ERROR CONTROL CODING (ECE 548) January 24, 2019
Rutgers, Spring 2019 Course Syllabus

ERROR CONTROL CODING, ECE 548

You may have heard of Morse, bar and QR codes, ISBN, and blockchains. These codes, and many more, play important roles in numerous scientific disciplines and virtually all telecommunication systems. In practice, codes are used to efficiently insure reliable, secure, and private transmission and storage of information. In theory, codes are used to e.g., study computational complexity, design screening experiments, provide a bridge between statistical mechanics and information theory, and even help understand the (quantum) spacetime fabric of reality. One can also use codes for entertainment, e.g., to solve balance puzzles such as the penny weighing problem, or to design social (hat color) guessing-game strategies that significantly increase the odds of winning.

Learning Objectives:
The students will learn the fundamentals of coding theory and practice, as well as a selected number of more advanced topics of their individual interests.

Instructor: Emina Soljanin `emina.soljanin@rutgers.edu`

Office hours: by appointment, CoRE 511, `848-445-5256`.

Class time and place: Tue&Thr, 1:40 PM – 3:00 PM, EE-203.

Prerequisites: probability and algebra at the undergraduate level.

Grading: homework 20%, 3 midterm exams 15% each, final project 35%.
Exams will be in class, on February 12 and on (or about) March 14 and April 18.

Text: *Error Control Coding* (2nd Edition) by Lin and Costello
not required

Course outline:

- Fundamental concepts through examples
- Linear block codes, Ch. 2–7
- Convolutional codes, Ch. 11&12
- Turbo codes, Ch. 16
- LDPC codes, Ch. 17
- Hybrid ARQ, Ch. 22
- Network, rateless, storage, polar, and spatially-coupled codes
- Coding theory in other scientific disciplines

The extent to which these topics will be covered will vary in accordance with the backgrounds and research needs of the students in the class.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #2, January 24

This lecture[2] informally introduces several important notions: information, hashing, and (decoding) algorithms.

Bits of Information

Bit is a unit of information that we get when we ask a yes/no question – yes or no, true or false, on or off, 0 or 1. Suppose you want to find out the position of the black king (that can be equally likely anywhere) on a chessboard. Take a look at Fig. 1. What is the minimum number of yes/no questions you need to ask?

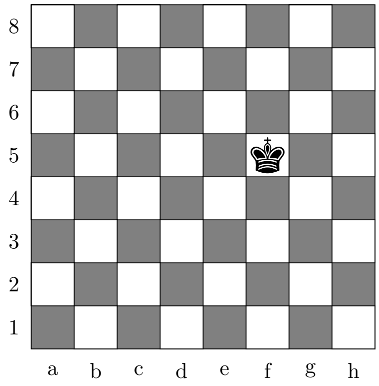


Figure 1: What is the minimum number of yes/no questions that have to be asked to locate the king on a chessboard?

To represent a bit in a computer, we need a physical entity which can exist in two distinguishable physical states. For example, magnetized cells in hard disk drives could be oriented in two different directions: "up" (0) or "down" (1). Flesh memory cells made from floating-gate transistors act as switches that could be open (0) or closed (1). (There are multi-level cell devices that can store more than one bit per cell.)

A physical system with $N = 2^b$ distinguishable physical states can represent b bits of information. To specify an object in a set of N, we need $\lceil \log_2 N \rceil$ binary digits; Table 1 show how for $N = 8$.

Decimal	Binary	mod 2	Parity
0	000	0	0
1	001	1	1
2	010	0	1
3	011	1	0
4	100	0	1
5	101	1	0
6	110	0	0
7	111	1	1

Table 1: We can specify each object in a set of 8 by assigning to it a unique label, e.g., a decimal number or a binary string.

Note that distinguishable is the crucial word here. Distinguishable how? By the naked eye? By a given measuring apparatus? Is there some fundamental limit to the number of states that can be distinguished by a physical measurement regardless of whether we can build it or not? If your measuring apparatus can only tell you the last digits of the numbers in Table 1, you will get only a single bit of information telling you whether the number is even or odd (see the third column of Table 1). You will get a single but different bit of information if your measuring apparatus can only tell you the *parity*, namely,

[1] Rutgers, ECE 548, Spring 2019

[2] You are likely to find this class more conceptually than technically hard. In that sense, this is the hardest lecture.

the XOR of the digits in the binary string representing the number (see the fourth column of Table 1).

Algorithms and Hashing

A Penny Weighing Problem: You are given a balance scale and 8 pennies, one of which has a different weight. What is the minimum number of measurements that will always let you determine which penny has a different weight? How will you perform the measurements?

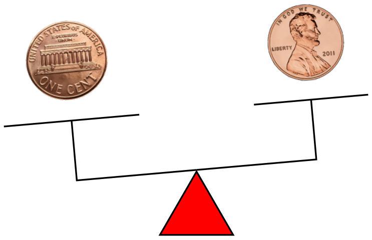


Figure 2: How would you use a balance scale to determine which of the 8 pennies has a different weight?

The minimum number of measurements that will always let us determine which penny has a different weight is three. Why? A possible way to perform the three measurements[3] is given in Table 2. The three rows starting with M1, M2, and M3 correspond to the three measurements. The table entry at the intersection between a column corresponding to a penny and a raw corresponding to a measurement indicates whether the penny is put on the scale in that measurement (0 if it is not) and if yes, whether it is placed on the left platform (L) or on the right platform R.

		PENNY							
		⓪	①	②	③	④	⑤	⑥	⑦
ON SCALE	M1	0	0	0	0	L	L	R	R
	M2	0	0	L	L	0	0	R	R
	M3	0	L	0	L	0	R	0	R

Table 2: Pennies placement on the scale in three measurements. A penny cen be placed left (L), right (R) or not at all.

Suppose that the penny 4 has different weight, then measurement M1 will result in an unbalanced state of the scale and M2 and M3 in the balanced state of the scale, as illustrated in Fig. 3.

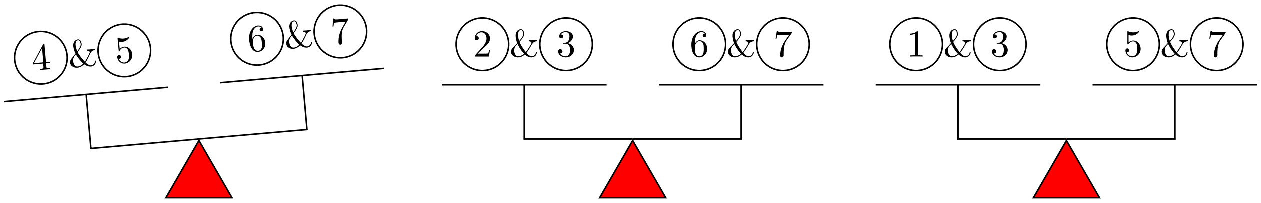


Figure 3: An example of measurement outcomes. Which penny has different weight?

Observe that since there is only one penny of different weight, a measurement will result in an unbalanced state of the scale iff the penny of different weight is placed on the scale in that measurement. Therefore, the possible measurement outcomes are as given in Table 3. In each measurement, the scale can be either balanced (0) or unbalanced (1). Not that for each of the 8 "different penny" possibilities,

[3] an algorithm

we have a different set of measurement outcomes. Therefore a set of measurement outcomes uniquely identifies a different penny.

		DIFFERENT PENNY							
		0	1	2	3	4	5	6	7
SCALE STATE	M1	0	0	0	0	1	1	1	1
	M2	0	0	1	1	0	0	1	1
	M3	0	1	0	1	0	1	0	1

Table 3: Scale states corresponding to measurements for each of the 8 "different panny" possibilities. The scale can be either balanced (0) or unbalanced (1).

Some Observations

1. We have committed to the way we perform the three measurements before the measuring process started. We do not *adapt*[4] our measuring actions based on the results of the previous measurement, e.g., how we perform M2 does not change based on the outcome of M1.
2. Having some additional information could be helpful in designing a set of measurements, even if it cannot reduce the number of measurements.
3. How we conduct measurements evidently depends on the kind of scale we have. And so does the number of measurements. What would you do if you had a scale which has the unit weight corresponding to a regular penny fixed to the right tray, as in Fig. 4, and you can only use the left tray to place pennies?

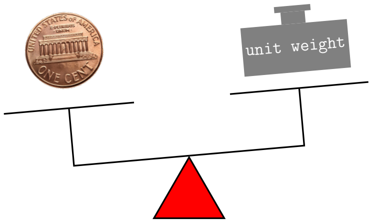


Figure 4: In this scale, there is some unit weight fixed to the right tray.

Homework - due January 31

Find all 1×7 binary raw vectors $\mathbf{c}$ such that $\mathbf{c} \cdot \mathrm{H} = \mathbf{0}$ where

$$\mathrm{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

and the addition and multiplication are defined as follows:

+	0	1
0	0	1
1	1	0

·	0	1
0	0	0
1	0	1

In other words, find all vectors which are orthogonal to all three rows of the matrix. For example, one such vector is $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$.

[4] Non-adaptive measuring can be as powerful as adaptive.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #3, January 29

This lecture covers some fundamental material in abstract algebra we use in coding theory.

Bits as Mathematical Objects

In this class, we will treat bits as mathematical objects.[2] For us, bits take values in the set $\{0, 1\}$ where we can add and multiply as follows:

XOR

$\oplus$	0	1
0	0	1
1	1	0

AND

$\cdot$	0	1
0	0	0
1	0	1

Figure 1: Binary arithmetic.

Associative and distributive laws for binary addition and multiplication are identical to those for real numbers. Strings of n bits are mathematical objects that live in the field $\mathbb{F}_{2^n}$, which is a set of 2^n elements with specially defined addition and multiplication.

Let $\mathbf{c}$ be a 1×7 binary raw vector and H the following binary matrix:

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Recall our penny weighing problem, and consider the product $\mathbf{c} \cdot H = \mathbf{0}$, where the arithmetic is done as defined in Fig. 1. Check to see that this multiplication is performing the *measurements* we did with a balance scale in the penny weighing problem. The multiplication result corresponds to the binary representation of the number indicating the position of the different bit.

After the measuring, in order make all bits identical, we would have to flip the bit at position 3. This flipping can be performed by e.g., 1) a NOT operation on the bit in position 3 or 2) an XOR operation (binary addition in Fig. 1) on the bit n position 3 together with the bit of the fixed value 1.

Math Interlude

Coding theory relies on several branches of mathematics, which we will introduce as the need arises. In order to be able to study linear codes, we next review some basic algebraic notions.

[1] Rutgers, ECE 548, Spring 2019

[2] Other classes at ECE and Physics study bits as physical systems.

Fundamental Structures in Abstract Algebra

Group $(G, \circ)$ A group is a set G together with an operation $\circ$: $G \times G \to G$ satisfying:

1. $\circ$ is associative: $(a \circ b) \circ c = a \circ (b \circ c)$
2. There is an element e in G s.t. $a \circ e = a$ and $e \circ a = a$ for every element a in G. e is called neutral element.
3. For every element a in G, there is an element a^{-1} in G s.t. $a \circ a^{-1} = a^{-1} \circ a = e$. a^{-1} is called the inverse of a.

If $\circ$ is commutative, we say that G is commutative or Abelian.

Depending on the context, we will call a group 1) *additive*, its operation $+$ addition, and its neutral element 0, or 2) *multiplicative*, its operation $*$ or $\cdot$ multiplication, and its neutral element 1 (unity).

Two examples:

1. $(\{0,1\}, \oplus)$ is an additive group.
2. $(\{0,1\}, \cdot)$ is not a group.

Ring $(A, +, *)$ The most basic of the two-operation structures is called a ring: Ring is a set A with operations called addition $+$ and multiplication $*$ satisfying:

1. $(A, +)$ is an Abelian group.
2. Multiplication is associative.
3. Multiplication is distributive over addition. That is, for all a, b, and c in A, we have $a(b+c) = ab + ac$

Examples of rings:

1. set of integers $\mathbb{Z}$
2. set of $n \times n$ matrices over $\mathbb{Z}$

Natural numbers $\mathbb{N}$ is not a ring.

When the multiplication operation is commutative, we say that A is a commutative (Abelian) ring.

Field $(\mathbb{F}, +, *)$ If $(\mathbb{F}, +, *)$ is a commutative ring with unity in which every nonzero element has a multiplicative inverse, it is called a field:

1. $(\mathbb{F}, +)$ is an Abelian group.
2. $(\mathbb{F} \setminus \{0\}, *)$ is an Abelian group.

Examples of fields:

1. set of rational numbers $(\mathbb{Q}, +, \cdot)$
2. set of complex numbers $(\mathbb{C}, +, \cdot)$
3. **finite field** $\mathbb{F}_2 = (\{0,1\}, \oplus, \cdot)$

A linear space over a field $\mathbb{F}$ is an additive Abelian group v together with an operation of *multiplication by scalars* $\mathbb{F} \times V \to V$. The elements of v are called vectors and the elements of $\mathbb{F}$ are called scalars. The product of $\alpha \in \mathbb{F}$ and $v \in V$ is denoted by $\alpha v \in V$. In addition, there are requirements connecting $\mathbb{F}$ and v:

For all $\alpha, \beta \in \mathbb{F}$ and $v, w \in V$, we have

1. $(\alpha\beta)v = \alpha(\beta v)$
2. $\alpha(v + w) = \alpha v + \alpha w$

Examples of linear spaces over the binary field $\mathbb{F}_2$ (any field):

1. $\mathbb{F}_2^n$ of n-tuples over $\mathbb{F}_2$ aka n-space over $\mathbb{F}_2$
2. $\mathbb{F}_2^{k \times n}$ of $k \times n$ matrices over $\mathbb{F}_2$
3. All polynomials over $\mathbb{F}_2$

3. $(\alpha + \beta)v = \alpha v + \beta v$

4. $1v = v$, where 1 is the unity in $\mathbb{F}$

A linear combination of vectors $v_1, \dots, v_m$ is a vector of the form

$$\alpha_1 v_1 + \cdots + \alpha_m v_m.$$

Linear subspace W of V is a nonempty subset ($W \subseteq V$) closed under linear combinations.

The span of vectors $v_1, \dots, v_m$ is the set of all linear combinations of $v_1, \dots, v_m$:

$$\text{span}(v_1, \dots, v_m) = \{\alpha_1 v_1 + \cdots + \alpha_n v_m \mid \alpha_1, \dots, \alpha_m \in \mathbb{F}_q\}.$$

Vectors $v_1, \dots, v_m$ are linearly independent when

$$\alpha_1 v_1 + \cdots + \alpha_m v_m = 0 \text{ only if } \alpha_1 = \cdots = \alpha_m = 0.$$

A basis of a vector space V is a set of linearly independent vectors in V that spans v.

The dimension of a vector space v is the number[3] of vectors of a basis of v over $\mathbb{F}$.

An inner product space is a vector space V over the field $\mathbb{F}$ together with an inner product map $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{F}$. We will mostly work with n-spaces over finite fields and define the inner product as follows: Let $u = (u_1, \dots, u_n)$ and $v = (v_1, \dots, v_n)$ be two vectors in $\mathbb{F}^n$, then

$$\langle u, v \rangle = \sum_{i=1}^{n} u_i v_i = u_1 v_1 + \cdots + u_n v_n.$$

We say that two vectors are orthogonal iff their inner product is zero.[4]

Let V and W be linear spaces over the same field. Then $f : V \to W$ is a linear map if for every $v, u \in V$ and $\alpha \in \mathbb{F}$, we have

1. $f(v + u) = f(v) + f(u) \quad \leftarrow \quad$ additive

2. $f(\alpha v) = \alpha f(v) \quad \leftarrow \quad$ homogeneous

An $[n, k]_q$ linear code is a k dimensional subspace of $\mathbb{F}_q^n$.
Encoder for an $[n, k]_q$ linear code is a linear map $\mathbb{F}_q^k \to \mathbb{F}_q^n$.

[3] Does each basis have the same number of vectors?

[4] Show that the set of all 1×7 binary raw vectors $\mathbf{c}$ such that $\mathbf{c} \cdot H^T = \mathbf{0}$ form a subspace of $\mathbb{F}_2^7$. (H is the 3×7 matrix defined above.)

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #4, January 31

This lecture defines (systematic) linear block codes and introduces the notion of minimum distance and error correction.

An $[n, k]_q$ linear code is a k dimensional subspace of $\mathbb{F}_q^n$.
Encoder for an $[n, k]_q$ linear code is a linear map $\mathbb{F}_q^k \to \mathbb{F}_q^n$.

We have seen last time that the set of all 1×7 binary raw vectors $\mathbf{c}$ such that $\mathbf{c} \cdot H^T = \mathbf{0}$ form a subspace of $\mathbb{F}_2^7$, where H is the 3×7 matrix

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Every $[n, k]$ linear block code $\mathbb{C}$ has associated with it an $(n-k) \times n$ matrix H with linearly independent raws and the property that

$$\mathbf{c} \cdot H^T = 0$$

We refer to such H as the <u>parity check matrix</u> of the code.[2] Linear code is the null space of the parity check matrix. A <u>generator matrix</u> G is a $k \times n$ matrix whose rows form a basis for a linear code.[3]

What About Encoding?

With the Parity Check Matrix: We need to generate $1 \times n$ vectors that satisfy $(n-k)$ linearly independent equations $\mathbf{c} \cdot H^T = 0$. We can do that by taking some k of the components of these vectors to be arbitrarily specified, and then solve for the remaining $(n-k)$ components. The arbitrarily specified components correspond to the data (message, information) word.

With the Generator Matrix: If we know the generator matrix G, we can find all codewords as the linear combinations of the raws of G. The coefficients in this linear combinations correspond to data word.

We can map the data into codewords by, e.g., encoding circuitry or lookup tables.

Systematic Codes

If the generator matrix for an [n,k]-code is in a standard form

$$G = \begin{bmatrix} I_k | P \end{bmatrix}$$

[1] Rutgers, ECE 548, Spring 2019

[2] Is the parity check matrix unique?

[3] What is GH^T equal to?

we say that the code is systematic. The parity check matrix for a systematic code can be easily derived from its generator matrix (and vice versa):

$$H = \left[-P^{\top}|I_{n-k}\right],$$

$\Rightarrow \ G \cdot H^{T} = P - P = 0$

Figure 1: A Systematic [7, 4] Hamming Code

data	codeword
0000	0000000
0001	0001011
0010	0010111
0011	0011100
0100	0100110
0101	0101101
0110	0110001
0111	0111010
1000	1000101
1001	1001110
1010	1010010
1011	1011001
1100	1100011
1101	1101000
1110	1110100
1111	1111111

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

What about Errors?

What can we do with these codes? How *good* are they? The bit-flip error model is described by the binary addition of vectors having 1s at places where errors occurred and zeros otherwise. That is, if we send a codeword $\mathbf{c} \in \mathbb{C}$ and receive it with its bits flipped according to error vector $\mathbf{e} \in \mathbb{F}_2^n$, we receive word $\mathbf{r}$ given by[4]

$$\mathbf{r} = \mathbf{c} + \mathbf{e}$$

For $n = 7$, we can have, e.g.,

$\mathbf{e} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ meaning an error happened at position 1.

$\mathbf{e} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ meaning errors happened at positions 6 and 7.

[4] $\Rightarrow \mathbf{r}$ can be any vector in $\mathbb{F}_2^n$.

What about Decoding?

We can find $\mathbf{c}$ from $\mathbf{r}$ by computing the syndrome:

$$\mathbf{r} \cdot H^T = (\mathbf{c} + \boldsymbol{e}) \cdot H^T = \underbrace{\mathbf{c} \cdot H^T}_{=0} + \boldsymbol{e} \cdot H^T = \boldsymbol{e} \cdot H^T \;\leftarrow\; \text{syndrome } \mathbf{s}.$$

$\mathbf{s} = \mathbf{0} \;\Rightarrow \mathbf{r}$ is a codeword.
In our original [7, 4] example, when $\mathbf{s}$ matches the i-th column of H, that means that the i-th bit of $\mathbf{c}$ was flipped. $\Rightarrow$ we can get $\mathbf{c}$ by flipping back the i-th bit of $\mathbf{r}$.[5]

When can one codeword be sent and another decoded?

The Hamming Weight and Distance

The Hamming Distance between two binary words is the number of positions at which one has a 1 and the other 0.

The Hamming Weight of a binary word is the number of 1s in the word (distance from the all 0 word).

The Minimum Distance d of a code is the smallest of all pairwise distances of its codewords (the minimum weight for linear codes).

There is one-to-one mapping between weight w codewords and linear combinations of w columns of H that sum to 0. We are here dealing with a $[7, 4, 3]_2$ code.

Error Correction Capabilities

If a code has minimum distance d, it can 1) detect $d - 1$ and correct $\lfloor \frac{d-1}{2} \rfloor$ errors, and 2) recover from $d - 1$ erasures. An error can be correctable, detectable, or undetectable.

The Error Probability over the BSC

The binary symmetric channel (BSC) with crossover probability p flips its inputs with probability p. If the channel is memoryless, the each bit in the input sequence is flipped independently of others. I an n-bit word is sent through the channel, the probability that an error will be made is

1. $1 - (1 - p)^n$ if we don't use a code (all bits have to be received correctly)
2. $1 - (1 - p)^n - \sum_{\ell=1}^{t} \binom{n}{\ell} p^\ell (1 - p)^{n-\ell}$ if we use a code with distance d with $t = \lfloor \frac{d-1}{2} \rfloor$

What is the price we have to pay for using coding?

[5] We will talk about syndrome decoding and other forms of decoding later in the class.

Hamming Codes With Parameter $r \geqslant 2$

Its $r \times (2^r - 1)$ parity-)check has for its columns all the non-zero length-r binary strings.

$\Rightarrow$

block length	$2^r - 1$
message length	$2^r - 1 - r$
distance	3
alphabet size	2
$[n, k, d]_q$	$[2^r - 1, 2^r - r - 1, 3]_2$-code

Homework – Due February 7

For the $[7, 4, 3]$ Hamming code, find

1. the histogram of the pairwise distances between codewords,
2. the histogram of the codeword weights weights, and
3. the histogram of the distances between a non-zero codeword of your choice and all codewords in the code.

Compare the three histograms.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #5, February 5

This lecture derives three bounds on codes.

The Hamming Weight and Distance

The Hamming Distance between two q-ary words is the number of positions at which they differ.

The Hamming Weight of a q-ary word is the number of non-zero elements in the word (distance from the all-zero word).

The Minimum Distance d of a code is the smallest of all pairwise distances of its codewords (the minimum weight for linear codes).

Hamming Sphere $B_\rho(x)$ of radius $\rho > 0$ centered at a point x in $\mathbb{F}_q^n$ is defined by

$$B_\rho(x) = \{y \in \mathbb{F}_q^n \mid d(y,x) \leqslant \rho\}.$$

Three Bounds on Codes

We would like to have codes with many codewords and large minimum distance, but these are two competing requests. Why?

The Hamming (Sphere Packing) Bound

Let $\mathbb{C}$ be a q-ary code with length n and minimum distance d, and denote $t = \lfloor \frac{d-1}{2} \rfloor$. Then, because spheres of radius t around codewords do not touch,

$$\sum_{\mathbf{c} \in \mathbb{C}} |B_t(\mathbf{c})| \leqslant q^n$$

Since $|B_t(\mathbf{c})| = \sum_{\ell=0}^{t} \binom{n}{\ell}(q-1)^\ell$, we have the following upper bound on the size of any q-ary code $\mathbb{C}$:

$$|\mathbb{C}| \leqslant \frac{q^n}{\sum_{\ell=0}^{t} \binom{n}{\ell}(q-1)^\ell}$$

$\Rightarrow$ Any binary code with $d = 3$ has at most $2^n/(1+n)$codewords.

Codes which achieve the sphere packing bound with equality are called perfect codes. Hamming codes are perfect codes.[2]

[1] Rutgers, ECE 548, Spring 2019

[2] Show that a Hamming code with block length n has $2^n/(1+n)$ codewords.

The Gilbert-Varshamov Bound

Let $A_q(n, d)$ denote the maximum possible size a q-ary code with length n and minimum distance d can have. Consider the radius $d - 1$ spheres around codewords in a code of the maximum size. Any word not in the code has to be in at least one such sphere. Why? Consequently, we have

$$A_q(n, d) \geqslant \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j} (q-1)^j}.$$

The Singleton Bound

If we erase the first $d - 1$ letters of each codeword of a q-ary code with length n and minimum distance d, the resulting words will be all different. Since there are at most q^{n-d+1} different q-ary words of length $n - (d - 1)$, we have

$$|\mathbb{C}| \leqslant q^{n-d+1}.$$

Codes that achieve equality in the Singleton bound are called MDS codes (maximum distance separable).

The Hat Problem

WHAT ARE THE ODDS OF WINNING THE FOLLOWING GAME?

Seven (in general n) prisoners enter a courtroom, and a red or a blue hat is placed on each person's head. The judge determines the color of each hat by a fair-coin toss, with the outcome of one coin toss having no effect on the others.[3] Each person can see the other prisoners' hats but not his own.

No communication of any sort is allowed, except for an initial *strategy session* before the game begins. Once they have had a chance to look at the other hats, the prisoners must simultaneously guess the color of their own hats or pass. The prisoners play as a team and all win freedom when at least one prisoner guesses the color of his or her own hat without any incorrect guesses being made.

WHAT IS A GOOD STRATEGY?

One of the prisoners proposes that he always guess RED while the other prisoners pass, which would result into fifty-fifty odds for winning. But there is a coding theorist in the group who claims they can do better, if they follow his strategy.

The prisoners have to learn the codewords of the $[7, 4, 3]$ Hamming code, and will call the two colors 0 and 1. Each prisoner is assigned a position number. If the 6 ordered bits visible to the prisoner look like

[3] Bernoulli trials.

a codeword that is punctured at his position, the prisoner guesses the value that will make it a non-codeword. Otherwise, he passes.

WHAT IS THE BEST WE CAN DO?

The number of correct guesses x cannot be larger than the number of the incorrect guesss y. Let W be the number of winning and L the number of loosing outcomes. Note that $W + L = 2^n$. The strategy that minimizes L concentrates the y incorrect guesses into the L loosing outcomes and spreads the x correct guesses over the W winning outcomes.

$$L \cdot n = y \geqslant x = W$$

Since $W + L = 2^n$, we have

$$W \leqslant \frac{2^n \cdot n}{n+1} \quad \text{and} \quad L \geqslant \frac{2^n}{n+1}$$

$\Rightarrow$ The highest probability of winning is $\frac{n}{n+1}$.

WHY IS THE HAMMING CODE STRATEGY OPTIMAL?

- Consider the example when $n = 3$ and the repetition code.
- Note that the prisoners will lose the game each time the hat color assignment corresponds to a codeword.
- Prove that the prisoners will win the game each time the hat color assignment corresponds to a non-codeword.

$\Rightarrow$ The probability of wining is[4]

$$\frac{2^n - 2^k}{2^n} = \frac{2^k(2^{n-k} - 1)}{2^{n-k}} = \frac{2^r - 1}{2^r} = \frac{n}{n+1}.$$

Therefore the Haamming code strategy achieves the highest probability of winning.
(You may want to see a NY Times article on this problem.)

[4] Recall the definition of Hamming codes with parameter r from the last lecture.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #6, February 7

This lecture introduces dual codes and code weight enumerators.

Dual Code of a Linear Code

Recall that an $[n,k]_q$ linear code $\mathbb{C}$ is a k dimensional subspace of $\mathbb{F}_q^n$. Its dual code $C^\perp$ is its orthogonal complement:

$$\mathbb{C}^\perp = \{x \in \mathbb{F}_q^n \mid cx^T = 0 \; \forall c \in C\}$$

Which codeword of the dual code you already know?
The parity-check matrix of C generates $C^\perp$.
What is the parity-check matrix of $C^\perp$? What is its dimension?
What is the dual of the dual code?

The dimension of C and its dual add up to the the dimension of the vector space where they complement each other, that is $\mathbb{F}_q^n$:

$$\dim C + \dim C^\perp = n.$$

$\Longrightarrow C^\perp$ is an $[n, n-k]_q$ linear code.
Note that a code can be *self-dual*.

The $[7,3]$ *Simplex Code*

The parity check matrix of a code is the generator matrix of its *dual code*. The dual code to the Hamming $[7,4]$ code is the $[7,3]$ Simplex code. Therefore, its generator matrix is

$$G = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

It encodes 3 data symbols a, b, and c into 7 coded symbols:

$$\begin{bmatrix} c & b & a \end{bmatrix} \cdot G = \begin{bmatrix} a & b & a+b & c & a+c & b+c & a+b+c \end{bmatrix}$$

We will talk more about dual codes and simplex codes in later classes.

Code Weight Distribution and Enumerator

The weight distribution consists of numbers A_w that count codewords of weight w:

$$A_w = \big|\{c \in C \mid w_H(c) = w\}\big|.$$

[1] Rutgers, ECE 548, Spring 2019

What is $\sum_w A_w$ equal to?
The weight enumerator is the homogeneous, bivariate polynomial:

$$W_C(x,y) = \sum_{w=0}^{n} A_w x^w y^{n-w}.$$

For the $[7,4]$ Hamming code, we have $W_C(x,y) = x^7 + 7x^4y^3 + 7x^3y^4 + y^7$. Note that $A_0 = 1$ and $A_1 = \cdots = A_{d-1} = 0$.

The distance distribution of an $[n,k]$ code C consists of numbers B_ℓ that cont pairs of codewords at distance ℓ.

$$B_\ell = \frac{1}{|C|}\left|\{(c_1,c_2) \in C \times C \mid d(c_1,c_2) = \ell\}\right| \qquad 0 \leqslant \ell \leqslant n.$$

The distance enumerator polynomial is

$$A_C(x,y) = \sum_{\ell=0}^{n} B_\ell x^\ell y^{n-\ell}$$

We can use the weight enumerator, e.g., to evaluate P_u, the probability of undetected error on the BSC with the crossover probability p:

$$P_u = \sum_{w>0}^{n} A_w p^w (1-p)^{n-w}.$$

Note that when $\mathbb{C}$ is a linear code, than the distance and weight enumerators are identical.
Recall that the probability of error for an $[n,k,d]$ code is

$$P_e = \sum_{\ell=t+1}^{n} \binom{n}{\ell} p^\ell (1-p)^{n-\ell} \text{ where } t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

The MacWilliams identity relates the weight enumerator of the code and its dual:

$$W_{C^\perp}(x,y) = \frac{1}{|C|} W_C(y-x, y+x).$$

Homework - due February 25

Evaluate the performance of the $[7,4]$ Hamming code on the BSC channel: Plot P_u and P_e as a function of the BSC crossover probability p for $p \in [0, 0.5]$

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #7, February 19

This lecture is about the Reed-Solomon codes.

Finite Fields

A finite field or Galois field (so-named in honor of Évariste Galois) is a field that contains a finite number of elements. So far we worked with GF(2) (or $\mathbb{F}_2$, or the binary field). We can get GF(4) as an extension of GF(2). To define field extensions, we use irreducible polynomials.

A polynomial is irreducible over $\mathbb{F}$ if its coefficients belong to $\mathbb{F}$ and it cannot be factored into the product of two non-constant polynomials with coefficients in $\mathbb{F}$. Do you know an irreducible polynomial in $\mathbb{R}$?

We can interpret GF(4) (or $\mathbb{F}_{2^2}$) as a quadratic extension[2] of $\mathbb{F}_2$ by the roots of the irreducible polynomial $x^2 + x + 1$. Note that if α is a root of this polynomial, so is $1 + \alpha$. Therefore, GF(4) consists of the set $\{0, 1, \alpha, 1 + \alpha\}$ and two operations:

$+$	0	1	α	$1+\alpha$
0	0	1	α	$1+\alpha$
1	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$1+\alpha$	$1+\alpha$	α	1	0

$\cdot$	0	1	α	$1+\alpha$
0	0	0	0	0
1	0	1	α	$1+\alpha$
α	0	α	$1+\alpha$	1
$1+\alpha$	0	$1+\alpha$	1	α

A primitive element of a finite field GF(q) is a generator of the multiplicative group of the field. Note that $\alpha^2 = 1 + \alpha$ and $\alpha^3 = 1$. Therefore, α is the primitive element of GF(4).

Reed-Solomon Codes

Reed-Solomon codes are one of the most important family of codes both in theory and in practice. They were introduced in

Reed, Irving S.; Solomon, Gustave (1960), SIAM 8 (2)
"Polynomial Codes over Certain Finite Fields"

[1] Rutgers, ECE 548, Spring 2019

[2] Recall that $\mathbb{C}$ is the quadratic extension of $\mathbb{R}$ by the roots of the polynomial $x^2 + 1$.

Definition – The Evaluation and the Generator Matrix View

There are many ways to define Reed-Solomon codes, and we will start as follows:
$\mathrm{RS}[n,k]_q \subseteq \mathbb{F}_q^n$ where $q \geqslant n$ (we will use $n = q-1$) is defined for a set of points $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$ as the following set of n-dimensional vectors over $\mathbb{F}_q$:

$$\begin{aligned}\mathrm{RS}[n,k]_q = \{&(\mathbf{a}(\alpha_1), \mathbf{a}(\alpha_2), \ldots, \mathbf{a}(\alpha_n)) \in \mathbb{F}^n \mid \\ &\mathbf{a} \text{ is a polynomial over } \mathbb{F}_q \text{ of degree } < k\}\,.\end{aligned}$$

To encode data vector $a = (a_0, a_1, \ldots, a_{k-1}) \in \mathbb{F}_q^k$, we take its symbols to be the coefficients of the polynomial

$$\mathbf{a}(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1} \in \mathbb{F}_q[x].$$

We then evaluate this polynomial at the points $\alpha_1, \alpha_2, \ldots, \alpha_n$ to get the codeword corresponding to a. Note that the i-th codeword symbol is $\mathbf{a}(\alpha_i)$.

Equivalently, to evaluate the polynomial $\mathbf{a}$ on the points $\alpha_1, \ldots, \alpha_n$, we multiply the message vector a by the following $k \times n$-matrix G:

$$G = \begin{bmatrix} 1 & \ldots & 1 \\ \alpha_1 & \ldots & \alpha_n \\ \alpha_1^2 & \ldots & \alpha_n^2 \\ \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \ldots & \alpha_n^{k-1} \end{bmatrix}$$

Note that the matrix G is a generator matrix for $\mathrm{RS}[n,k]_q$, i.e., $\boxed{c = aG}$ $\Longrightarrow$ Reed-Solomon codes are linear.

The Minimum Distance

A codeword in $\mathrm{RS}[n,k]_q$ can have at most $k-1$ zeros. Why? Thus the minimum weight[3] a codeword in $\mathrm{RS}[n,k]_q$ can have is $n-(k-1)$. Because RS codes are linear, the minimum distance is equal to the minimum weight a codeword can have. Therefore, the minimum distance of $\mathrm{RS}[n,k]_q$ is at least $n-k+1$. On the other hand, by the Singleton bound, we know that the minimum distance is at most $n-k+1$. Therefore, the minimum distance is exactly equal to $n-k+1$, and thus the RS codes are MDS, i.e., they achieve the Singleton bound.

Here is another way to show that the minim distance of RS codes is equal to $n-k+1$. Consider a $k \times k$ matrix V that consists of some k columns of G, say $i_1, i_2, \ldots i_k$. Compute the determinant of V. Note that V is a Vandermonde matrix. Its determinant is given by

$$\prod_{i_\ell < i_m} (\alpha_{i_\ell} - \alpha_{i_m}) \neq 0.$$

[3] number of non-zero elements

Suppose that in the codeword $c = a \cdot G$ some $n-k$ symbols are erased. We can still recover the data a because the remaining symbols of c are related to data a by an invertible matrix.

Definition – The Parity Check Matrix View

Polynomial representations:

- Data word $a = (a_0, a_1, \dots, a_{k-1})$:

$$a(x) = a_0 + a_1 x + \dots + a_{k-1} x^{k-1}$$

- Codeword $c = (c_0, c_1, \dots, c_{n-1})$:

$$c(x) = c_0 + c_1 x + \dots + c_{n-1} x^{n-1}$$

We can now define the $\text{RS}[n, k]_q$ as the set of all codewords whose associate polynomials have degree at most n and are multiples of the $\underline{\text{generator polynomial}}$

$$g(x) = (x - \alpha^{j_0})(x - \alpha^{1+j_0}) \dots (x - \alpha^{n-k-1+j_0}), \quad j_0 \geqslant 1$$

where α is a primitive element of $\mathbb{F}_q$. What do we need j_0 for?[4]

Why is this a parity-check point of view? Note that $c(\alpha^{j_0+i}) = 0$ for $i = 0, \dots, n-k+1$. Let $j_0 = 1$ and consider:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \dots & (\alpha^2)^{n-1} \\ \vdots & \ddots & \vdots & & \\ 1 & \alpha^{n-k} & (\alpha^{n-k})^2 & \dots & \alpha^{n-k})^{n-1} \end{bmatrix}$$

Note that $c(\alpha^{i+j_0}) = 0$ for $0 \leqslant i \leqslant n-k+1$, hence the parity view. Then $\boxed{cH^T = 0}$. Here we see that a dual code of an MDS codes is an MDS code.

Encoding

1. By multiplication $c(x) = \mathbf{a}(x) \cdot g(x)$
2. $c(x) = x^{n-k} a(x) + b(x)$
   where $b(x)$ is the remainder resulting from dividing $x^{n-k} a(x)$ by $g(x)$.

Note that $\deg(b(x)) < \deg(g(x)) = n - k$.
Note that the codeword generated by the encoding method 2, consists of data symbols and coefficients of $b(x)$. Thus we have a systematic code.

[4] Choice of j_0, and the encoding method impacts the complexity of the encoding circuitry.

Two Research Problems Related to MDS Codes

MDS Codes with Constrained Generator Matrices

1. Are there RS codes whose generator matrices have zeros at prescribed places?

2. If there are, how large field do we need?

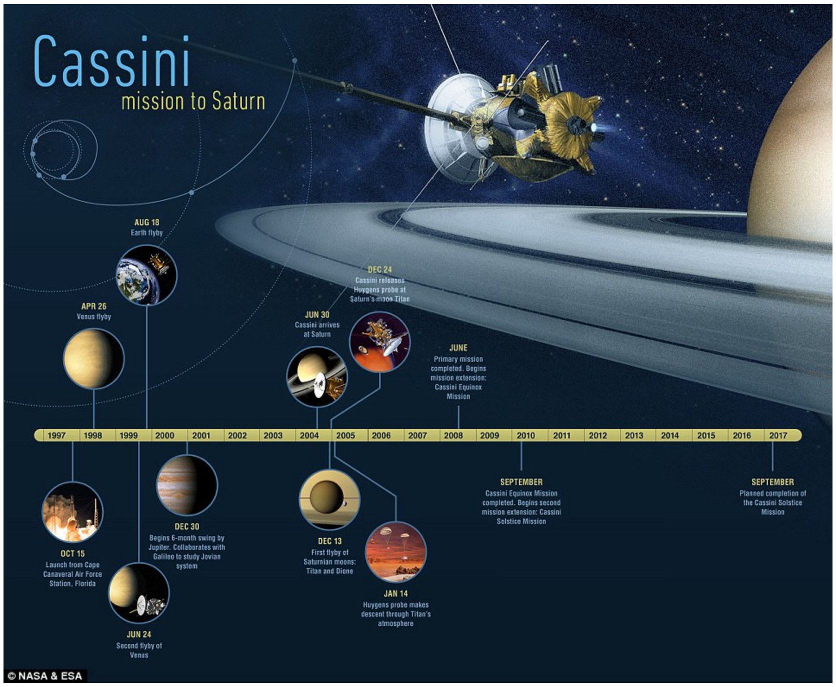
3. Can the zeros be anywhere? Example, a systematic MDS code?

The MDS Conjecture

The Main Conjecture on MDS Codes states that for every linear $[n, k]_q$ MDS code, if $k \leqslant q$, then $n \leqslant q + 1$, except when q is even and $k = 3$ or $k = q - 1$, in which cases $n \leqslant q + 2$. This problem is a long standing open problem, and is related to some problems in algebraic geometry.[5]

[5] Problems of Segre

RS Codes in Action


Cassini
mission to Saturn
AUG 18
Earth flyby
APR 26
Venus flyby
JUN 30
Cassini arrives at Saturn
DEC 24
Cassini releases Huygens probe at Saturn's moon Titan
JUNE
Primary mission completed. Begins mission extension: Cassini Equinox Mission
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
OCT 15
Launch from Cape Canaveral Air Force Station, Florida
JUN 24
Second flyby of Venus
DEC 30
Begins 6-month swing by Jupiter. Collaborates with Galileo to study Jovian system
DEC 13
First flyby of Saturnian moons: Titan and Dione
JAN 14
Huygens probe makes descent through Titan's atmosphere
SEPTEMBER
Cassini Equinox Mission completed. Begins second mission extension: Cassini Solstice Mission
SEPTEMBER
Planned completion of the Cassini Solstice Mission

© NASA & ESA

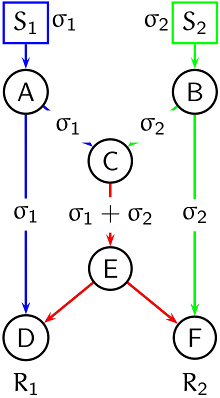
NETWORK CODED MULTICAST

ERROR CONTROL CODING (ECE 548)

Rutgers, Spring 2019

Lecture #8, February 21

NETWORK MULTICAST – The Butterfly



- Sources S_1 and S_2 produce bits σ_1 and σ_2.
- Each receiver needs bits from both sources.
- The edges have unit capacity.

Can both sources simulaneosly transmit to both reseivers?

Yes if nodes can XOR bits.

NETWORK MULTICAST MODEL

- Network is represented as a directed, acyclic graph.
- Edges have unit-capacity and parallel edges are allowed.
- There are h unit-rate information sources $S_1, \ldots, S_h$.
- There are N receivers $R_1, \ldots, R_N$ located at N distinct nodes.

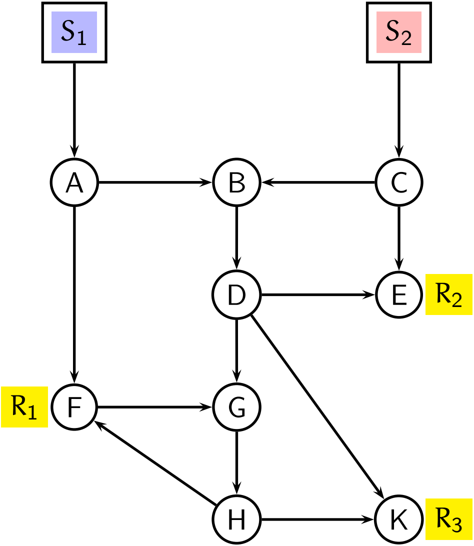
Can all sources simultaneously transmit at full rate to all receivers?

NETWORK MULTICAST – Throughput

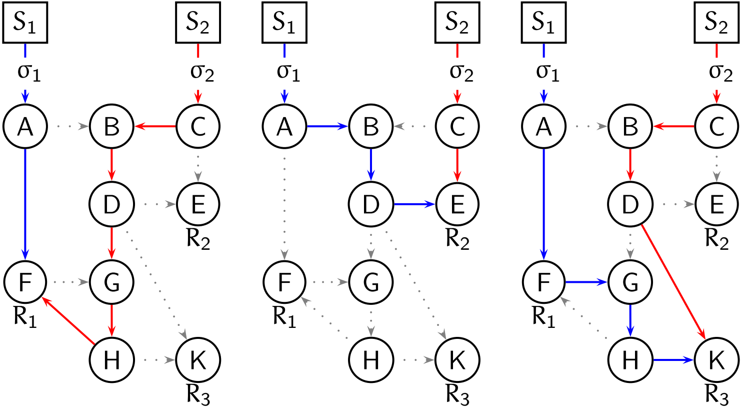
- Can all sources simultaneously transmit to receiver R_j?
  Yes, if between the sources and the j-th receiver node
  - the number of edges in the min-cut is h (or equivalently)
  - there are h edge-disjoint paths (S_i, R_j) for $1 \leqslant i \leqslant h$.

  [Ford, Fulkerson], [Elias, Feinstein, Shannon] $\sim$ 50s
- Can all sources simultaneously transmit to all receivers?
  Yes, if in addition each node of G can re-encode information.
  [Alshwede, Cai, Li, Yeung] $\sim$ 2000

A Network for Multicast

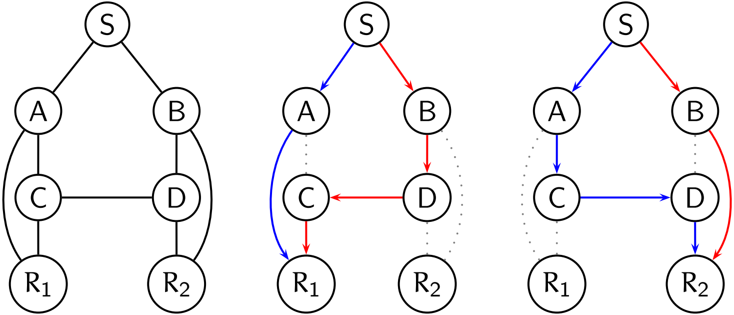


Three Unicasts in a Multicast Network



UNDIRECTED GRAPHS

- The main theorem does not hold.
- Coding can at most double the throughput.

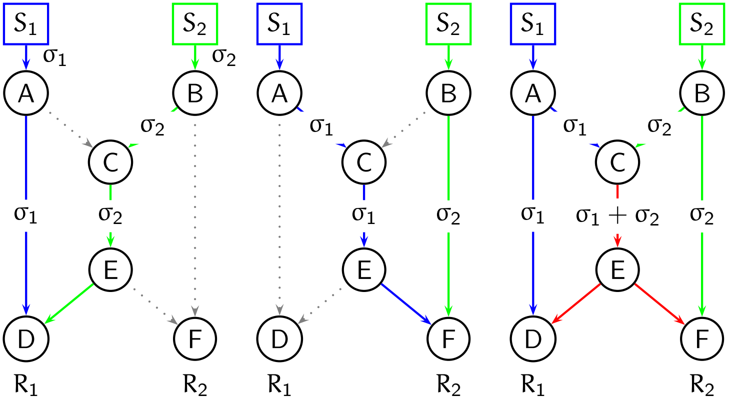


Original Graph

Paths to R_1

Paths to R_2

NETWORK MULTICAST – Linear Combining



(a) Routing to R_1 (b) Routing to R_2 (c) Network coding

Network Multicast Theorem

Conditions:

- Network is represented as a directed, acyclic graph.
- Edges have unit-capacity and parallel edges are allowed.
- There are h unit-rate information sources $S_1, \ldots, S_h$.
- There are N receivers $R_1, \ldots, R_N$ located at N distinct nodes.
- Between the sources and each receiver node,
  - the number of edges in the min-cut is h (or equivalently)
  - there are h edge-disjoint paths (S_i, R_j) for $1 \leqslant i \leqslant h$.

Claim: There exists a multicast transmission scheme of rate h.

Moreover, multicast at rate h

- cannot always be achieved by routing, but
- can be achieved by allowing the nodes to linearly combine their inputs over a sufficiently large finite field.

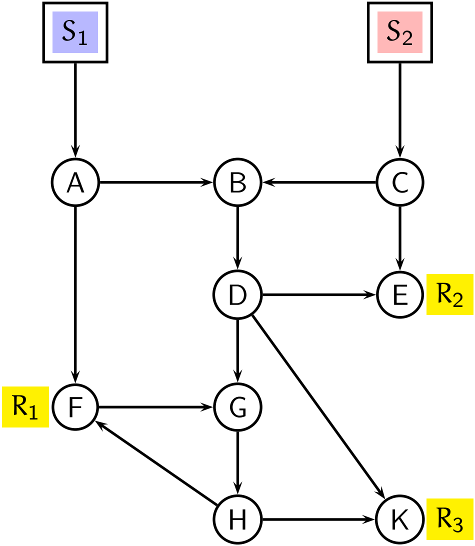
Network Multicast – Linear Combining

- Source S_i emits σ_i which is an element of some finite field.
- Edges carry linear combinations of their parent node inputs.
- Consequently,
  edges carry linear combinations of source symbols σ_i.

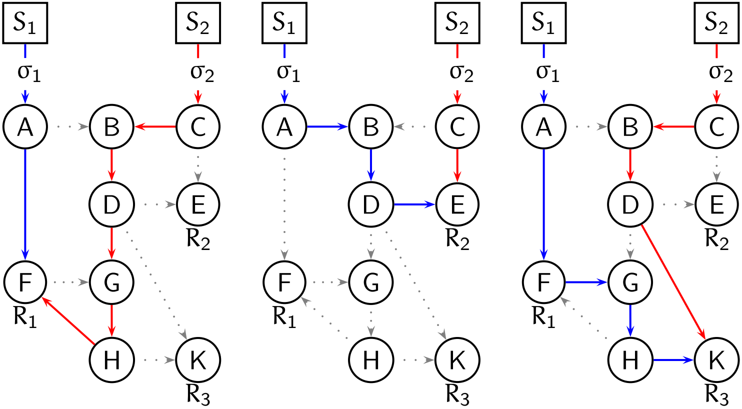
Network Coding Multicast Problem:

How should nodes combine their inputs to ensure that any h edges observed by a receiver carry independent combinations of σ_i-s?

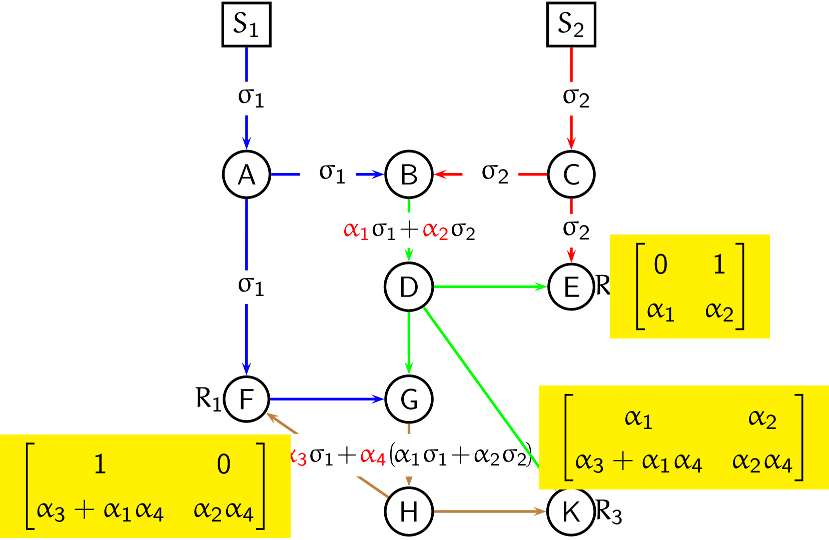
Network Multicast – Example



Network Multicast – Example



Network Multicas – Example



S_1

S_2

σ_1

σ_2

A

σ_1

B

σ_2

C

$\alpha_1\sigma_1+\alpha_2\sigma_2$

σ_2

D

E

R

$$\begin{bmatrix} 0 & 1 \\ \alpha_1 & \alpha_2 \end{bmatrix}$$

σ_1

R_1

F

G

$$\begin{bmatrix} \alpha_1 & \alpha_2 \\ \alpha_3+\alpha_1\alpha_4 & \alpha_2\alpha_4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ \alpha_3+\alpha_1\alpha_4 & \alpha_2\alpha_4 \end{bmatrix}$$

$\alpha_3\sigma_1+\alpha_4(\alpha_1\sigma_1+\alpha_2\sigma_2)$

H

K

R_3

Network Multicast – Code Design

- Edges carry linear combinations of their parent node inputs; $\{\alpha_k\}$ are the coefficients used in these linear combinations.
- ρ_i^j is the symbol on the last edge of the path (S_i, R_j) ⇒ Receiver j has to solve the following system of equations:

$$\begin{bmatrix} \rho_1^j \\ \vdots \\ \rho_h^j \end{bmatrix} = \mathbf{C}_j \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_h \end{bmatrix}$$

where the elements of matrix $\mathbf{C}_j$ are polynomials in $\{\alpha_k\}$.

The Code Design Problem:

Select $\{\alpha_k\}$ so that all matrices $\mathbf{C}_1 \ldots \mathbf{C}_N$ are full rank.

Network Multicast – Code Existence

- The goal is to select $\{\alpha_k\}$ so that $\mathbf{C}_1 \ldots \mathbf{C}_N$ are full rank.
- Equivalently, the goal is to select $\{\alpha_k\}$ so that

$$f(\{\alpha_k\}) \triangleq \det(\mathbf{C}_1) \cdots \det(\mathbf{C}_N) \neq 0.$$

Can such $\{\alpha_k\}$ be found?

RLNC [Ho et al.]

Yes, by selecting $\{\alpha_k\}$ uniformly at random from a "large filed", we will have the polynomial $f(\{\alpha_k\}) \neq 0$ with "high probability".

LIF [Jaggi et al.]

Yes, $\{\alpha_k\}$ can be selected form $\mathbb{F}_q$ where $q > N$.

But, we don't know of any networks for which $q > \mathcal{O}(\sqrt{N})$ is required.

THE OPERATING FIELD SIZE

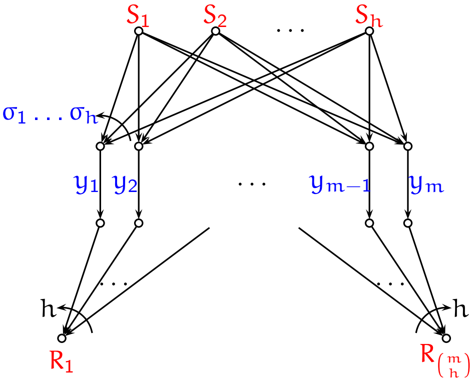
- Why not any finite field?
- Polynomial

  $$x(x+1)+x+x^2$$

  is identically equal to zero over any field with characteristic 2.
- A polynomial not identically equal to zero over $\mathbb{F}_q$ can evaluate to zero on all elements of $\mathbb{F}_q$.
- Consider polynomial $x(x+1)$ over $\mathbb{F}_2$

Combination Network $B(h, m)$

A Popular Network With a Small-Alphabt Code



$B(h, m)$ has

- h information sources,
- $\binom{m}{h}$ receivers, and
- m bottlenecks.

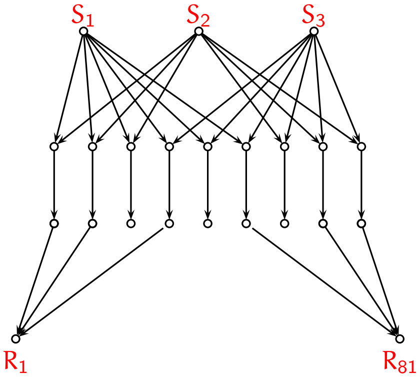
Design a rate-h multicast!

Map $\{\sigma_j\}$ to $\{y_k\}$ by an $[m, h]$ Reed-Solomon code.

But, what if fewer than h sources are available at the bottlenecks?

A Distributed Combination Network

Fewer than h sources are available at the bottlenecks



There are

- 3 information sources,
- 9 bottlenecks, and
- $\binom{9}{3} - 3$ receivers.

Design a rate-3 multicast!

Only information that is locally available can be combined.

Non-Monotonicity

There may be a solution over $\mathbb{F}_{q_0}$ but not over $\mathbb{F}_q$ for some $q > q_0$

Coding vectors for our example network:

$$\left[\begin{array}{ccc|ccc|ccc} a_1 & a_2 & a_3 & b_1 & b_2 & b_3 & 0 & 0 & 0 \\ c_1 & c_2 & c_3 & 0 & 0 & 0 & d_1 & d_2 & d_3 \\ \underbrace{0 \quad 0 \quad 0}_{v_1} & & & \underbrace{e_1 \quad e_2 \quad e_3}_{v_2} & & & \underbrace{f_1 \quad f_2 \quad f_3}_{v_3} & & \end{array}\right]$$

All 3×3 sub-matrices, except v_1, v_2, v_3, should be non-singular.

In which fields $\mathbb{F}_q$ does a solution exist?

- **No** solution exists when $q < 7$.
- A solution exists for all $q \geqslant 9$.
- A solution exists for $q = 7$
- **No** solution exists for $q = 8$.

What Would We Like To Do?

... short of solving the problem ...

Find relations (equivalences) with other problems, e.g.,

Something old :

Three problems of Segre in $\mathbb{PG}(h-1, q)$

1. What is the size $g(h, q)$ of the maximal arc,
   and which arcs have $g(h, q)$ points?
2. For which q and $h < q$ are all arcs with $q + 1$ points equivalent?
3. What are the sizes of the complete arcs,
   and what is the size of the second largest complete arc?

Something new :

constrained MDS codes, codes with locality constraints,
minimal multicast graph topologies vs. geometry of arcs.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #9, February 26

> This lecture covers several fundamental urns and balls problems we use in coding theory.

Urns and balls models refer to basic probabilistic experiments in which balls are thrown randomly into urns, and we are interested in various patterns of urn occupancy (e.g., the number of empty urns). These models are central in many disciplines such as combinatorics, statistics, analysis of algorithms, and statistical physics. Some modern network communications scenarios give rise to problems that are related to the classical urns and balls questions. Some new models and problems emerge as well, because information packets can be processed in a way their physical counterparts, urns and balls, cannot.

... any problem of probability appears comparable to a suitable problem about bags containing balls, and any random mass phenomenon appears as similar in certain essential respects to successive drawings of balls from a system of suitably combined bags.
Polya in "Mathematics and Plausible Reasoning" 1954.

Urns&Balls and Coupons – Classical Probability Models

Two equivalent experiments:

1. b balls are thrown into n urns, e.g., $b = 140$ and $n = 365$.

2. b objects are drawn with replacement from a set of size n.

Questions (many can be asked):

How many draws (balls) it takes for some event $\mathcal{E}$ to occur, e.g.,

- $\mathcal{E}_b$: a coupon (any) gets drawn twice (an urn is hit twice) ← birthday problem

- $\mathcal{E}_c$: each coupon gets drawn at least once (all urns are nonempty) ← coupon collection

Estimates (some are known):

- For $\mathcal{E}_b$, we need $\mathcal{O}(\sqrt{n})$ draws on average.

- For $\mathcal{E}_c$, we need $\mathcal{O}(n \log n)$ draws on average.

Information Source

- produces sequences of letters in a finite alphabet $\mathcal{U}$

- outputs each letter independently of the rest

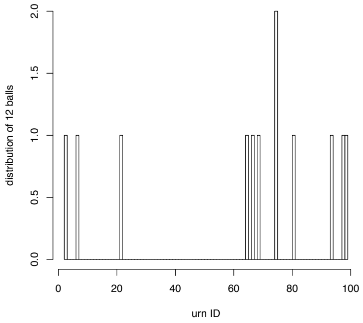
- probability of letter x is P_x

[1] Rutgers, ECE 548, Spring 2019

- e.g., coin tossing with $\mathcal{U} = \{H, T\}$, $P_H = P_T = 1/2$

Each source sequence of length b corresponds to

- an outcome of throwing b balls into $|\mathcal{U}|$ urns, or
- b draws, with replacement, form the set $\mathcal{U}$ of coupons.

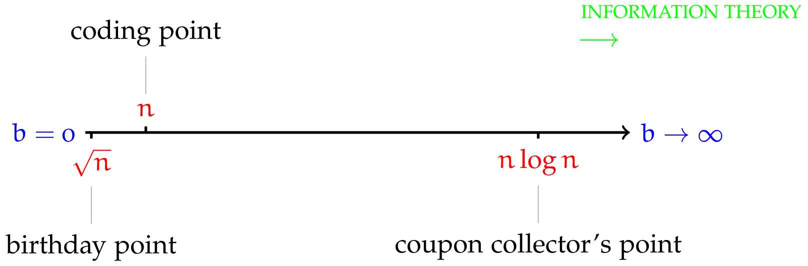


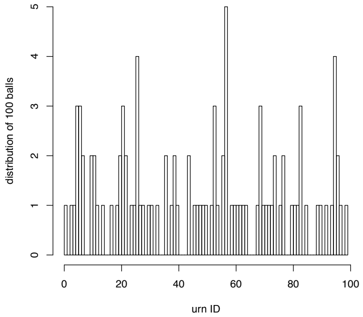
How do b Balls Get Distributed Among n Urns?

of Urns n vs. # of Balls b MATTERS

Suppose the number of urns n is given, e.g., $n = 1000$.

Which values of b are of interest?

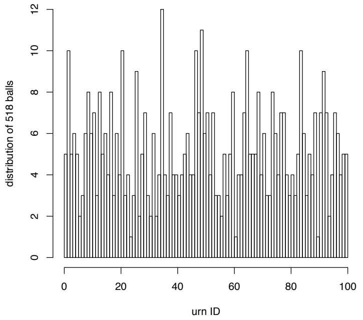




Time to Collect All Coupons

- The first draw brings a new coupon for sure.
- Suppose we have collected r different coupons. Then
  - a draw brings a new coupon with probability $p_r = (n - r)/n$,
  - the average number of draws to get a new coupon is

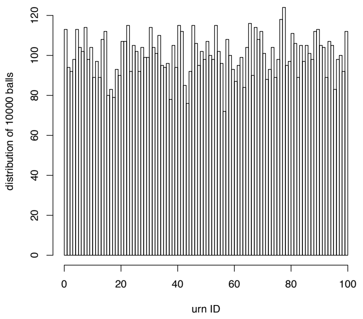
$$\sum_{\ell=1}^{\infty} \ell \cdot p_r (1 - p_r)^{\ell - 1} = \frac{1}{p_r} = \frac{n}{n - r}.$$



- The average number of draws to get all coupons is

$$\sum_{r=0}^{n-1} \frac{1}{p_r} = n\left(\frac{1}{n} + \frac{1}{n-1} + \cdots + 1\right) = nH_n$$

$H_n = \log n + \gamma + \mathcal{O}(n^{-1})$ is the harmonic number.
$\gamma = 0.5772156649$ is the Euler's constant.



Average Time to Collect Some Coupons

$\sum_{r=0}^{k-1} \frac{n}{n-r} = n(H_n - H_{n-k})$ *for ANY* k *Coupons:*

Once r different coupons have been collected, it takes $n/(n-r)$ draws on average to get one of the unseen $n-r$ coupons.

$\sum_{r=0}^{k-1} \frac{n}{k-r} = nH_k$ *for SPECIFIC* k *coupons:*

Once r different coupons have been collected, it takes $n/(k-r)$ draws on average to get one of the remaining $k-r$ desired coupons.

How do b Balls Get Distributed Among n Urns?

Let μ_r be the number of urns containing r balls. Are μ_r independent?
Note that $\sum_{r=1}^{n} \mu_r = n$ and $\sum_{r=1}^{n} r \cdot \mu_r = b$.
We can compute $E(\mu_r)$:

- $\theta_r^\ell \in \{0, 1\}$ indicates if the ℓ-th urn has r balls or not
- $\mu_r = \sum_{\ell=1}^{n} \theta_r^\ell \Rightarrow$

$$E(\mu_r) = E\left\{\sum_{\ell=1}^{n} \theta_r^\ell\right\} = \sum_{\ell=1}^{n} P(\theta_r^\ell = 1) = n\binom{b}{r}\left(\frac{1}{n}\right)^r\left(1 - \frac{1}{n}\right)^{b-r}$$

b draws will, on average, bring $n - E(\mu_0)$ different coupons,

$$n - E(\mu_0) = n\left[1 - (1 - 1/n)^b\right] \gtrapprox n\left(1 - e^{-b/n}\right).$$

A Brotherhood Problem

Collecting all n coupons with a little sister:

- The collector has a little sister to whom he gives his duplicates.
- When he collects all coupons, how many does she miss on average?

After b draws, the expected number of coupons with r copies is

$$E(\mu_r) = n\binom{b}{r}\left(\frac{1}{n}\right)^r\left(1 - \frac{1}{n}\right)^{b-r}$$

We can answer the following related question:

After $b = nH_n$ draws, how large are the siblings' collections on average?

- The brother's collection size is on average $n - E(\mu_0) \sim n$.
- The sister's collection size is on average $n - E(\mu_1) \sim n - H_n$.

How many additional draws will complete the sister's collection?

RAPTOR CODES
From a Math Idea to LTE eMBMS

ERROR CONTROL CODING (ECE 548)
Rutgers, Spring 2019

Lecture #10, February 28, 2019

The plan is to ...

1. introduce LT codes and Raptor codes
2. provide insights into their design
3. address some common misconceptions

The Ideal Abstract Properties of Codes

1. The encoder should be able to generate, from k source symbols, as many encoded symbols as required for decoding. ← rateless
2. ANY k encoded symbols should be sufficient for decoding.
3. Encoding/Decoding computation time should be linear in k.

Raptor Codes are Good for MBMS Because

They enable reliable communications over multiple, unknown channels:

1. They are rateless ⇒ their redundancy can be flexibly adapted to changing channel/network conditions of e.g. as in mobile wireless.
2. In the case of multiple erasure channels as in Multimedia Broadcast/Multicast Service (MBMS), they can be made to universally achieve the channel capacity for all erasure rates.

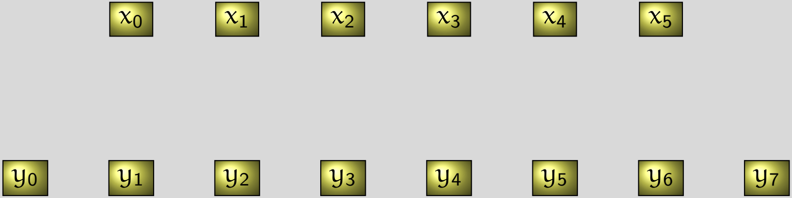
They are simple to encode and decode.

How do Raptor Codes achieve that, and can other codes perform as well?

Raptor Codes are Concatenated Codes

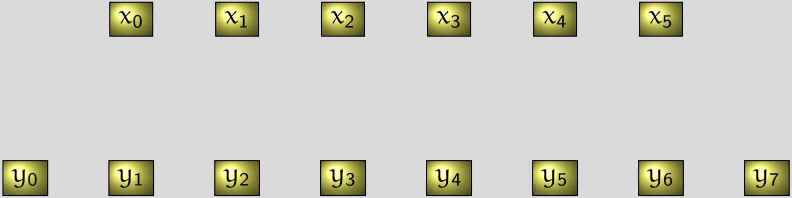
In the beginning, there were LT codes ...

LT Codes – Rateless Encoding



face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes – Rateless Encoding

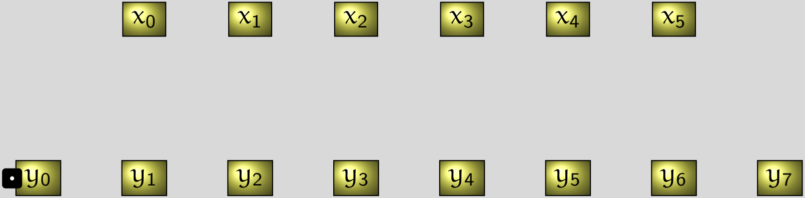

x_0 x_1 x_2 x_3 x_4 x_5

y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7

⚅⚅⚅⚅⚅⚅⚅⚅

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

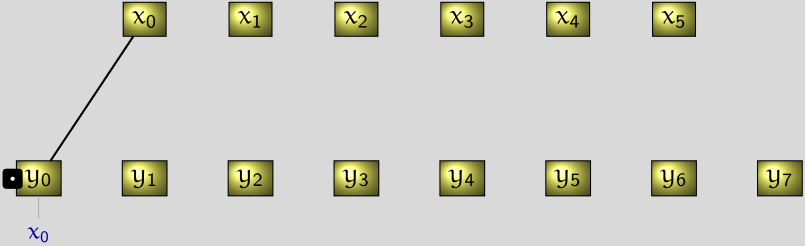
LT Codes – Rateless Encoding



⚅⚅⚅⚅⚅⚅⚅

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

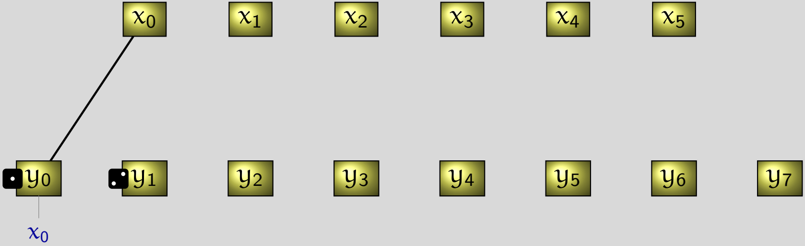
LT Codes – Rateless Encoding



⚅⚅⚅⚅⚅⚅⚅

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

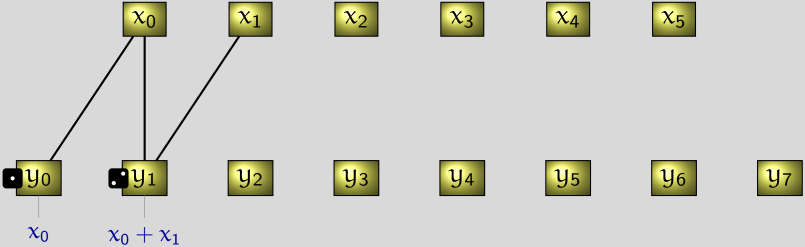
LT Codes – Rateless Encoding



⚅⚅⚅⚅⚅⚅

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

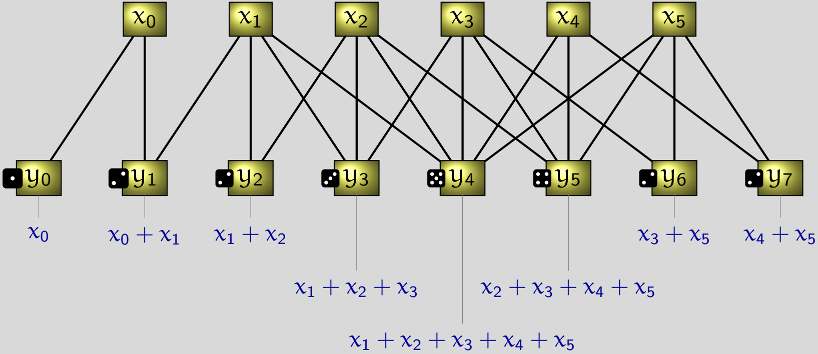
LT Codes – Rateless Encoding



⚅⚅⚅⚅⚅⚅

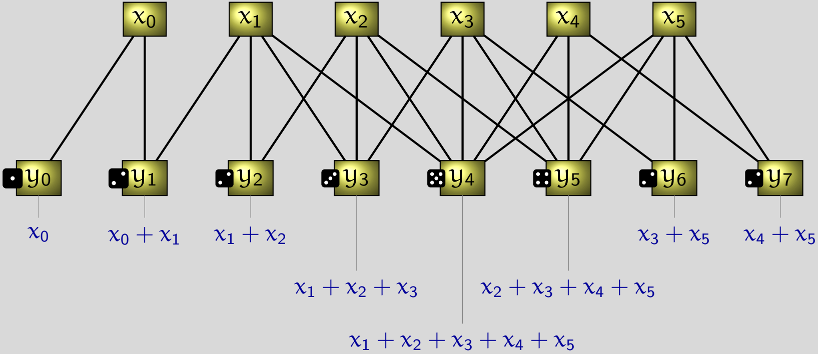
face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes – Rateless Encoding



face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes – Rateless Encoding



When can we hope to be able to recover the x values from y values?

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

Linear (Rateless) Codes

- Data symbols $\{x_1, \dots, x_k\}$ are mapped into code symbols $\{y_i\}_1^{\infty}$.
- y_i are linear combinations of $\{x_1, \dots, x_k\}$, $x_j, y_i \in \mathbb{F}_q$:

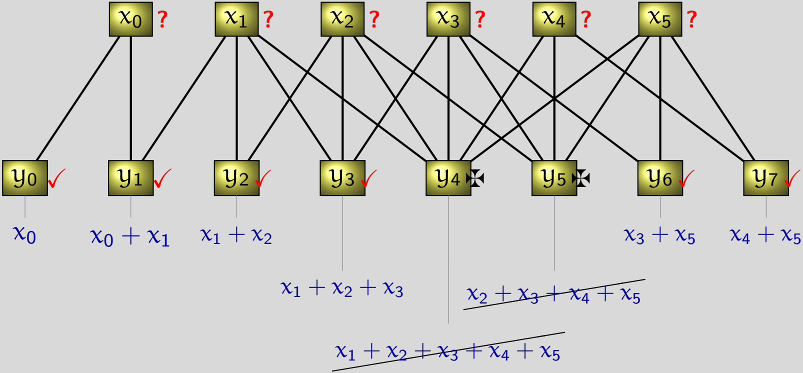
$$y_i = \alpha_1^i x_1 + \cdots + \alpha_k^i x_k, \quad \alpha_j^i \in \mathbb{F}_q$$

- For random codes, α_i are chosen uniformly at random.
- For each y_i in LT codes,
  1. degree d is picked (non-uniformly) at random from $\{1, \dots, k\}$, according to some degree distribution $\{p_1, \dots, p_k\}$;
  2. only d of $\{\alpha_1^i, \dots, \alpha_k^i\}$ are non-zero; picked uniformly at random.
  3. the values for the non-zero α_i are chosen uniformly at random.

  To design an LT code means to pick the degree distribution.

What are the unique advantages of LT Codes?

LT Codes – Simple (Belief Propagation) Decoding



x_0

$x_0 + x_1$

$x_1 + x_2$

$x_1 + x_2 + x_3$

~~$x_2 + x_3 + x_4 + x_5$~~

$x_3 + x_5$

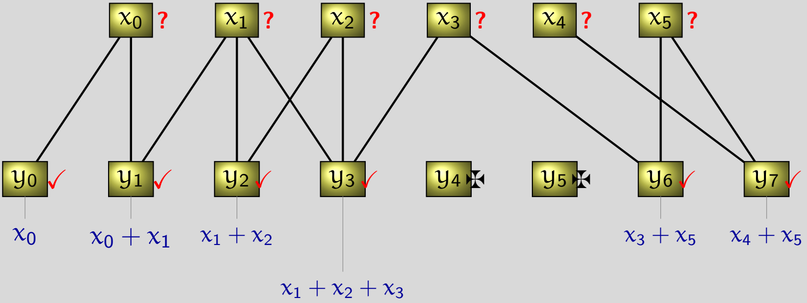
$x_4 + x_5$

~~$x_1 + x_2 + x_3 + x_4 + x_5$~~

⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

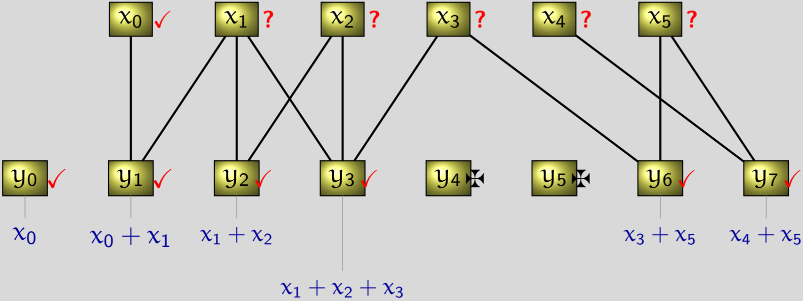
LT Codes – Simple (Belief Propagation) Decoding



⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

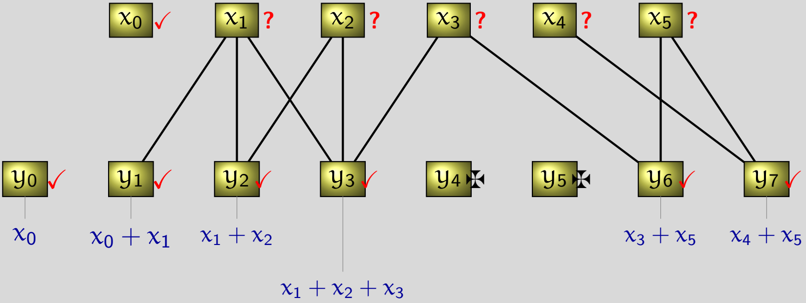
LT Codes – Simple (Belief Propagation) Decoding



⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

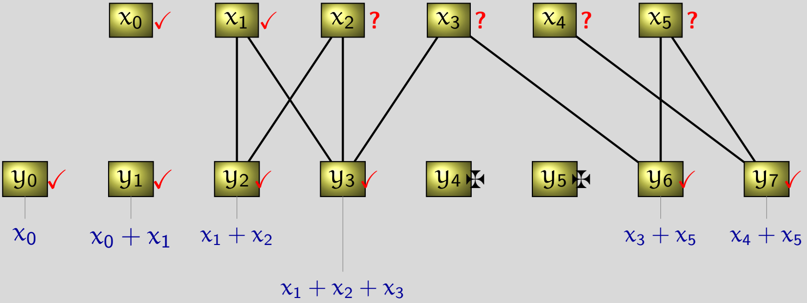
LT Codes – Simple (Belief Propagation) Decoding



⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

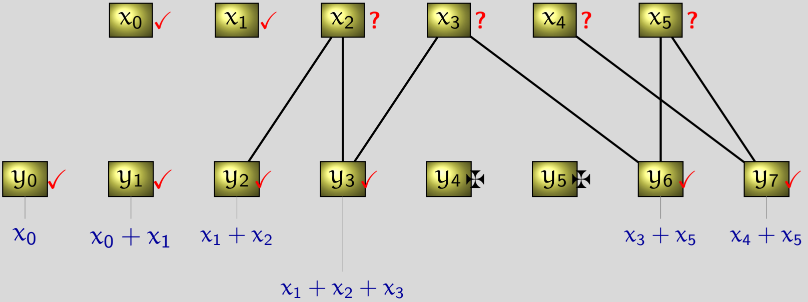
LT Codes – Simple (Belief Propagation) Decoding



⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

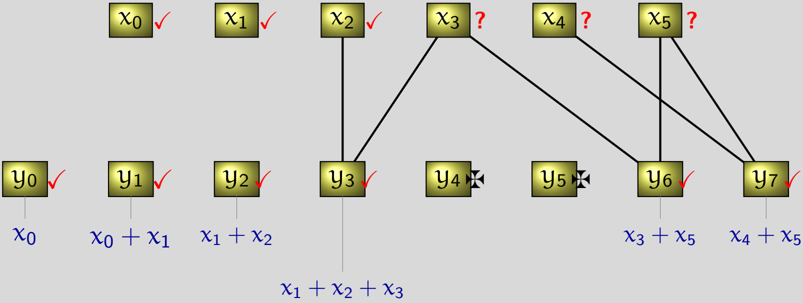
LT Codes – Simple (Belief Propagation) Decoding



⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

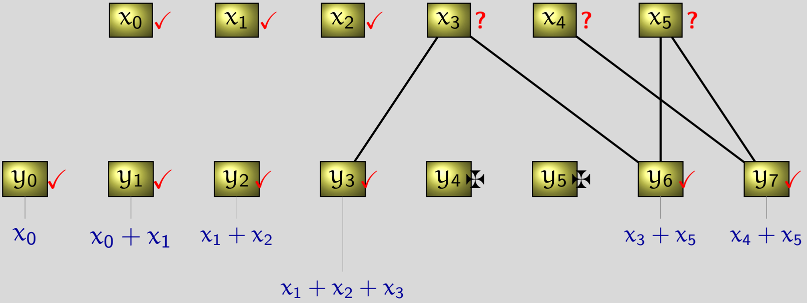
LT Codes – Simple (Belief Propagation) Decoding



⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes – Simple (Belief Propagation) Decoding

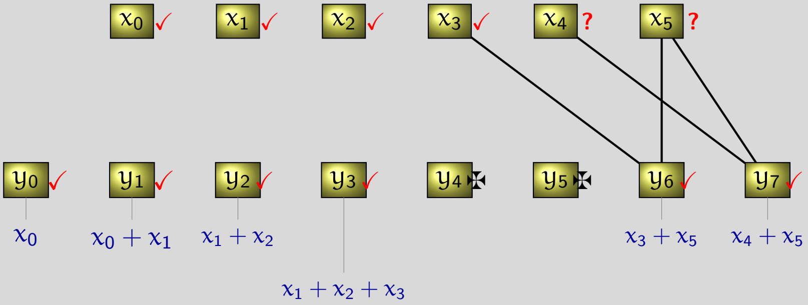


Complexity is determined by the number of edges in the decoding graph.

⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes – Simple (Belief Propagation) Decoding

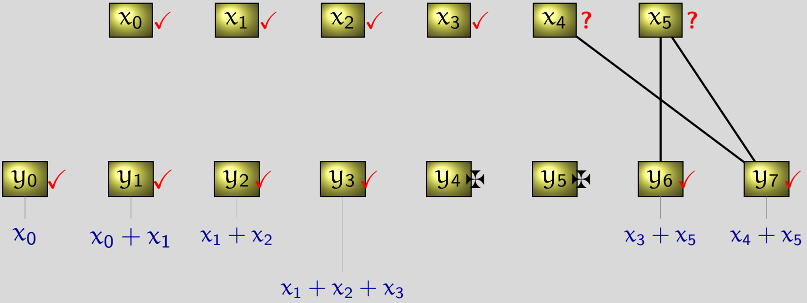


Complexity is determined by the number of edges in the decoding graph.

⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes – Simple (Belief Propagation) Decoding

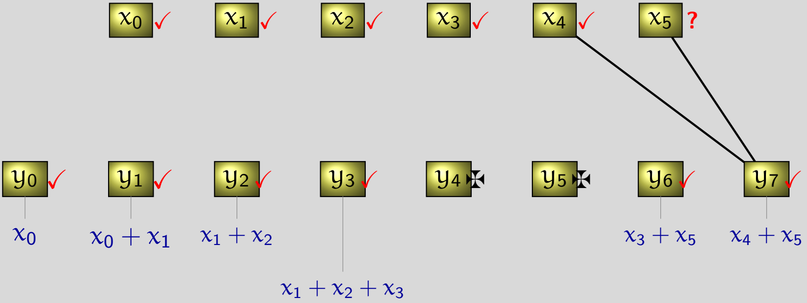


Complexity is determined by the number of edges in the decoding graph.

⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes – Simple (Belief Propagation) Decoding

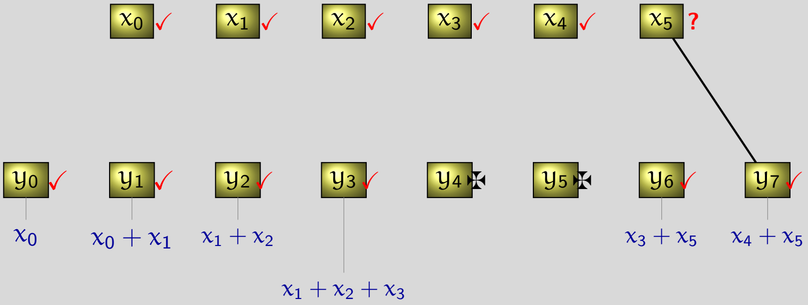


Complexity is determined by the number of edges in the decoding graph.

⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes – Simple (Belief Propagation) Decoding

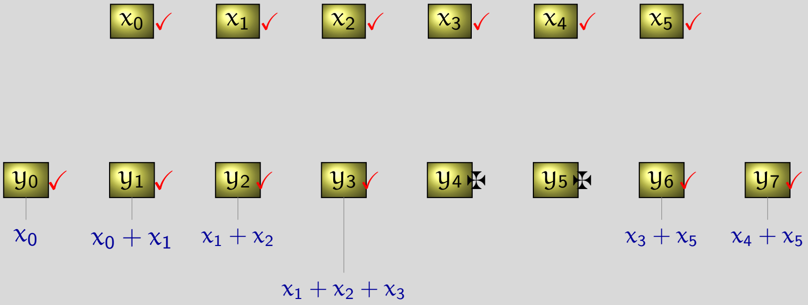


Complexity is determined by the number of edges in the decoding graph.

⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes – Simple (Belief Propagation) Decoding



Complexity is determined by the number of edges in the decoding graph.

⚀⚁⚁⚂⚄⚃⚁⚁

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes Design Objectives

- Data symbols $\{x_1, \dots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$.
- y_i are linear combinations of $\{x_1, \dots, x_k\}$, $x_j, y_i \in \mathbb{F}_q$:

$$y_i = \alpha_1^i x_1 + \cdots + \alpha_k^i x_k, \quad \alpha_j^i \in \mathbb{F}_q$$

- For each y_i in LT codes,
  1. degree d is picked (non-uniformly) at random from $\{1, \dots, k\}$, according to some degree distribution $\{p_1, \dots, p_k\}$;
  2. only d of $\{\alpha_1^i, \dots, \alpha_k^i\}$ are non-zero; picked uniformly at random.
  3. the values for the non-zero α_i are chosen uniformly at random.

The distribution should enable simple linear time decoding of $\{x_1, \dots, x_k\}$ as soon as any $k(1 + \epsilon)$ of y-s are received.

Can such a degree distribution be found?

Can LT Codes Meet the Ideal Design Objectives?

The Ideal Abstract Properties of Codes:

1. The encoder should be able to generate, from k source symbols, as many encoded symbols as required for decoding. ← rateless
2. ANY k encoded symbols should be sufficient for decoding.
3. Encoding/Decoding computation time should be linear in k.

vs.

LT Codes Design

1. Data symbols $\{x_1, \ldots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$ by an algorithm using a degree (probability) distribution $\{p_1, \ldots, p_k\}$.
2. The distribution should enable simple linear time decoding of $\{x_1, \ldots, x_k\}$ as soon as any $k(1+\epsilon)$ of y-s are received.

Can Complexity be Linear ...

if data symbols are recoverable from any $k(1+\epsilon)$ code symbols?

- Complexity is determined by the number of edges incident to y's but only for BP decoding; otherwise a matrix inversion is required.
- Data symbols $\{x_1, \ldots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$ by an algorithm using a degree (probability) distribution $\{p_1, \ldots, p_k\}$.
- We may be able to recover $\{x_1, \ldots, x_k\}$ from some n y's only if
  1. $n \geqslant k \Leftrightarrow$ more equations than unknowns
  2. each x is in at least one equation whp

     $\Leftrightarrow$ here are $\mathcal{O}(k \log k)$ edges incident to y's
- What else is required for decoding?

A Polya-Like Urn Model & Coding Overhead

An urn contains k balls that are numbered from 1 to k.

A multiple-ball draw from the urn is carried out in two stages:

1. Number i of balls to be drawn is selected with probability p_i.
2. i balls are drawn without replacements, their numbers are recorded, and the balls are then returned to the urn.

The number of draws necessary to see all balls is, on average,

$$\frac{1}{a_1} k H_k + \frac{a_1 - a_2}{2a_1^2}(H_k - 1),$$

where a_i is the i-th moment of $\{p_i\}_{i=1}^k$ and $H_k = \sum_{\ell=1}^{k} \frac{1}{\ell} = \mathcal{O}(\log k)$.

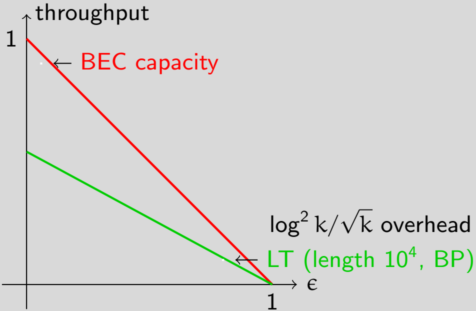
What About Coding Overhead?

When $\{x_1, \ldots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$ by using the following, Ideal Soliton, degree distribution:

$$p_d = \begin{cases} 1/k, & d = 1, \\ 1/[d(d-1)], & d = 2, 3, \ldots, k, \end{cases}$$

the variance in the decoding process will cause BP decoder to fail.

More robust distributions can guarantee a BP decoding failure rate of at most δ, when $k(1 + \log^2(k/\delta/\sqrt{k})$ or more y's are received.



The Raptor (Partial) Remedy

Soliton based LT code with BP decoding failure probability δ has

1. decoding complexity $\mathcal{O}(k \log(k/\delta))$
2. average overhead $\log^2(k/\delta)/\sqrt{k}$

Q: Can we recover all data in linear time from $\mathcal{O}(k)$ code symbols?

A: No, under our probabilistic model, but we can recover a large fraction.

Raptor codes are concatenated codes where

- data $\{x_1, \ldots, x_k\}$ are first pre-coded into $\{z_1, \ldots, z_m\}$, $m > k$, then
- precode symbols $\{z_1, \ldots, z_m\}$ are mapped into code symbols $\{y_i\}_1^\infty$

We don't have to recover all z's from $\mathcal{O}(k)$ received y's BUT just enough of them to allow us to decode all $\{x_1, \ldots, x_k\}$.

The Coupon Collection Problem

An urn contains k balls that are numbered from 1 to k.

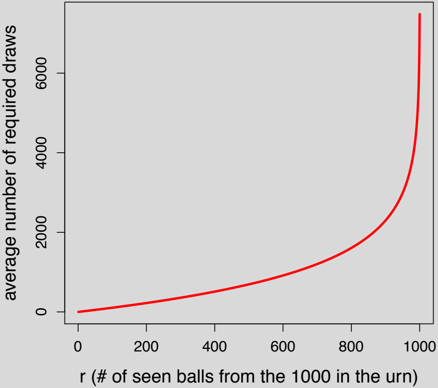
Balls are drawn one at the time with replacement.

The average number of draws reacquired to see r different balls:

$$\sum_{\ell=0}^{r-1} \frac{k}{k-\ell} = k(H_k - H_{k-r})$$

$$\sim \boxed{k \log \frac{k}{k-r}}$$

for large k and $k-r$.

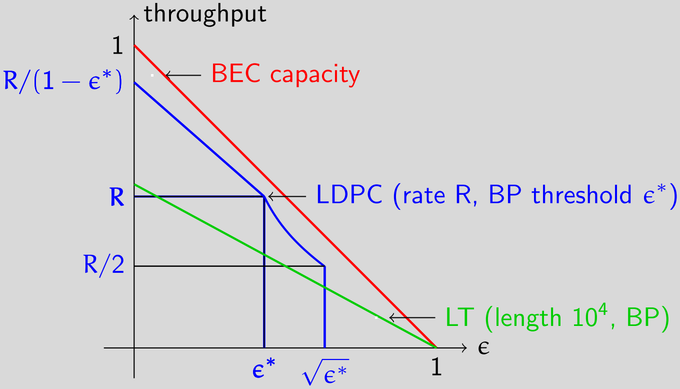


What is Ratelessness and is it Overrated?

Ratelessness matters and means different things to different people:

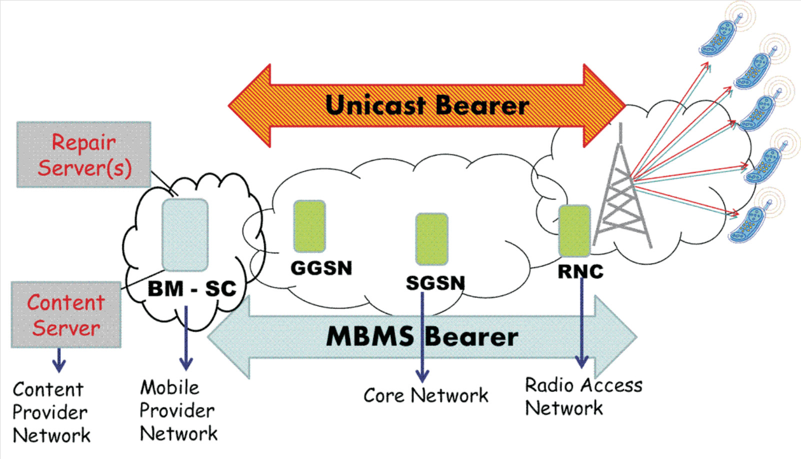
- encoder can produce potentially infinite stream of symbols
- each code symbol is statistically identical

Can fixed rate codes be made rateless?



The Unbearable Ratelessness of Coding

Two eMBMS Phases: Multicast Delivery & Unicast Repair:



Error Control Coding [1]

Prof. Emina Soljanin

Lecture #11, March 5

This lecture is concerned with random, linear (network) coding.

Collecting Numbers

You need to collect n numbers $x_1, \ldots, x_n$ in finite field $\mathbb{F}_q$ by drawing random linear combinations of the numbers (coding):

$$\rho = \alpha_1 x_1 + \cdots + \alpha_n x_n, \quad \alpha_i \in \mathbb{F}_q$$

(the coefficients α_i and the result of combining will be revealed). How many such draws do you need to make?[2]

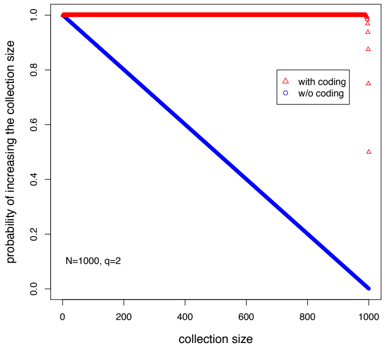
Rate of Collecting Coupons vs. Numbers

Depending whether or not we use coding, collection size is either

- the number of collected distinct pieces w/o coding, or
- the number of collected linearly independent equations w/ coding.

Suppose the collection size is $r < n$. Then a random draw will increase the collection size with probability

- $1 - r/n$ without coding,
- $1 - \frac{q^r}{q^n} > 1 - \frac{1}{q}$ with coding.



Time to Collecting Numbers

Once ℓ linearly independent combinations have been collected,

- a draw brings a new linearly independent equation with probability
$$p_\ell = (q^n - q^\ell)/q^n.$$
- the average number of draws that have to be made to get a new linearly independent equation is equal to $1/p_\ell = q^n/(q^n - q^\ell)$

The average number of draws to get r linearly independent equations is

$$\sum_{\ell=0}^{r-1} \frac{q^n}{q^n - q^\ell} < r \cdot \frac{q}{q-1}$$

[1] Rutgers, ECE 548, Spring 2019

[2] How many draws does it take to get n linearly independent equations?

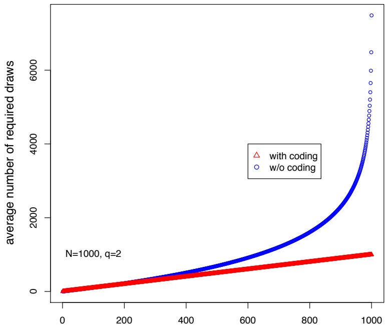
Time to Collect Coupons vs. Numbers

The average number of draws necessary for collection size r is

- without coding

$$\sum_{\ell=0}^{r-1} \frac{n}{n-\ell} = n(H_n - H_{n-r})$$

- with coding (see homework)



To collect n numbers, we will have to, on average, draw about

- $n \log n$ times if we use the random collection strategy,
- n times if we use coding.

Coding within Generations

$x_1, \dots, x_{n=mh}$ are partitioned into m sets, generations, of size h:

$$\boxed{x_1, \dots, x_h} \; \boxed{x_{h+1}, \dots, x_{2h}} \cdots \boxed{x_{(m-1)h+1}, \dots, x_{mh}}$$

In each draw we follow a two stage collection process:

1. a generation chosen uniformly at random (with replacement)
2. numbers in the chosen generation are linearly combined

Implications:

1. we collect generations as coupons, and code within generations $\Rightarrow$
2. we need to draw each generation at least h times to decode all data

Generations – Throughput/Complexity Tradeoff

Tradeoff: the collector will

- have to solve $h \times h$ systems of equations rather than $n \times n$
- collect data at a slower pace

If the generation size is h, we would like to know

- How many draws will it take to acquire the data?
- What are other advantages/disadvantages of having generations?

The Double Dixie Cup Problem

(aka Collector's Brotherhood Problem) is concerned with T_m^h: number of draws needed to acquire h complete sets of all m coupons.

$$E\left[T_m^h\right] = n\int_0^\infty \left[1-(1-S_h(x)e^{-x})^m\right] dx$$

$$S_h(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^{h-1}}{(h-1)!} \quad (h \geqslant 1)$$

Asymptotically,[3]

$$E\left[T_m^h\right] = m\log m + (h-1)m\log\log m + C_h m + o(m) \text{ for large } m$$

$$(C_h = \gamma - \log(h-1)!,\ \gamma \text{ is Euler's constant}) E\left[T_m^h\right]$$

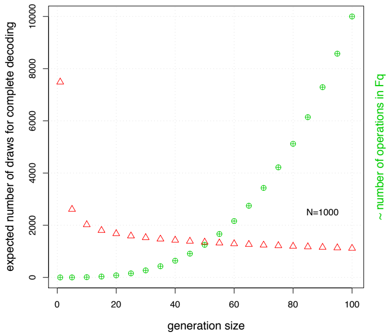
$$\to hm \text{ for large } h$$

(studied by Newman & Shepp '60, Erdös & Rényi '61, Flatto '82)

Generations – Throughput/Complexity Tradeoff

Data $x_1, \ldots, x_{n=mh}$ are partitioned into m generations of size h. With generation size h, we need

- $\mathcal{O}(h^3)$ operations in $\mathbb{F}_q$ for coding,



and for small h and large n, we need

- $\frac{n}{h}\left\{\mathcal{O}\left(\log\frac{n}{h}\right) + (h-1)\log\left[\log\left(\frac{n}{h}(1+o(1))\right)\right]\right\}$ draws to collect all data.

Generations - Computation/Tracking Cost Tradeoff

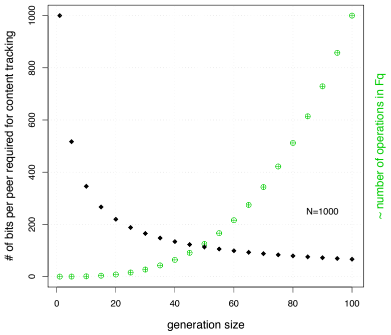
In P2P networks, to track a peer collection, without coding, we store a string of n bits; the i-th bit is 1 if the peer has data piece i. To track a peer collection, with coding over all data, we store the number of independent combinations in $\log_2(n+1)$ bits.

With generation size h, we need

- $\mathcal{O}(h^3)$ operations in $\mathbb{F}_q$ for coding,

and

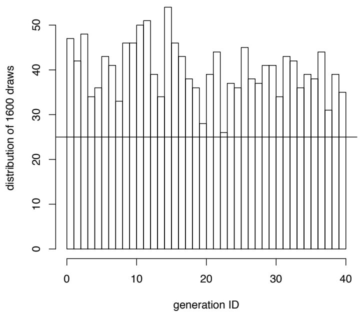
- $\log_2(h+1)$ bits to track the collection within each generation
- $(n/h)\log_2(h+1)$ bits in total.



[3] We have shown this for $h = 2$

Overhead per Generation

- Set of 1000 packets is partitioned into 40 generations of size 25.
- How do n random draws get distributed over the generations?
- Can the "lucky" generations help the "unlucky"?



Overlapping Generations

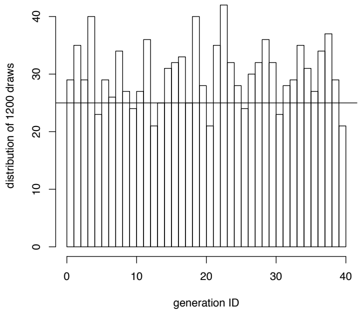
Overlapping generations of $\mathcal{F} = \{x_1, \dots, x_{mh}\}$ are formed in two steps:

1. $\mathcal{F}$ is partitioned into base generations B_i, $i = 1, \dots, m$, of size h.
2. ℓ randomly selected elements of $\mathcal{F} \setminus B_i$ are added to B_i, giving G_i.

At the moment the first generation, say G_j, can be decoded,

- it must have collected $h + \ell$ linearly independent coded packets,
- the other generations' still require innovative packets, BUT, their requirements get reduced because of their overlaps with G_i

How does random coding over generations compare to LT codes?



Homework - due March 12 - strict deadline

- You need to collect k numbers $x_1, \dots, x_k$ in finite field $\mathbb{F}_q$ by drawing random linear combinations of the numbers (random coding):

  $$\rho = \alpha_1 x_1 + \cdots + \alpha_k x_k, \quad \alpha_i \in \mathbb{F}_q$$

  (the coefficients α_i and the result of combining will be revealed). How many draws does it take on average to get n linearly independent equations?

- Consider a set of m vectors in $\mathbb{F}_q^n$ whose entries are chosen uniformly at random from $\mathbb{F}_q$, where $n < m$. Find the probability of having n linearly independent vectors in the set. Hint: Arrange the vectors in a matrix and consider its rank.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #12, March 7

This lecture introduces LDPC codes.

Binary Low Density Parity Check (LDPC) Codes

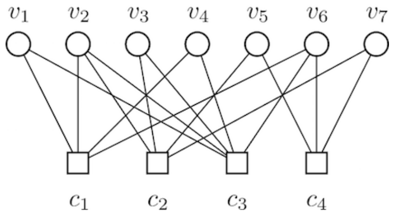
Definition

Generally speaking, a linear code defined by its parity check matrix H is an LDPC[2] code if the percentage of 1's in H is low, i.e., H is sparse (low-density).

We can represent the parity check matrix H of a linear code by a graph $\mathcal{H}$ in the following way. $\mathcal{H}$ has two disjoint sets of vertices referred to as *variable* nodes and *check* nodes. There are n variable nodes corresponding to the columns in H and r check nodes corresponding to the rows in H. If the element at position (i, j) of H is one, then there is an edge in $\mathcal{H}$ between the i-th check and the j-th variable node. Note that the only edges in $\mathcal{H}$ are hose that connect variable and check nodes, and thus $\mathcal{H}$ is a bipartite graph. Note that if H is sparse, then $\mathcal{H}$ is sparse. $\mathcal{H}$ is often referred as the Tanner graph of the code.

Note that we can define the code by $\mathcal{H}$ alone as the set of all length-n binary sequences on the variable nodes such that for each check node, the sum of the settings of the adjacent variable nodes is zero. If in the Tanner graph, every variable node has degree ℓ and every check node has degree m, we say that the codes is (ℓ, m)-biregular.

$$\mathbf{H} = \begin{array}{cc} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} & \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} & \begin{array}{c} c_1 \\ c_2 \\ c_3 \\ c_4 \end{array} \end{array}$$



For irregular codes, some fraction λ_d of variable node has degree d and some fraction ρ_d of check nodes has degree d. These codes are characterized by their variable and check degree distribution generating polynomials $\lambda(x)$ and $\rho(x)$:

$$\lambda(x) = \sum_d \lambda_d x^{d-1} \qquad \rho(x) = \sum_d \rho_d x^{d-1}.$$

[1] Rutgers, ECE 548, Spring 2019

[2] LDPC codes were invented by Robert Gallager in his PhD thesis in 1960. Soon after that, they were forgotten, and then reinvented several times for the next 30 years.

Decoding of Binary LDPC Codes on the on the Erasure Channel

Assume that all-zero codeword was sent, and some bits got erased.

Q: When do we know the value of the variable node?

A: When

1. it is not erased

   OR

2. it is connected to a check node whose all other variable nodes neighbors are not erased.

A Message Passing Algorithm

Round v: Each variable node passes a message to all its adjacent check nodes, ∞ if the corresponding bit is not erased and 0 if it is.[3]

Round cv: Check node c passes a message to an erased adjacent variable node v, ∞ if all the variable nodes incident to c except v are not erased. (Note that the message reflects how certain c is about v.)

Round vc: Variable node v passes a message to an adjacent check node c, ∞ if in the previous round v received ∞ from at least one of its adjacent check nodes other than c.

We iterate between these two rounds. The hope is that the number of 0 (i.e., uncertainty) messages goes down in each iteration

Another Algorithm

It is helpful to recall the algorithm we used for LT codes. However, there, we had data-bit and code-bit nodes, whereas here, we have variable (code-bit) and check nodes (equations). Note that the algorithm we described in the previous section is equivalent to the following: |

1. Initialize the values of all the check nodes to zero. ← initialization

2. For all variable nodes v, if the value is received (its bit not erased), then add its value to the values of all adjacent check nodes and remove v together with all edges emanating from it from the graph. ← from received variables to checks

3. If there is a check node c of degree one, substitute its value into the value of its unique neighbor among the variable nodes, add that value into the values of all adjacent check nodes and remove the variable nodes and all edges emanating from it from the graph. ← from checks to variables and back

[3] Note that the variable node's message reflects its certainty about its bit value.

Observations:

1. Decoding complexity is proportional to the number of edges in the graph.
2. There is no guarantee that the algorithm can decode all variable nodes.

Performance Analysis

We will use the following notation:

p_i the probability that the messages passed from variable nodes to check nodes at round i of the algorithm is 0.

q_i the probability that the message passed from check nodes to variable nodes at round i of the algorithm is 0.

Note that for successful decoding, we need p_i to decrease with i. Observe the following:

- At round $i+1$, a variable node v will send message 0 to check node c iff 1) v was erased and 2) all the messages that v received at round i from the neighboring check nodes other than c were 0. Therefore, for a variable node of degree d_ℓ, we have
$$p_{i+1} = \epsilon q_i^{d_\ell - 1}.$$
- The check node c passes a message ∞ to the variable node v iff all the neighboring variable nodes except for v send a message ∞ to c in the previous round. Therefore, for a check node of degree d_r, we have
$$q_i = 1 - (1 - p_i)^{d_r - 1}$$

Note that these recursions are conditioned on the variable and check node degrees. Recall the degree distribution generating polynomials $\lambda(x)$ and $\rho(x)$:

$$\lambda(x) = \sum_{d_\ell} \lambda_{d_\ell} x^{d_\ell - 1} \qquad \rho(x) = \sum_{d} \rho_d x^{d-1}.$$

By the formula for the total probability, we get

$$\begin{aligned} p_{i+1} &= \sum_{d_\ell} \lambda_{d_\ell} \epsilon \Big(\sum_{d_r} \rho_{d_r} \big(1 - (1 - p_i)^{d_r - 1}\big)\Big)^{d_\ell - 1} \\ &= \epsilon \sum_{d_\ell} \lambda_{d_\ell} \Big(1 - \sum_{d_r} \rho_{d_r} (1 - p_i)^{d_r - 1}\Big)^{d_\ell - 1} \\ &= \epsilon \lambda(1 - \rho(1 - p_i)) \end{aligned}$$

Therefore, for successful decoding, we need

$$\epsilon\lambda\big(1-\rho(1-x)\big) \leqslant x \text{ for } 0 < x < \epsilon. \tag{1}$$

This condition can be used to calculate the maximal fraction of erasures a random LDPC code with given degree distributions can correct using the simple decoding algorithm.

Homework - due April 2

Consider a random Tanner graph in which each variable node has degree 3, and each check node has degree 6.

1. How are such graphs called? What are $\lambda(x)$ and $\rho(x)$?

2. Find the maximum fraction of erasures ϵ that the message passing decoding algorithm can recover

   (a) from the successful decoding condition (1).

   (b) by simulating the decoder on many large $(3;6)$-biregular random graphs.

3. Compare the performance of the two algorithms.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #13, March 12

This lecture talks about the Hadamard matrices and two important classes of codes based on these matrices.

Hadamard Matrices

Hadamard Matrices were introduced by Jacques Hadamard in 1893, and have received much attention in various fields. However, the question of existence has not been fully answered.

Definition and Properties

A Hadamard matrix is a square matrix whose entries are either 1 or -1 and whose rows are pairwise orthogonal over the field of real numbers.

$\Longrightarrow$

- Each row-vector has length (Euclidean norm) $\sqrt{n}$,
- $\det(H) = \pm n^{\frac{n}{2}}$
  (Proof: $HH^T = nI_n \implies \det(HH^T) = n^n \implies \det(H) = \pm n^{\frac{n}{2}}$.)

The order of a Hadamard matrix must be 2 or a multiple of 4, and the Hadamard conjecture proposes that a Hadamard matrix of order $4m$ exists for all positive integers m.

Let $\mathbf{A}$ be an $m \times n$ matrix and B a $p \times q$ matrix. Then the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ is the $mp \times nq$ matrix

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}.$$

Let $\mathbf{A}$ and C be $n \times n$ matrices and B and D $m \times m$ matrices. Then

$$(\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}.$$

Therefore, if H_n and H_m are Hadamard matrices of respective orders n and m, then $H_n \otimes H_m$ is a Hadamard matrix of order nm.

Sylvester Construction

Hadamard matrices of order 2^k for every positive integer k, can be constructed as follows:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix} = H_2 \otimes H_{2^{k-1}} \text{ for } k \geqslant 2.$$

[1] Rutgers, ECE 548, Spring 2019

$\Longrightarrow$ There exists a Hadamard matrix of order 2^k for all k.
These Hadamard matrices are sometimes called Sylvester matrices. Recall that the conjecture is that for all $k \geqslant 1$, there exists a Hadamard matrix of order $4k$. The smallest dimension open case of this conjecture is currently $4k = 668$.

The 0, 1 *Counterpart*

If we replace each 1 by 0, and each -1 by 1, we get the 0, 1 counterpart. These matrices are important in coding and design theory. The Sylvester H_8 matrix becomes the following matrix χ_8:

$$\chi_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Codes Related to Hadamard Matrices

We will use Hadamard matrices to define Simplex and Hadamard codes.[2] We will use Sylvester matrices to construct linear codes, but the procedure holds for the Hadamard matrices in general, just the codes will not be necessarily linear.

The Simplex Codes

If we delete the first column from χ_8, we get the following matrix:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

The rows of this matrix are codewords of the $[7, 3, 4]$ Simplex code, which is the dual of the $[7, 4, 3]$ Hamming code. In general, if we start with H_{2^k}, the rows of the corresponding χ_{2^k} with the first 0 removed form a $[2^k - 1, k, 2^{k-1}]$ simplex code. This code is the dual of the $[2^k -$

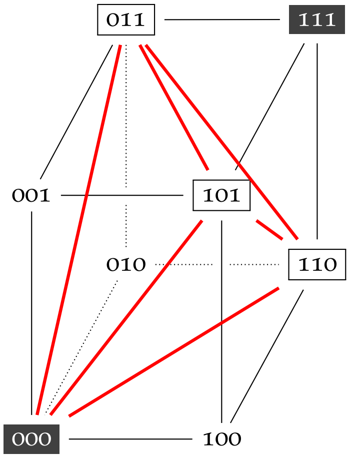
[2] Hadamard matrices are also related to Reed-Muller and polar codes.

$1, 2^k - 1 - k, 3]$ Hamming code. What are the rows of the generator matrix of the simplex code?

Note that there are 2^k codewords of the simplex code and they live in a $2^k - 1$ dimensional space, hence the name simplex.[3] A simplex (plural: simplexes or simplices) is the higher-dimensional generalization of the triangle. The triangle is a 3-dimensional polytope in the 2-dimensional space.

Example: Codewords of the $[3, 2]$ Simplex code are connected in a tetrahedron:

(Its dual code is the Hamming code $\{000, 111\}$)



The $[32, 6, 16]$ Hadamard code is the code used in the period 1969 – 1972 by the Mariner spacecraft to transmit images of Mars.

The Hadamard Codes

The rows of the matrix

$$\begin{bmatrix} \chi_{2^{k-1}} \\ -\chi_{2^{k-1}} \end{bmatrix}$$

form codewords of the Hadamard $[2^{k-1}, k, 2^{k-2}]$.

[3] An m-simplex is a m-dimensional polytope which is the convex hull of its $m + 1$ vertices.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #14, March 26

This lecture introduces 1) combinatorial designs and 2) codes for distributed storage.

Combinatorial Designs

A design is a pair $(\mathcal{P}, \mathcal{B})$ where $\mathcal{P}$ is a set of elements points and $\mathcal{B}$ is a set of non-empty subsets of $\mathcal{P}$ called blocks. The blocks are required to satisfy certain "balance" properties, and the combinatorial design theory asks when that is possible.[2] The theory originated with the design and analysis of statistical experiments. Contemporary applications are wide, and include, e.g., biology and networking.

Block Designs aka $t-$*Designs*

The most studied designs are the balanced (incomplete) block designs (BIBD), often called just block designs or $t-$designs:

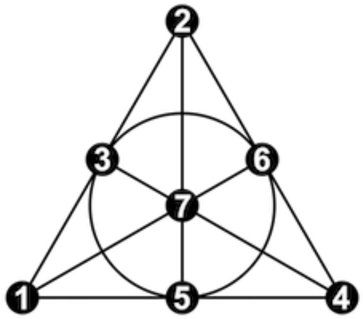
Definition: A $t-(v,k,\lambda)$ design is a pair $(\mathcal{P}, \mathcal{B})$ where $\mathcal{P}$ is a set of v elements, called points, and $\mathcal{B}$ is a collection of distinct subsets of t points of $\mathcal{P}$, called blocks,[3] such that every subset of points of size t is contained in exactly λ blocks.[4]

The number of blocks in $\mathcal{B}$ is determined by the parameters t, v, k, and λ. There are several special cases of $t-$designs, e.g.,

Example: A $2-(7,3,1)$ design:

$X = \{1,2,3,4,5,6,7\}$

$\mathcal{A} = \{123, 145, 167, 246, 257, 347, 356\}$



- If $\lambda = 1$, a $t-$design is called a Steiner $S(t,k,v)$ system or a Steiner $t-$design. When $t = 2$, we have a *Steiner triple system.* The $2-(7,3,1)$ design in the figure is also an $S(2,3,7)$ Steiner triple system. The blocks are the 7 lines, each containing 3 points. Every pair of points belongs to a unique line.
- If $b = v$, the t-design is symmetric and $k - \lambda$ is called its order.
- A symmetric $2-(v,k,1)$ design, i.e., a symmetric $S(2,k,v)$ design) turns out to be a projective plane of order $k-1$. The $2-(7,3,1)$ design in the figure is also known as the the Fano plane in finite geometry. It has 7 points and 7 lines, with 3 points on every line and 3 lines through every point. Note the symmetry. Where is the 7-th line? The Fano plane

Example: The Kirkman's Schoolgirl Problem:
Fifteen girls walk to school three side-by-side for seven days in succession: it is required to arrange them daily so that no two walk side-by-side more than a single day. ⟵ *another Steiner triple system.*

[1] Rutgers, ECE 548, Spring 2019

[2] As for codes, we are concerned with strings of letters and relationships between these strings.

[3] "incomplite" comes from $k < v$.

[4] The balance condition.

It is often convenient to describe a $t-$design by giving a matrix that indicates the points that are in each block.

Definition: An incidence matrix of a (v, k, λ) block design is a $v \times k$ zero/one matrix indexed by the points and the blocks which has a 1 at the entry i, j iff point i belongs to block j.

Projective planes and Geometries

Consider a vector space $\mathbb{F}_q^k$. The number of different basis of this space (that is, the number of ways we can chose k linearly independent vectors) is given by is

$$(q^k - 1)(q^k - q) \ldots (q^k - q^{k-1})$$

How many $k-$dimensional subspaces does the $n-$dimensional space $\mathbb{F}_q^n$ have? This number is given by the Gaussian binomial coefficient[5] defined as

$$\begin{bmatrix} n \\ k \end{bmatrix}_q = \frac{(q^n - 1)(q^n - q) \ldots (q^n - q^{k-1})}{(q^k - 1)(q^k - q) \ldots (q^k - q^{k-1})}$$

Consider the 3-dimensional vector space $\mathbb{F}_q^3$. There are $q^2 + q + 1$ different 1-dimensional subspaces of $\mathbb{F}_q^3$. They will correspond to our points and all 2-dimensional subspaces of $\mathbb{F}_q^3$ will be our blocks. there are $q + 1$ points in each block, and any two blocks intersect at exactly one point. Observe that this structure is a $2 - (q^2 + q + 1, q + 1, 1)$ design. For $q = 2$, we have the Fano plane.

We can obtain d-dimensional projective geometry $\mathbb{PG}_d(q)$ and the corresponding

$$2 - \left(\frac{q^{d+1} - 1}{q - 1}, \frac{q^d - 1}{q - 1}, \frac{q^{d-1} - 1}{q - 1}\right) - \text{design}$$

by generalizing this reasoning to $\mathbb{F}_q^{d+1}$ with its $1-$dimensional subspaces as points and $(d-1)-$dimensional subspaces as blocks.

Coding in Distributed Storage

In distributed storage, disks fail and data changes, but storage reliability must be maintained.

If a disk (node) fails, we want to

1. still be able to recover data from the remaining storage (reliability)[6]
2. reproduce the lost data (or reliability) on each replacement disk with, e.g.,

Example: Consider storing 2 equal-size content pieces a and b on 4 identical disks:
System 1: stores a, a, b, b and
System 2: stores a, b, $a + b$, $a - b$

[5] q analog of the binomial coefficient

[6] Which system is more reliable?

- minimal data download from the remaining storage (repair bandwidth)[7]

  or
- by downloading (coded) data from only a few other nodes (locality).[8]

If the stored data changes, we must accordingly update the storage.[9]

Codes with Locality and Availability

Locality

Suppose a single position in a codeword is erased. How many other symbols do we need to read to recover the erased symbol?
If the code is $[n, k]$ MDS, we need k symbols.
Locally repairable codes (LRC) allow repair of an erasure using fewer than k other symbols. A set of symbols that can repair a given symbol is referred to as the *repair group* of that symbol.

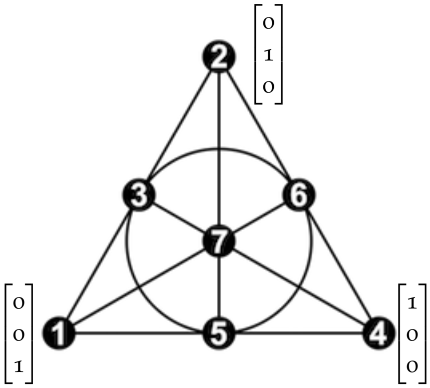
We say that a code symbol has locality r if it can be recovered by accessing at most r other code symbols (that is, the largest repair group has r symbols.) If all code symbols have locality r, we say that the code is r-LRC.

Availability

We say that a code provides availability, when one erased symbol can be recovered form multiple disjoint sets of other code symbols, that is there are multiple repair groups for each symbol.
We say that a code has (r, t)-availability if each symbol has t disjoint repair groups each of size at most r

Example: The $[7, 3]$ Simplex code is a $(2, 3)$-availability code.

$$G = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$



$\begin{bmatrix}0\\1\\0\end{bmatrix}$
2
3
6
7
$\begin{bmatrix}0\\0\\1\end{bmatrix}$
1
5
4
$\begin{bmatrix}1\\0\\0\end{bmatrix}$

repair groups for the first position:

$\{23, 45, 67\}$
(any two columns in G that XOR to 1)

Implications on G *and Code Parameters*

The minimum distance penalty for an (n, k, r, t)-LRC in which any repair group contains only 1 parity symbol

$$d_{min} \leqslant n - k - \left\lceil \frac{kt}{r} \right\rceil + t + 1$$

Note that when $r = k$, we get the Singleton bound.

For an (n, k, r, t)-LRC (linear or non-linear), we have

$$d_{min} \leqslant n - k - \left\lceil \frac{t(k-1)+1}{t(r-1)+1} \right\rceil + 2$$

The locality and availability dictate the (linear) dependence within the columns of the generator matrix. The i-th symbol has locality r

[7] Which system needs less repair bandwidth?

[8] Which system has better locality?

[9] Which system is more update efficient?

if there are r other columns of G such that the i-th column is in their span. If there are t disjoint sets of such r columns, then the i-th symbol has availability t.

Recall that for $[2^m - 1, m, 2^{m-1}]$ binary simplex code, the columns of the generator matrix G are all distinct nonzero vectors of $\mathbb{F}_2^m$. Therefore, any two columns of G add up to another column of G, and thus the $[2^m - 1, m, 2^{m-1}]$ binary simplex code has locality $r = 2$ and availability $t = (2^m - 1 - 1)/2 = 2^{m-1} - 1$.

Homework – due on April 4

1. Construct the incidence matrix of the points and lines in the Fano plane, and compare with χ_8 defined in Lecture 13.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #15, March 28

This lecture is about Reed-Muller codes and majority logic decoding.

Reed-Muller Codes

Definition(s)

$\mathrm{RM}_q[r;m]$ Reed Muller[2] code with integer parameters m and r, $0 \leqslant r \leqslant m$, is the linear code over $\mathbb{F}_q$ obtained when all polynomials over $\mathbb{F}_q$ in m variables and the total degree at most r are evaluated on the elements of $\mathbb{F}_q$. Data symbols are interpreted as the coefficients of the monomials in the polynomial (as for RS codes). Cdeword symbols are obtained by evaluating the polynomial on a different combination of m variables in $\mathbb{F}_q$.

We will denote the m variables by $v_1, v_2, \ldots, v_m$. For $\mathrm{RM}_q[r;m]$, this set of variables take values in $\mathbb{F}_q^m$, and the monomials we use to build our evaluation polynomial are given by

$$v_1^{d_1} v_2^{d_2}, \ldots, v_m^{d_m}, \quad d_i \geqslant 0 \text{ and } \sum_{i=1}^{m} d_i \leqslant r.$$

The polynomial is a linear combination of these monomials,[3] and data symbols are coefficients uses for in these linear combinations.

We will mostly consider binary RM codes.[4]

Example: Show that $\mathrm{RM}_2[m;m]$ is the $[2^m, 2^m, 1]$ universe code.

Example: Show that $\mathrm{RM}_q[0;m]$ is the $[q^m, 1, q^m]$ repetition code.

Example: For the $\mathrm{RM}_q[1;m]$ code, we are linearly combining the following monomials:

$$1, v_1, v_2, \ldots, v_m$$

into the polynomial

$$a_0 \cdot 1 + a_1 \cdot v_1 + a_2 \cdot v_2 + \cdots + a_m \cdot v_m$$

Example: What can we say about $\mathrm{RM}_2[1;3]$ code?[5]
The $\mathrm{RM}_2[1;3]$ code is generated by the set $\{1, v_1, v_2, v_3\}$, and for each combination of values of (v_1, v_2, v_3), there is a code symbol. Since $(v_1, v_2, v_3) \in \mathbb{F}_2^3 = \{(0,0,0), (0,0,1), \ldots, (1,1,1)\}$, the generator matrix

[1] Rutgers, ECE 548, Spring 2019

[2] RM codes are introduced in 1954, first by Muller and shortly after by Reed, who also provided a decoding algorithm.

[3] How many such monomials can we have?

[4] $\mathrm{RM}_2[1;5]$ was used by Mariner 9 to send pictures from Mars in 1971, and also in 3G cell phones for protecting some important meta data.

[5] $\mathrm{RM}_2[1;5]$ and $\mathrm{RM}_2[2;5]$ are featured in the famous Apple-Samsug law suits.

of $RM_2[1;3]$ is

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Another Way to Construct an $RM_2[r;m]$ Code

RM codes can be constructed recursively. $RM_2[r;m]$ is the following concatenation of $RM_2[r;m-1]$ and $RM_2[r-1;m-1]$:

$$RM_2[r;m] = \{(u, u+w) \mid u \in RM_2[r;m-1], w \in RM_2[r-1;m-1]\}.$$

To see that this is true, note that a polynomial $a(v_1, \ldots, v_{m-1}, v_m)$ in m variables can be decomposed as

$$\begin{aligned} a(v_1, \ldots, v_{m-1}, v_m) &= f(v_1, \ldots, v_{m-1}) + v_m g(v_1, \ldots, v_{m-1}) \\ &= \begin{cases} f(v_1, \ldots, v_{m-1}), & \text{when } v_{m+1} = 0, \\ f(v_1, \ldots, v_{m-1}) + g(v_1, \ldots, v_{m-1}), & \text{when } v_{m+1} = 1. \end{cases} \end{aligned}$$

We get the first 2^{m-1} code symbols of $RM_2[r;m]$ by evaluating the polynomial $a(v_1, \ldots, v_{m-1}, v_m)$ on $\mathbb{F}_2^{m-1} \times 0$ and the other 2^{m-1} code symbols by evaluating $a(v_1, \ldots, v_{m-1}, v_m)$ on $\mathbb{F}_2^{m-1} \times 1$.

Code Parameters

Data Ward Length

What is the message length k, i.e., how many different monomials in m variables and total degree at most r are there in $\mathbb{F}_q$? When $q = 2$, we can easily compute this number this number. Since $v \cdot v = x$, each variable in the monomial has power either 0 or 1. Since, no more than r variables can have degree 1, the message length is given by

$$k = \sum_{d=0}^{r} \binom{m}{d}.$$

Code Length

The length of $RM_q[r;m]$ is q^m. Each codeword symbol is obtained by evaluating the polynomial on a different combination of m variables in $\mathbb{F}_q$.

The Minimum Distance

The minimum distance of $RM_2[r;m]$ codes is 2^{m-r}.

Proof (by induction): Recall that $RM_2[0;m]$ is a length 2^m repetition code and thus the minimum distance $2^m = 2^{m-0}$, and the claim holds for all m. The $\mathrm{RM}_2[m; m]$ is the universe code, hence $d_{\min} = 1 = 2^{m-m}$.

As an inductive hypothesis, assume that the claim holds up to $m - 1$ for all r s.t. $0 \leqslant r \leqslant m - 1$. Recall that

$$\mathrm{RM}_2[r; m] = \{(u, u + w) \mid u \in \mathrm{RM}_2[r; m - 1], w \in \mathrm{RM}_2[r - 1; m - 1]\}.$$

Let $u, u' \in \mathrm{RM}_2[r; m - 1]$ and $w, w' \in \mathrm{RM}_2[r - 1; m - 1]$. We consider 2 cases:

1. If $w = w'$ then

$$\begin{aligned} d(c_1, c_2) =& d\big((u, u \oplus w), (u', u' \oplus w')\big) \\ =& 2d(u, u') \geqslant 22^{m-1-r} \end{aligned}$$

   with equality when $d(u, u') = 2^{m-1-r}$.

2. If $w \neq w'$ then

$$\begin{aligned} d(c_1, c_2) =& d(u, u') + d(u \oplus w, u' \oplus w') \\ =& d(u \oplus u', 0) + d(w \oplus w' \oplus u \oplus u', 0) \\ \geqslant& d(u \oplus u', 0) + d(w \oplus w', 0) - d(u \oplus u', 0) \\ =& d(w, w') \geqslant 2^{(m-1-(r-1))} = 2^{m-r} \end{aligned}$$

   We used $d(x + y, 0) \geqslant d(x, 0) - d(y, 0)$, which follows from the triangle inequality of the Hamming distance and $\mathbb{F}_2$.

Majority Logic Decoding

Example: The $\mathrm{RM}_2[2; 4]$ code is a $[16, 11, 4]$ code. The monomials up to degree 2 are

$$(1, v_1, v_2, v_3, v_4, v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4, v_3v_4)$$

which give raise to the following generator matrix:

1	1111111111111111
v_4	0000000011111111
v_3	0000111100001111
v_2	0011001100110011
v_1	0101010101010101
v_3v_4	0000000000001111
v_2v_4	0000000000110011
v_1v_4	0000000001010101
v_2v_3	0000001100000011
v_1v_3	0000010100000101
v_1v_2	0001000100010001

We denote the 11 data bits by

$$(a_0, a_4, a_3, a_2, a_1, a_{34}, a_{24}, a_{14}, a_{23}, a_{13}, a_{12})$$

and the codeword $c_0, c_1, \ldots, c_{15}$.

Note that the following identities hold:

$$\begin{aligned} a_{12} &= c_0 + c_1 + c_2 + c_3 \\ a_{12} &= c_4 + c_5 + c_6 + c_7 \\ a_{12} &= c_8 + c_9 + c_{10} + c_{11} \\ a_{12} &= c_{12} + c_{13} + c_{14} + c_{15} \end{aligned}$$

Therefore if only one error occurred (one of the codeword bits flipped), 3 out of 4 of the sums above, i.e., the majority, will have the correct value for a_{12}.[6]

[6] Observe the locality and availability properties!

Error Control Coding [1]

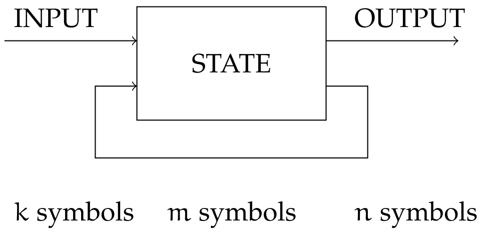
Prof. Emina Soljanin

Lecture #16, April 2

This lecture introduces convolutional codes.

Finite State Codes

Finite state codes have memory (state), and the encoder output depends on the input and the state. The state depends on the input and the previous state. When these dependences are linear, we say that the code is convolutional.



Convolutional Codes

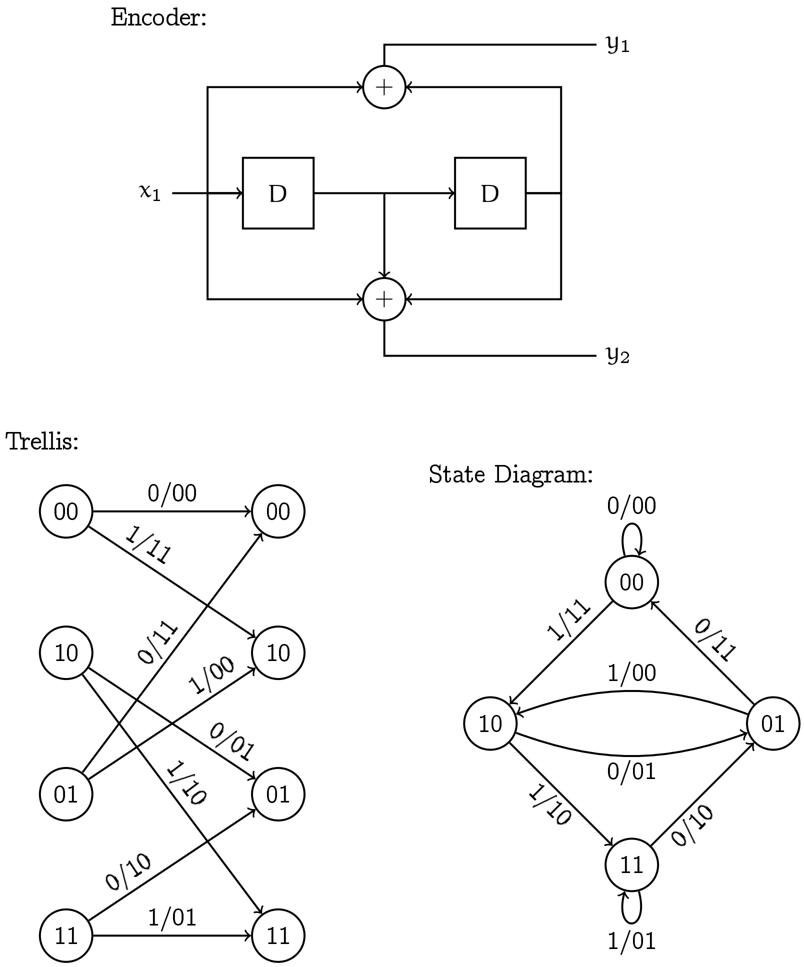
A convolutional encoder is realized as a linear sequential circuit with m memory elements, k symbols and n output symbols. Therefore, the output depends on the current k input symbols (current input block) and the m preceding input blocks. The encoder is a set of n digital filters (linear time-invariant systems). We get the code sequence by interleaving the outputs of these filters. Typically k and n are small integers. Large minimum distances are achieved by increasing the memory order m rather than k and n. The rate of the code is k/n.

Convolutional codes were introduced by Elias in 1955. Various decoding schemes followed: sequential decoding in 1961 by Wozencraft, majority logic decoding in 1963 by Messey Massey, ML decoding in 1967 by Viterbi, MAP decoding in 1974 by Bahl, Cocke, Jelinek and Raviv (BCJR). One of the most important class of codes, turbo codes, are based on convolutional codes and the BCJR algorithm. Turbo codes were invented in 1993 by Berrou, Glavieux, and Thitimajshima.

[1] Rutgers, ECE 548, Spring 2019

Convolutional Code Representations

There are several ways to define a convolutional code. The figure shows 3 such possibilities for a rate 1/2 convolutional code.



Encoding

Let x_t denote the input to the encoder at time t. Then, for the encoder in the above figure, the state at time t is given by $\boxed{x_{t-1}, x_t}$, and the output of the encoder is given by

$$y_t^1 = x_t + x_{t-2} \qquad y_t^2 = x_t + x_{t-1} + x_{t-2}$$

To a rate k/n convolutional code, we associate a $k \times n$ transfer function matrix $G(D)$. For our rate 1/2 example code, we have

$$G(D) = \begin{bmatrix} 1 + D^2 & 1 + D + D^2 \end{bmatrix}$$

The infinite generator matrix of the code starts as follows:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & & & & \\ & & 1 & 1 & 0 & 1 & 1 & 1 & & \\ & & & & 1 & 1 & 0 & 1 & 1 & 1 \\ & & & & & & 1 & 1 & 0 & 1 \end{bmatrix}$$

Systematic Encoder

Transfer function matrix $G(D)$ can be turned into a systematic form by identifying a $k \times k$ submatrix $T(D)$ and multiplying $G(D)$ by $T^{-1}(D)$ to get the corresponding systematic matrix $G^s(D)$:

$$G^s(D) = T^{-1}(D)G(D)$$

Our rate 1/2 code has the following systematic transfer function matrix:

$$G(D) = \begin{bmatrix} 1 & \frac{1+D+D^2}{1+D^2} \end{bmatrix}$$

Note that if the elements of $G(D)$ are FIR filters (polynomials in D), then the elements of $G^s(D)$ are in general rational functions in D, that is, IIR filters.

Example: Suppose

$$G(D) = \begin{bmatrix} 1+D & D & 1 \\ D^2 & 1 & 1+D+D^2 \end{bmatrix}.$$

We can take the first two columns of $G(D)$ as $T(D)$, and get

$$T(D) = \begin{bmatrix} 1+D & D \\ D^2 & 1 \end{bmatrix} \qquad T^{-1}(D) = \frac{1}{1+D+D^3} \begin{bmatrix} 1 & D \\ D^2 & 1+D \end{bmatrix}.$$

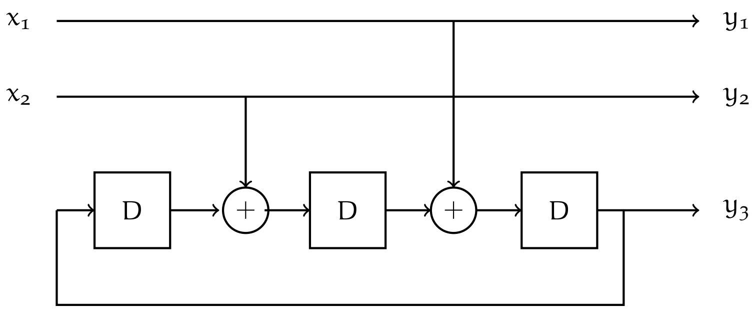
Then

$$G^s(D) = T^{-1}(D)G(D) = \begin{bmatrix} 1 & 0 & \frac{1+D+D^2+D^3}{1+D+D^3} \\ 0 & 1 & \frac{1+D^2+D^3}{1+D+D^3} \end{bmatrix}$$

Recursive Encoder

Example: Consider the rate 2/3 code with the systematic matrix transfer function:

$$G(D) = \begin{bmatrix} 1 & 0 & \frac{D}{1+D^3} \\ 0 & 1 & \frac{D^2}{1+D^3} \end{bmatrix}$$



Catastrophic Encoder

When errors occur in a codeword, decoding may result in even more errors. Consider, for example, the $[3,1,3]_2$ repetition where 000 received as 110 will be decoded as 111. In block codes errors cannot propagate very far because of finite size blocks. This is not necessarily the case for convolutional codes as it is possible for a finite error in the code sequence to have an infinite error in the corresponding input sequence.

We say that a convolutional mapping is catastrophic if there is some code sequence $y(D)$ with finitely many 1s that results from an input sequence $x(D)$ with infinitely many 1s. Every code has catastrophic and non-catastrophic encoders. Every systematic generator matrix of a convolutional code, is non-catastrophic.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #17, April 4

This lecture is about Reed-Muller codes and majority logic decoding.

Viterbi Decoding of Convolutional Codes

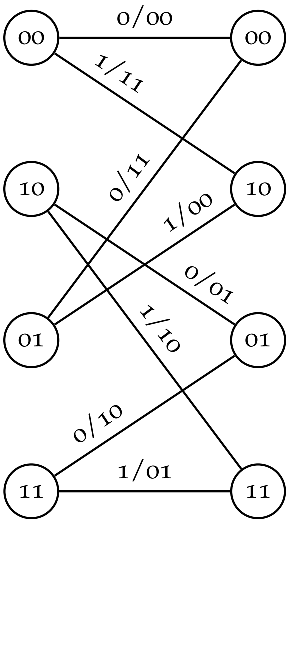
Distance Measures

The ℓ-th order column distance of a rate k/n encoder is the minimum Hamming weight of the code sequences of length $\ell + 1$ n-tuples (branches in the trellis), which result from an information sequence with non-zero first k-tuple.

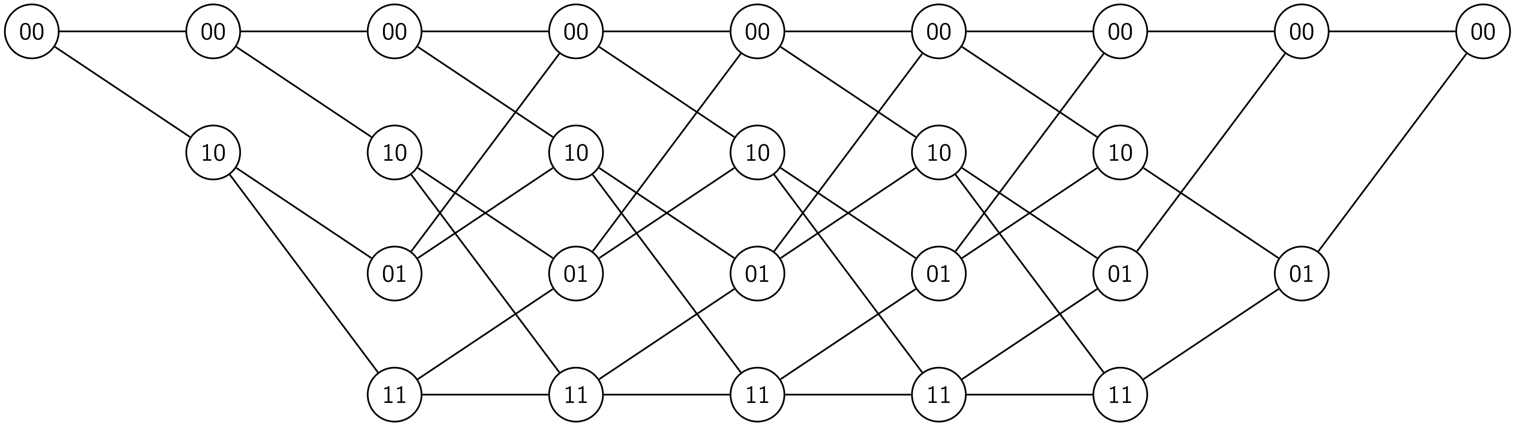
The free distance d_{free} of a convolutional code is defined as the minimum Hamming weight of any non-zero codeword of that code. Recall that we assume that m termination blocks are appended to the input data, and thus the codewords start from and end in the all-zero state. Our example code has a minimum distance 5.

We can find d_{free} by using the trellis diagram of the code. We start from the all zero state. We consider paths that diverge from the zero state and merge back to the zero state, regardless of the length but without intermediate passes through the zero state. The procedure is similar to decoding of convolutional codes by the Viterbi algorithm.

Decoding on the Trellis



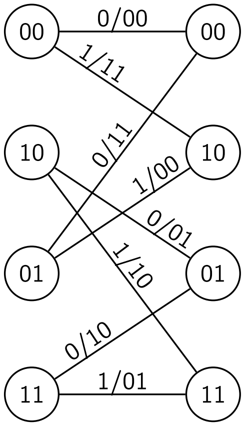
[1] Rutgers, ECE 548, Spring 2019



11 10 10 00 10 01 10 11 SENT

RECEIVED

DECODED



Error Control Coding [1]

Prof. Emina Soljanin

Lecture #18, April 9

This lecture covers syndrome decoding of Linear Codes.

Syndrome decoding finds the codeward which is the closest to the received word, i.e., it is the minimum distance decoding.

The $[7, 4]$ *Hamming Code Example*

$$\mathbb{C} = \{0000000, 0001111, 0010110, 0011001, 0100101, 0101010, 0110011, 0111100, \\ 1000011, 1001100, 1010101, 1011010, 1100110, 1101001, 1110000, 1111111\}.$$

For all $\mathbf{c} \in \mathbb{C}$, we have

$$\mathbf{c} \cdot H^T = 0 \text{ where } H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Linear code is the null space of the parity check matrix.

Error Model

The bit flip error model corresponds to binary addition of vectors having 1s at places where errors occurred and 0s otherwise, that is, we receive

$$r = c + e$$

where, e.g.,

$\boldsymbol{e} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ means an error happened at position 1.

$\boldsymbol{e} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ means errors happened at positions 6 and 7.

$\Rightarrow$ $\mathbf{r}$ can be any word, i.e., any vector in $\mathbb{F}_2^7$.

Note that

$$\mathbf{r} \cdot H^T = (\mathbf{c} + \boldsymbol{e}) \cdot H^T = \underbrace{\mathbf{c} \cdot H^T}_{=0} + \boldsymbol{e} \cdot H^T = \boldsymbol{e} \cdot H^T \ \leftarrow \text{ syndrome } s.$$

$\mathbf{s} = \mathbf{0} \ \Rightarrow$ no error (true for all codes)
In this example, when s matches the i-th column of H, the i-the bit of c was flipped. $\Rightarrow$ we can get c by flipping back the i-th bit of r.

How helpful is the syndrome in general? In general, syndrome decoding is not this simple. For short codes, it is still very efficient, essentially a lookup table. Lookup tables are traditionally avoided for longer codes over large fields, and for some classes of linear codes, more efficient algorithms exist.

[1] Rutgers, ECE 548, Spring 2019

Cosets and the Standard Array

Let G be a group, Γ a subgroup of G, and x an element of G. Then

$$x + \Gamma = \{xy : y \in \Gamma\} \quad \text{and} \quad \Gamma + x = \{yx : y \in \Gamma\}$$

are the left and the right cosset of Γ in G with respect to x. Observe the following:

- When G is Abelian, we have $x + \Gamma = \Gamma + x$.
- When $x \in \Gamma$, we have $x + \Gamma = \Gamma + x = \Gamma$

For an $[n, k, d]$ linear code C, the received words will be in $\mathbb{F}_2^n$, We are interested in all cossets of C in $\mathbb{F}_2^n$. A standard array of C when $d \geqslant 3$ is a table of 2^{n-k} rows and 2^k columns, where we arrange all elements $\mathbb{F}_2^n$ as follows:

- In the top row (row 0), we list the codewords, starting with the all-zero codeword.
- For the next row, we choose some[2] minimum-weight word not yet in the array, say γ, and list $\gamma + C$ starting with γ.
- We repeat the last step until all element of $\mathbb{F}_2^n$ appear in the table.

We call the first element of each row is the coset leader. If codeword c appears in the first row and j-th column, then the row with cosset leader γ has $\gamma + c$ in the j-the column. Each received word is corrected to the codeword on top of its column.

Decoding of the $[3, 1, 3]$ Hamming and the $[3, 2, 2]$ Simplex Codes

The $[3, 1, 3]$ Hamming Code

Recall that

$$G = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad H = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

We have the following standard array:

000	111	← code (syndrome 00)
001	110	← coset with syndrome 11
010	101	← coset with syndrome 10
100	011	← coset with syndrome 01

[2] The resulting array depends on this choice.

The $[3, 2, 2]$ *Simplex Code*

Recall that

$$G = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

We have the following standard array:

000	011	101	110	← code (syndrome 0)
001	010	100	111	← coset with syndrome 1

Recall that the $[2^k - 1, k, 2^{k-1}]$ simplex code is the dual of the $[2^k - 1, 2^k - 1 - k, 3]$ Hamming code. Note that there are 2^k codewords of the simplex code and they live in a $2^k - 1$ dimensional space, hence the name simplex. A simplex (plural: simplexes or simplices) is the higher-dimensional generalization of the triangle. The triangle is a 3-dimensional polytope in the 2-dimensional space, and An m-simplex is a m-dimensional polytope which is the convex hull of its $m + 1$ vertices.

Homework - due April 16

Compute a standard array for the $[7, 3, 4]$ Simplex code.

Error Control Coding [1]

Prof. Emina Soljanin

Lecture #19, April 11

This lecture covers Syndrome Decoding of Linear Codes.

Decoding of the $(3,6)$ LDPC Code on the BEC

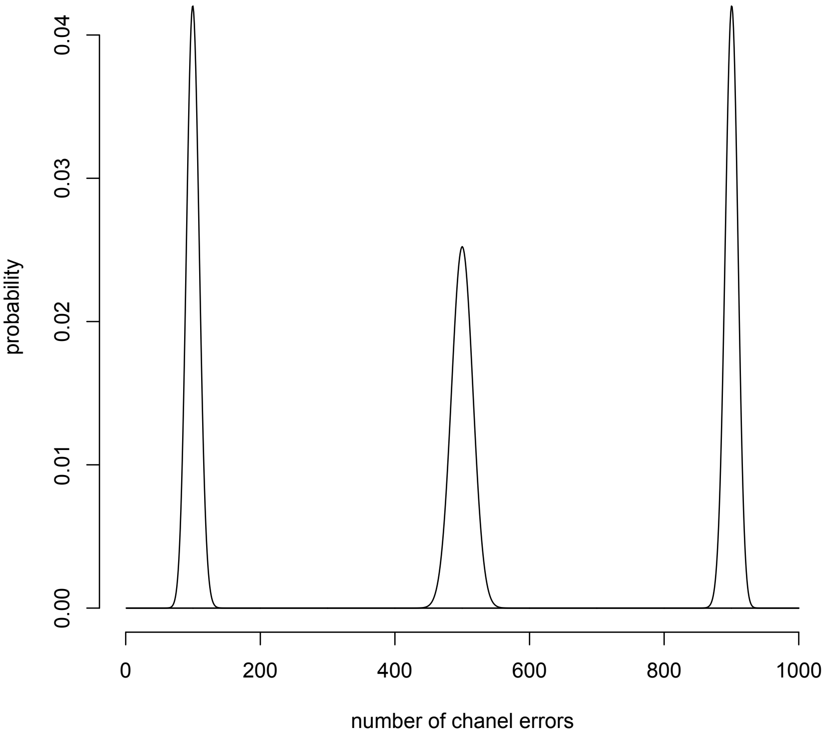
The BEC(ϵ)

What is the fraction of errors over n channel uses? The number of errors is the Binomial random variable $X \sim B(n,p)$. The probability of getting exactly m errors is given by

$$\Pr(X = m) = \binom{n}{m} \epsilon^m (1-\epsilon)^{n-m}, \qquad m = 0, 1, 2, \ldots, n.$$

Note that any number of errors from 0 to n can happen, but as n increases, it tends to concentrate around the mean $n\epsilon$. Recall that the variance is $n\epsilon(1-\epsilon)$.

Example: $n = 1000$, $\epsilon = 0.1$, 0.5, 0.9.



[1] Rutgers, ECE 548, Spring 2019

Message Passing Decoding

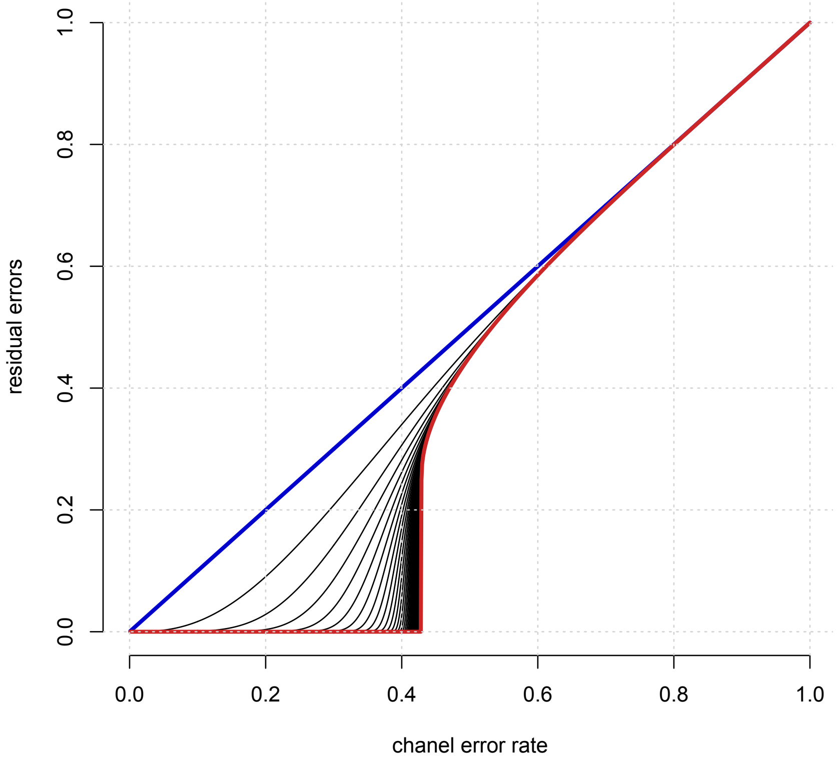
In message passing decoding of an LDPC code on the BEC, we derived the evolution of the probability that the messages passed from variable to check nodes is 0 "erasure") from round i to round $i+1$ to be

$$p_{i+1} = \epsilon\lambda(1-\rho(1-p_i)) \tag{1}$$

for a code with the left degree distribution generating polynomial $\lambda(x)$ and the right degree distribution generating polynomial $\rho(x)$, and the channel erasure rate ϵ. For the $(3,6)$ code, we have

$$\lambda(x) = \sum_{d_\ell} \lambda_{d_\ell} x^{d_\ell - 1} = x^5 \quad \text{and} \quad \rho(x) = \sum_{d} \rho_d x^{d-1} = x^2.$$

If we start with $p_1 = \epsilon$ and run the recursions (1) for 100 iterations, we get the following evolution as a function of ϵ:



Note that for small ϵ, very few iterations are sufficient to achieve 0 residual errors, while for ϵ close to the threshold ≈ 0.43 many iterations are required and contribute diminishing improvements. How would this figure differ from its counterpart obtained by simulation?

Q: For which ϵ will the function $\epsilon[1-(1-x)^5]^2$ have fixed points?

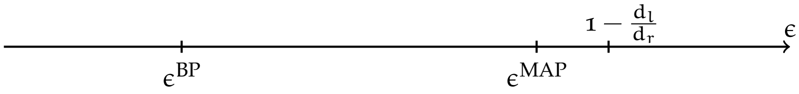
A: When $\epsilon > \epsilon^*$ where

$$\epsilon^* = \min_{x\in[0,1]} \frac{x}{[1-(1-x)^5]^2}.$$

For $\epsilon < 0.4294$, $\epsilon[1-(1-x)^5]^2$ has no fixed point $\implies p_i \to 0$ as $i \to \infty$.
For $\epsilon = 0.4295$, $\epsilon[1-(1-x)^5]^2$ gets a fixed point at 0.2652.

Bounds on the Decoding Threshold

The $(3,6)$ code is a rate $1/2$ code. What happens when more than a half of the data is erased? What happens when $\epsilon = 1/2$? What happens between $\epsilon = \epsilon^{BP}$ and $\epsilon = 0.5$?



For the $(3,6)$ code, we have $\epsilon^{BP} = 0.4294$, $\epsilon^{MAP} = 0.4881$, and $1-\frac{d_l}{d_r} = 0.5$.

LDPC Codes have very good performance, but they do have error floors. Theoretical proofs are mostly limited to BEC, but there are computational methods to estimate the performance in general. Irregular LDPC codes perform better, but their degree distributions have to be designed with a channel in mind as they do not perform universally well for all channels.

LDPC Codes & Statistical Physics

The Ising Model of Magnetism

Consider a physical system where n electrons with spins $\sigma_i \in \{+1,-1\}$ are arranged in a d-dimensional lattice[2] Λ. A spin configuration is an assignment of a spin value to each electron in Λ

Each lattice site j has an external magnetic field h_j interacting with it, and for any two adjacent lattice sites i and j, there is spin coupling (an interaction) characterized by J_{ij}. The energy of a configuration is given by the Hamiltonian function:

$$H(\sigma_1,\ldots,\sigma_n) = -\sum_{\langle i\ j\rangle} J_{ij}\sigma_i\sigma_j - \mu\sum_j h_j\sigma_j$$

The notation $\langle i\ j\rangle$ indicates that sites i and j are nearest neighbors, and μ denotes the magnetic moment.

[2] A lattice in $\mathbb{R}^d$ is a subgroup of $\mathbb{R}^d$ which is isomorphic to $\mathbb{Z}^d$ and spans $\mathbb{R}^d$.

The probability of a state with configuration $\sigma_1, \dots, \sigma_n$ in equilibrium follows the Boltzmann distribution:

$$P_\beta(\sigma_1, \dots, \sigma_n) \propto \exp\left[-\frac{H(\sigma_1, \dots, \sigma_n)}{k_B T}\right]$$

where k_B is the Boltzmann constant and T is the temperature. At high temperature, the spin system resembles a liquid At low temperature, it has minimum energy w.h.p. and can freeze into a ground state. For $d \geqslant 2$, there is a phase transition at a critical temperature.

LDPC Codes

Spin systems are mathematically similar to LDPC codes in that there is a global order from local interactions. Codewords are ordered crystalline structures. Code is defined using generalized coupling coefficients between the variable nodes. Local observations correspond to the local magnetic fields h_i.

Between the BP and the MAP thresholds, the LDPC system corresponds to a supercooled liquid. Correct answer (crystalline state) has minimum energy w.h.p. but spontaneous crystallization (i.e., decoding) does not occur w.h.p. The minimium-energy configuration corresponds to the MAP solution.

Spatially Coupled Codes on BEC

Spatially-Coupled (SC) LDPC codes are constructed by coupling many regular LDPC codes in a chain. To design a (d_l, d_r, L) SC-LDPC code, we start with L copies of uncoupled (d_l, d_r) LDPC codes, and then re-connect edges randomly with w neighboring codes.

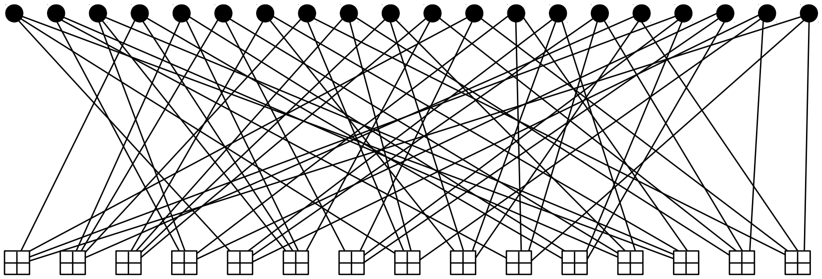
$$\lim_{w \leftarrow \infty} \epsilon^{BP}(d_l, d_r, L) = \epsilon^{MAP}(d_l, d_r)$$

Code blocks are spatially coupled by spreading edges over time. The resulting graph has a structured irregularity, which leads to wave-like decoding.

Spatially coupled codes came out of the "convolutional codes" ideas and community. However, their analysis is inspired by the connections with statistical mechanics.

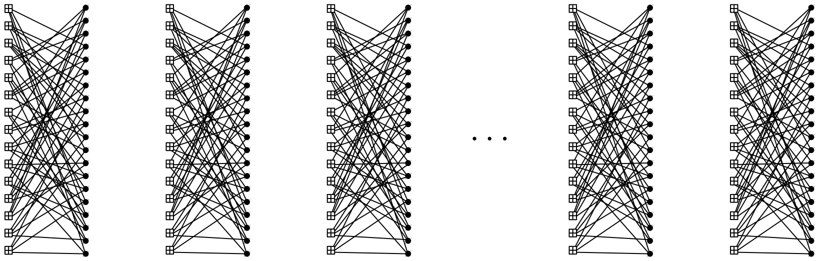
An Illustration by Dan Costello (check out the video)

The Tanner graph of the original code:



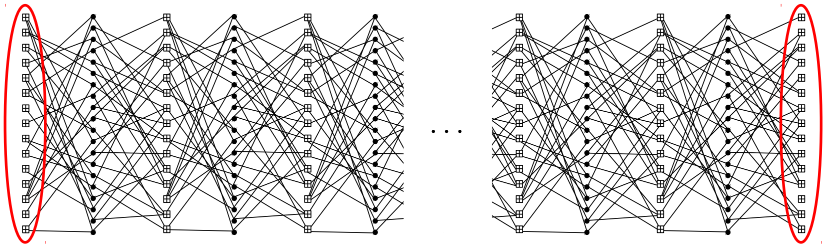
We start with a $(3, 4)$ regular LDPC code ...

L copies of the Tanner graph:



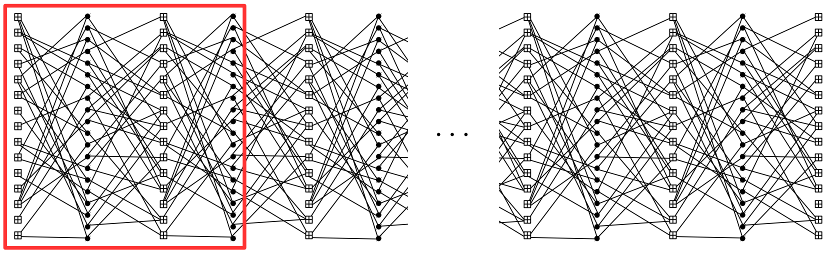
... and copy the graph L times.

Coupling of the neighboring copies:

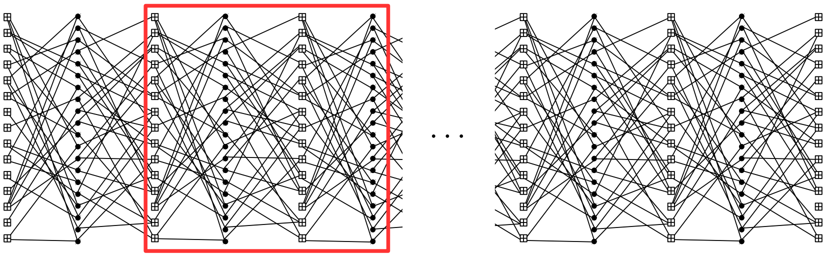


We then disconnect some edges from their original check nodes and connect them to the checks in other copies of the original graph. Note that we will have to add some additional check nodes, which means a reduction in code rate.

Sliding window decoding:



The resulting graph has a structured irregularity, which leads to wave-like decoding.



The decoding process is similar to a water freezing experiment you can see here.

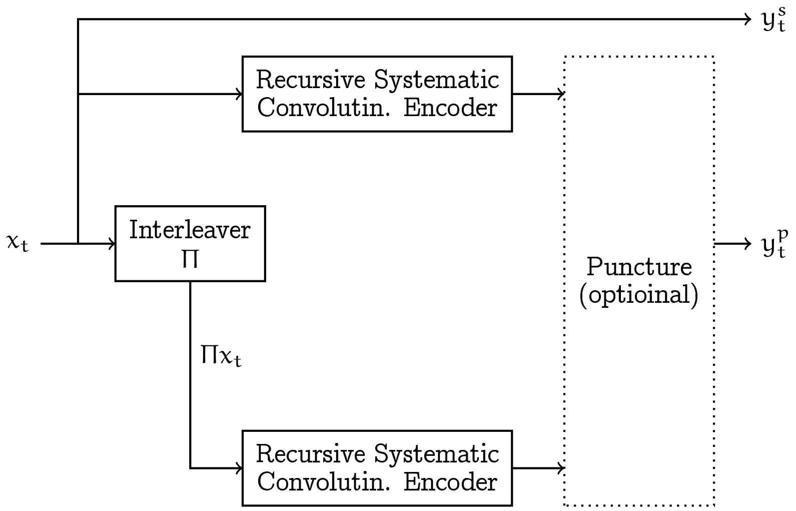
Error Control Coding [1]

Prof. Emina Soljanin

Lecture #20, April 16

This lecture is about HARQ.

Turbo Codes Encoders


y_t^s
Recursive Systematic Convolutin. Encoder
x_t
Interleaver Π
Puncture (optioinal)
y_t^p
Πx_t
Recursive Systematic Convolutin. Encoder

Turbo (mother) code in 3GPP has the following constituent code

$$G(D) = \begin{bmatrix} 1 & \frac{1+D+D^3}{1+D^2+D^3} & \frac{1+D+D^2+D^3}{1+D^2+D^3} \end{bmatrix}$$

Note that the code is systematic and recursive. Its rate is 1/5, but it is used at rates 1/2, 1/3, 1/4, and 1/5 with puncturing.

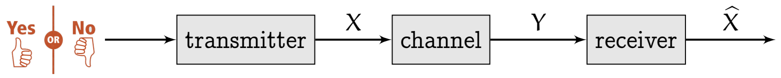
Communications Channel Models

A General Model

A communications channel involves at least 2 random variables:

X – the input; its range Ω_X is called the input alphabet

Y – the output; its range Ω_Y is called the output alphabet



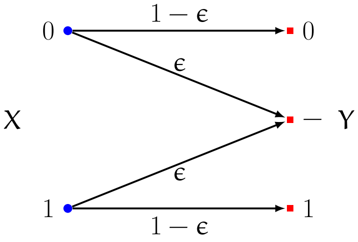
The relation between the input and the output is described by, e.g.,

[1] Rutgers, ECE 548, Spring 2019

- the conditional PDF of the output given the input $W(Y \mid X)$ (we call W the transition probability)
- a noise RV Z added to the input s.t. $Y = X + Z$.

Binary Erasure Channel BEC(ϵ)

Binary input and ternary output: $\Omega_X = \{0, 1\}$, $\Omega_Y = \{0, 1, -\}$
$W(1 \mid 0) = W(0 \mid 1) = 0$
$W(0 \mid 0) = W(1 \mid 1) = 1 - \epsilon$
$W(- \mid 0) = W(- \mid 1) = \epsilon$

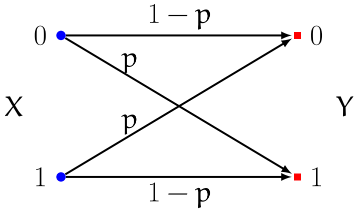


The Binary Symmetric Channel BSC(p)

Binary input and output: $\Omega_X = \Omega_Y = \{0, 1\}$
$W(0 \mid 0) = W(1 \mid 1) = 1 - p$
$W(1 \mid 0) = W(0 \mid 1) = p$
We can instead say

$$Y = X + Z \mod 2 \qquad \leftarrow \text{binary addition, XOR}$$

where $Z \sim \text{Bernoulli}(p)$



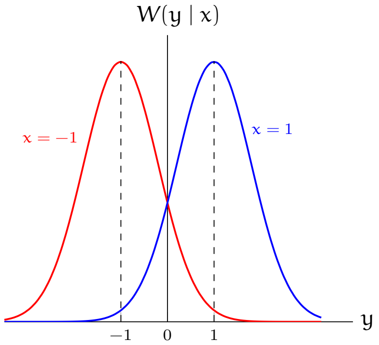
Binary Input Additive Gaussian Noise Channel

-1 or 1 input and real-valued output: $\Omega_X = \{-1, 1\}$, $\Omega_Y = \mathbb{R}$

$$W(y \mid -1) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x+1)^2}{2\sigma^2}}$$

$$W(y \mid 1) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-1)^2}{2\sigma^2}}$$

We can instead say $Y = X + Z$ where $Z \sim N(0, \sigma^2)$.



A Bound on Error Rate:

For a channel with binary input alphabet $\{0, 1\}$ and discrete output alphabet $\mathcal{Y}$, we have two probability mass function $W(Y|0)$ and $W(Y|1)$. The Bhattacharyya noise parameter[2] for the channel is defined as follows:

$$\gamma = \sum_{b \in \mathcal{Y}} \sqrt{W(b|0)W(b|1)}.$$

In the case of a time invariant and memoryless channel with the parameter γ, if two sequences are at Hamming distance d apart, then the receiver will confuse one for the other with probability lower than γ^d.

[2] Bhattacharyya coefficient is defined for any two discrete or continuous probability distributions measures, and it measures how close they are to each other.