x, y]$ is a relation on the set of monomials $\{x^a y^b | a, b \geq 0\} = \{x^\alpha \mid \alpha = (a, b)\}$.

Polynomials in Two Variables

- The ring of polynomials in x, y with coefficients in a field K is denoted as $K[x, y]$.
- $I = \langle x, y \rangle$ is a nonprincipal ideal in $K[x, y]$.
- A monomial order $>$ in $K[x, y]$ is a relation on the set of monomials $\{x^a y^b | a, b \geq 0\} = \{x^\alpha \mid \alpha = (a, b)\}$.
- A specific type of monomial ordering is the lexicographic order. For $x > y$, $x^a y^b >_{lex} x^c y^d$ if $a > c$, or $a = c$ and $b > d$.

Polynomials in Two Variables

- The ring of polynomials in x, y with coefficients in a field K is denoted as $K[x, y]$.
- $I = \langle x, y \rangle$ is a nonprincipal ideal in $K[x, y]$.
- A monomial order $>$ in $K[x, y]$ is a relation on the set of monomials $\{x^a y^b | a, b \geq 0\} = \{x^\alpha \mid \alpha = (a, b)\}$.
- A specific type of monomial ordering is the lexicographic order. For $x > y$, $x^a y^b >_{lex} x^c y^d$ if $a > c$, or $a = c$ and $b > d$.
- For example, in the weight order $>_{(1,3),\, lex}$, if $a + 3b > c + 3d$ or $a + 3b = c + 3d$ then $x^a y^b >_{lex} x^c y^d$.

Polynomials in Two Variables

- The ring of polynomials in x, y with coefficients in a field K is denoted as $K[x, y]$.
- $I = \langle x, y \rangle$ is a nonprincipal ideal in $K[x, y]$.
- A monomial order $>$ in $K[x, y]$ is a relation on the set of monomials $\{x^a y^b | a, b \geq 0\} = \{x^\alpha \mid \alpha = (a, b)\}$.
- A specific type of monomial ordering is the lexicographic order. For $x > y$, $x^a y^b >_{lex} x^c y^d$ if $a > c$, or $a = c$ and $b > d$.
- For example, in the weight order $>_{(1,3),\ lex}$, if $a + 3b > c + 3d$ or $a + 3b = c + 3d$ then $x^a y^b >_{lex} x^c y^d$.
- The following gives the monomials listed in increasing $(1, 3)$, lex order:

$$1 < x < x^2 < y < x^3 < xy < x^4 < x^2y < x^5 < y^2 < x^3y < x^6 < \ldots.$$

Leading Term

Definition

The leading term of a polynomial f with respect to a monomial order is the term of highest weighted degree in f. It is denoted as $\mathrm{LT}_{>}(f)$.

Weighted Degree

- Given $v \geq 1$, the $(1, v)$-degree of $x^a y^b$ is $a \cdot 1 + b \cdot v = a + bv$.

Weighted Degree

- Given $v \geq 1$, the $(1, v)$-degree of $x^a y^b$ is $a \cdot 1 + b \cdot v = a + bv$.
- $C(v, l)$ is the number of monomials $x^a y^b$ with $(1, v)$-degree $\leq l$.

Proposition

$$C(v, l) = \left(\left\lfloor \frac{l}{v} \right\rfloor + 1 \right) \left(l + 1 - \left\lfloor \frac{l}{v} \right\rfloor \cdot \frac{v}{2} \right)$$

Example

- Let's look at the $x^a y^b$ that have $(1, 4)$-degree ≤ 6.

Example

- Let's look at the $x^a y^b$ that have $(1,4)$-degree ≤ 6.
- $$\begin{aligned} C(v,l) = C(4,6) &= \left(\left\lfloor \frac{6}{4} \right\rfloor + 1\right)\left(6 + 1 - \left\lfloor \frac{6}{4} \right\rfloor \cdot \frac{4}{2}\right) \\ &= (1+1)(6+1-1\cdot 2) = 10. \end{aligned}$$

Example

- Let's look at the $x^a y^b$ that have $(1,4)$-degree ≤ 6.
- $$\begin{aligned} C(v,l) = C(4,6) &= \left(\left\lfloor \frac{6}{4} \right\rfloor + 1\right)\left(6 + 1 - \left\lfloor \frac{6}{4} \right\rfloor \cdot \frac{4}{2}\right) \\ &= (1+1)(6+1-1\cdot 2) = 10. \end{aligned}$$
- Thus there are 10 monomials $x^a y^b$ with $(1,4)$ degree ≤ 6.

Example

- Let's look at the $x^a y^b$ that have $(1,4)$-degree ≤ 6.
- $$\begin{aligned} C(v,l) = C(4,6) &= \left(\left\lfloor \frac{6}{4} \right\rfloor + 1\right)\left(6 + 1 - \left\lfloor \frac{6}{4} \right\rfloor \cdot \frac{4}{2}\right) \\ &= (1+1)(6+1-1\cdot 2) = 10. \end{aligned}$$
- Thus there are 10 monomials $x^a y^b$ with $(1,4)$ degree ≤ 6.
- These monomials are $1, x, x^2, x^3, x^4, x^5, x^6, y, xy$, and $x^2 y$.

Division for Polynomials in Two Variables

The division algorithm for polynomials in two variables works according to a monomial order. Given polynomials $f, f_1, \ldots, f_s \in K[x, y]$, using the division algorithm we can find

$$f = a_1 f_1 + \ldots + a_s f_s + r$$

where $\mathrm{LT}(a_i f_i) \leq \mathrm{LT}(f)$ for all i and $a_i, r \in K[x, y]$. Either the polynomial $r = 0$ or no term in r is divisible by any $\mathrm{LT}(f_i)$.

Gröbner Basis

Definition

If I is an ideal in $K[x, y]$ and $>$ is a monomial order then a subset $G \subset I$ is a Gröbner basis for I with respect to $>$ if

$$\langle LT_{>}(g) | g \in G\rangle = \langle LT_{>}(f) | f \in I\rangle.$$

Theorem

Given an ideal I and a monomial order $>$, there is a unique reduced Gröbner basis for I with respect to $>$.

- List decoding algorithms have two steps: interpolation and factorization.

- List decoding algorithms have two steps: interpolation and factorization.
- Interpolation finds a minimal polynomial $Q(x, y) = a_L(x)y^L + a_{L-1}(x)y^{L-1} + \ldots + a_0(x)$ such that

$$Q(\alpha^i, y_i) = 0 \text{ for all } i = 0, \ldots, q-2.$$

- List decoding algorithms have two steps: interpolation and factorization.
- Interpolation finds a minimal polynomial $Q(x,y) = a_L(x)y^L + a_{L-1}(x)y^{L-1} + \ldots + a_0(x)$ such that

$$Q(\alpha^i, y_i) = 0 \text{ for all } i = 0, \ldots, q-2.$$

- For every Reed-Solomon codeword within distance T of y, factorization gives some $y - f_i(x)$ with $\deg(f_i) \leq k-1$ that divides $Q(x,y)$. In other words, factoring gives

$$Q(x,y) = (y - f_1(x))(y - f_2(x)) \cdots (y - f_L(x)).$$

- List decoding algorithms have two steps: interpolation and factorization.
- Interpolation finds a minimal polynomial $Q(x,y) = a_L(x)y^L + a_{L-1}(x)y^{L-1} + \ldots + a_0(x)$ such that

$$Q(\alpha^i, y_i) = 0 \text{ for all } i = 0, \ldots, q-2.$$

- For every Reed-Solomon codeword within distance T of y, factorization gives some $y - f_i(x)$ with $\deg(f_i) \leq k-1$ that divides $Q(x,y)$. In other words, factoring gives

$$Q(x,y) = (y - f_1(x))(y - f_2(x)) \cdots (y - f_L(x)).$$

- The decoder returns $ev(f_1), ev(f_2), \ldots, ev(f_L)$.

Definition

$Q(x, y)$ has a zero of multiplicity at least m at (α^i, y_i) if

$$Q(x, y) = \sum_{k,l \geq 0} c_{k,l}(x - \alpha^i)^k (y - y_i)^l$$

and

$$c_{0,0} = c_{1,0} = c_{0,1} = \ldots = c_{k,l} = 0$$

for all $k, l \leq m - 1$. $Q(x, y)$ has a zero of multiplicity exactly m if $c_{k,l} \neq 0$ for some k, l with $k + l = m$.

Theorem

Let $\phi_j(x, y)$ denote monomials of the form $x^a y^b$ listed in increasing order according to an arbitrary monomial order and

$$Q(x, y) = \sum_{j=0}^{C} a_j \phi_j(x, y).$$

Then a nonzero $Q(x, y)$ polynomial exists that interpolates the points (α^i, y_i) for $i = 1, 2, \ldots, n$ with multiplicity m at each point if

$$C = n\binom{m+1}{2}.$$

Theorem

Let $K_m = \min\{K : C(k-1, mK-1)\} > \binom{m+1}{2}n$. Then if the following are satisfied:

$$\begin{cases} C(k-1, l) > \binom{m+1}{2}n \\ mK_m > l \\ p(x) \text{ has degree} \leq k-1 \\ y_i = p(\alpha^i) \text{ for at least } K_m \text{ different } i, \end{cases}$$

$Q(x, y)$ is divisible by $y - p(x)$.

Introduction

- Kwankyu Lee and Michael O'Sullivan introduced a new way to solve the interpolation step.

Introduction

- Kwankyu Lee and Michael O'Sullivan introduced a new way to solve the interpolation step.
- It finds the minimal polynomial of an ideal using Gröbner bases of modules.

Introduction

- Kwankyu Lee and Michael O'Sullivan introduced a new way to solve the interpolation step.
- It finds the minimal polynomial of an ideal using Gröbner bases of modules.
- It starts with a set of generators of the module induced from the ideal for the points $\{P_1, P_2, \ldots, P_n\}$ where $P_i = (\alpha^i, y_i)$.

Introduction

- Kwankyu Lee and Michael O'Sullivan introduced a new way to solve the interpolation step.
- It finds the minimal polynomial of an ideal using Gröbner bases of modules.
- It starts with a set of generators of the module induced from the ideal for the points $\{P_1, P_2, \ldots, P_n\}$ where $P_i = (\alpha^i, y_i)$.
- It then translates the generators to a Gröbner basis of the module.

Definition

$I_{v,m}$ is an ideal of all polynomials $p(x,y)$ in $\mathbb{F}_q[x,y]$ such that $p(x,y)$ vanishes to multiplicity m at all (α_i, v_i).

Definition

$\mathbb{F}_q[x,y]_l$ is a free module over $\mathbb{F}_q[x]$ with basis $\{1, y, y^2, \ldots, y^l\}$. It can be written as

$$\mathbb{F}_q[x,y]_l = \{p(x,y) \mid \deg_y(p(x,y)) \leq l\}.$$

Monomials in this module are $x^i y^j$ with $i \geq 0$ and $0 \leq j \leq l$.

Definition

$I_{v,m,l} = I_{v,m} \cap \mathbb{F}_q[x,y]_l$.

The Algorithm

- We will use a set of generators of $I_{v,m,l}$.

The Algorithm

- We will use a set of generators of $I_{v,m,l}$.
- It has input m, l, and $v = (v_1, v_2, \ldots, v_n)$ and monomial order $>_{k-1}$.

The Algorithm

- We will use a set of generators of $I_{v,m,l}$.
- It has input m, l, and $v = (v_1, v_2, \ldots, v_n)$ and monomial order $>_{k-1}$.
- We will let $g_i = \sum_{j=0}^{l} a_{ij} y^j$ for $0 \leq i \leq l$.

This algorithm is creating a Gröbner basis $\{g_0, g_1, \ldots, g_l\}$ of S such that $\deg_y(LT(g_i)) = i$ for $0 \leq i \leq l$. This algorithm begins with:

$$
\begin{aligned}
g_0 &= a_{00} \\
g_1 &= a_{10} + a_{11}y \\
g_2 &= a_{20} + a_{21}y + a_{22}y^2 \\
&\vdots \\
g_l &= a_{l0} + a_{l1}y + a_{l2}y^2 + \ldots + a_{11}y^l.
\end{aligned}
$$

The algorithm goes through the steps such that each time g_s and g_r are updated, $\{g_0, g_1, \ldots, g_l\}$ still generates S. It terminates when we have $\deg_y(LT(g_i)) = i$ for $0 \leq i \leq l$.

The FGLM Algorithm

- It takes an input of a Gröbner basis for a zero-dimensional ideal I and outputs another Gröbner basis for I for some other monomial order.

The FGLM Algorithm

- It takes an input of a Gröbner basis for a zero-dimensional ideal I and outputs another Gröbner basis for I for some other monomial order.
- We use the lex order as the new monomial order.

The FGLM Algorithm

- It takes an input of a Gröbner basis for a zero-dimensional ideal I and outputs another Gröbner basis for I for some other monomial order.
- We use the lex order as the new monomial order.
- $\mathbb{F}$ is a field and $R = \mathbb{F}[x_1, \ldots, x_n]$ is the ring of polynomials with n variables and coefficients in $\mathbb{F}$.

The FGLM Algorithm

- It takes an input of a Gröbner basis for a zero-dimensional ideal I and outputs another Gröbner basis for I for some other monomial order.
- We use the lex order as the new monomial order.
- $\mathbb{F}$ is a field and $R = \mathbb{F}[x_1, \ldots, x_n]$ is the ring of polynomials with n variables and coefficients in $\mathbb{F}$.
- A zero-dimensional ideal I is one such that

$$\dim_{\mathbb{F}} \mathbb{F}[x_1, \ldots, x_n]/I < \infty.$$

Remainder Arithmetic

- Dividing $f \in R$ by G results in:

$$f = h_1 g_1 + \ldots + h_t g_t + \overline{f}^G.$$

Remainder Arithmetic

- Dividing $f \in R$ by G results in:

$$f = h_1 g_1 + \ldots + h_t g_t + \overline{f}^G.$$

- $\overline{f}^G$ is a linear combination of the monomials $x^\gamma \notin \langle LT(I) \rangle$ which is a basis for $\mathbb{F}[x_1, \ldots, x_n]/I$.

Remainder Arithmetic

- Dividing $f \in R$ by G results in:

$$f = h_1 g_1 + \ldots + h_t g_t + \overline{f}^G.$$

- $\overline{f}^G$ is a linear combination of the monomials $x^\gamma \notin \langle LT(I) \rangle$ which is a basis for $\mathbb{F}[x_1, \ldots, x_n]/I$.
- Since G is a Gröbner basis, $f \in I$ if and only if $\overline{f}^G = 0$.

The Code

- Input: The lex order and G_1, the Gröbner basis of the original monomial ordering.

The Code

- Input: The lex order and G_1, the Gröbner basis of the original monomial ordering.
- Algorithm updates a list $G_2 = \{g_1, \ldots, g_k\}$ where each g_i is an element of the ideal I.

The Code

- Input: The lex order and G_1, the Gröbner basis of the original monomial ordering.
- Algorithm updates a list $G_2 = \{g_1, \ldots, g_k\}$ where each g_i is an element of the ideal I.
- Algorithm updates B which is a list of monomials that is initially empty.

The Code

- Input: The lex order and G_1, the Gröbner basis of the original monomial ordering.
- Algorithm updates a list $G_2 = \{g_1, \ldots, g_k\}$ where each g_i is an element of the ideal I.
- Algorithm updates B which is a list of monomials that is initially empty.
- Algorithm moves through a list of monomials of the form x^γ that increase by lex order to create the new Gröbner basis.

The Code

1. Compute $\overline{x^\gamma}^G$.
2. If $\overline{x^\gamma}^G$ is linearly dependent of the monomials in B then add g to the list of G_2 as the last element.
3. If $\overline{x^\gamma}^G$ is linearly independent of the monomials in B then add x^γ to B as the last element.
4. End if the leading term of the last added polynomial g is a power of x_1 where x_1 is the greatest variable in our lex order.
5. Replace x^γ by the next monomial in lex order which is not divisible by any of the monomials $LT(g_i)$ for $g_i \in G_2$ and go back to Step 1.

We are applying a module version of the above algorithm to the Gröbner basis $\{g_0, \ldots, g_l\}$ for $I_{v,m,l}$ with Position over Term order and converting it to a $>_{(l,k-1)}$ order Gröbner basis for $I_{v,m,l}$.

Theoretical Comparison

- We can compare both algorithms by calculating the upper bound of how many multiplication operations are needed.

Theoretical Comparison

- We can compare both algorithms by calculating the upper bound of how many multiplication operations are needed.
- If two polynomials of degree a and b are multiplied it requires $(a+1)(b+1)$ operations over $\mathbb{F}$.

Theoretical Comparison

- We can compare both algorithms by calculating the upper bound of how many multiplication operations are needed.
- If two polynomials of degree a and b are multiplied it requires $(a+1)(b+1)$ operations over $\mathbb{F}$.
- Big-O notation describes the behavior of a function when the variable tends to infinity.

Theoretical Comparison

- We can compare both algorithms by calculating the upper bound of how many multiplication operations are needed.
- If two polynomials of degree a and b are multiplied it requires $(a+1)(b+1)$ operations over $\mathbb{F}$.
- Big-O notation describes the behavior of a function when the variable tends to infinity.
- For example, if a function $f(n) = O(n^2)$, then $f(n) \leq cn^2$ for some constant c and all values of $n > n_0$.

Results

- The Lee-O'Sullivan algorithm requires

$$O(n^4m^5)$$

multiplication operations.

Results

- The Lee-O'Sullivan algorithm requires
$$O(n^4 m^5)$$
multiplication operations.
- The FGLM algorithm has at most
$$O(n^3 m^6)$$
multiplication operations.

Results

- The Lee-O'Sullivan algorithm requires
$$O(n^4m^5)$$
multiplication operations.
- The FGLM algorithm has at most
$$O(n^3m^6)$$
multiplication operations.
- Then for those codes that have big n but the same m, we expect that the FGLM algorithm is better. This corresponds to large fields with small m.

Experimental Comparison

- We used the original procedure for the Lee-O'Sullivan algorithm and strived to optimize the FGLM algorithm.

Experimental Comparison

- We used the original procedure for the Lee-O'Sullivan algorithm and strived to optimize the FGLM algorithm.
- An error vector was randomly created and added to a randomly chosen codeword to create the received word.

Experimental Comparison

- We used the original procedure for the Lee-O'Sullivan algorithm and strived to optimize the FGLM algorithm.
- An error vector was randomly created and added to a randomly chosen codeword to create the received word.
- Since Maple times varied, we calculated the average of 10 run times of each algorithm.

Experimental Comparison

- We used the original procedure for the Lee-O'Sullivan algorithm and strived to optimize the FGLM algorithm.
- An error vector was randomly created and added to a randomly chosen codeword to create the received word.
- Since Maple times varied, we calculated the average of 10 run times of each algorithm.
- For fields smaller than $\mathbb{F}_{11}$, the Lee-O'Sullivan algorithm won every time.

Field of Size 11

Run	Weight of error	Codewords	AVG FGLM	AVG L.O.	Winner
1	2	1	3.083	2.370	FGLM
2	3	2	3.540	3.341	FGLM
3	4	1	4.170	4.440	L.O.
4	2	1	3.063	2.415	FGLM
5	4	2	4.052	3.445	FGLM
6	3	2	1.931	2.528	L.O.
7	3	2	3.479	3.221	FGLM

Table: Comparison of Lee-O'Sullivan and FGLM algorithm for field of size q=11, multiplicity m=4, and lists of size l=9.

Field of Size 17

Run	Weight of error	Codewords	AVG FGLM	AVG L.O.	Winner
1	10	1	6.358	6.524	L.O.
2	9	1	6.515	6.409	FGLM
3	8	1	6.532	6.122	FGLM
4	8	2	6.357	5.802	FGLM
5	9	3	6.552	6.133	FGLM
6	10	2	6.714	6.534	FGLM

Table: Comparison of Lee-O'Sullivan and FGLM algorithm for field of size q=17, multiplicity m=3, and lists of size l=9.

Summary

- Decoding algorithms are useful to correct errors.

Summary

- Decoding algorithms are useful to correct errors.
- When the size of the field is greater than $\mathbb{F}_{11}$, we expect that the FGLM algorithm will consistently be faster than the Lee-O'Sullivan algorithm.

Summary

- Decoding algorithms are useful to correct errors.
- When the size of the field is greater than $\mathbb{F}_{11}$, we expect that the FGLM algorithm will consistently be faster than the Lee-O'Sullivan algorithm.
- If you are trying to decode received words from a smaller field, the Lee-O'Sullivan algorithm gives superior performance.

For Further Reading

- D. A. Cox, J. B. Little, and D. O'Shea, *Using Algebraic Geometry*, New York: Springer, 2005.
- J. C. Faugere, P. Gianni, D. Lazard, and T. Mora, *Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering*, Journal of Symbolic Computation **16** (1993), 329-44.
- K. Lee and M. O'Sullivan, *List Decoding of Reed Solomon Codes from a Gröbner Basis Perspective*, Journal of Symbolic Computation **43** (2008), 645-58.
- T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*, Hoboken, NJ: Wiley-Interscience, 2005.
- Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, *A method for solving key equation for decoding Goppa codes*, Inform. and Control **27** (1975), 87-99.