編碼理論與實驗

翁詠祿、謝欣霖、陳彥銘

Outline

Part I: Theory

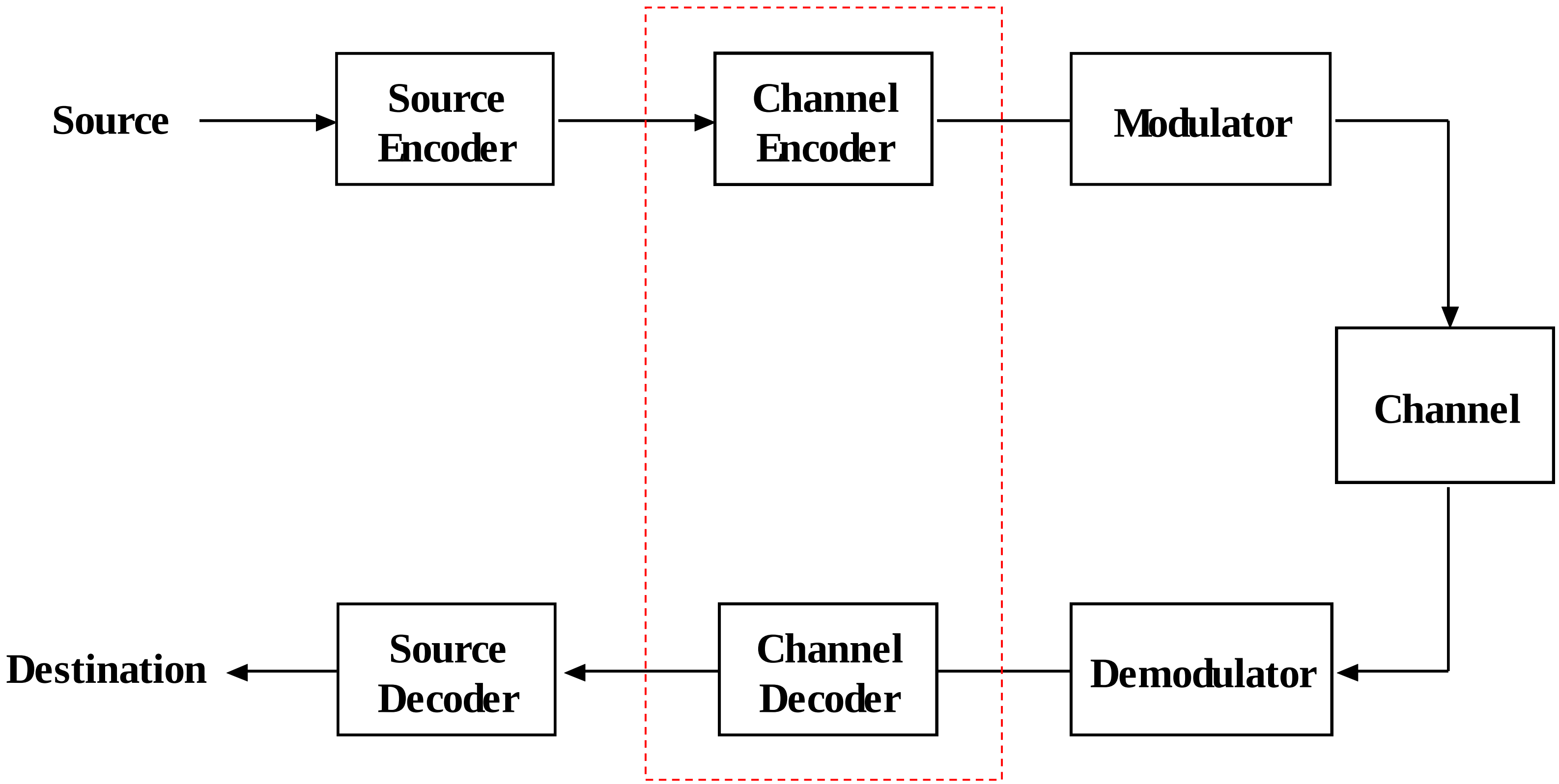
- ❑ Introduction to channel coding theory
- ❑ Introduction to Linear Block Code
- ❑ Introduction to the theory of Low Density Parity Check code (LDPC)
- ❑ Encoding and Decoding Algorithm of LDPC code
- ❑ Introduction to the theory of Polar Code
- ❑ Encoding and Decoding Algorithm of Polar code

Outline

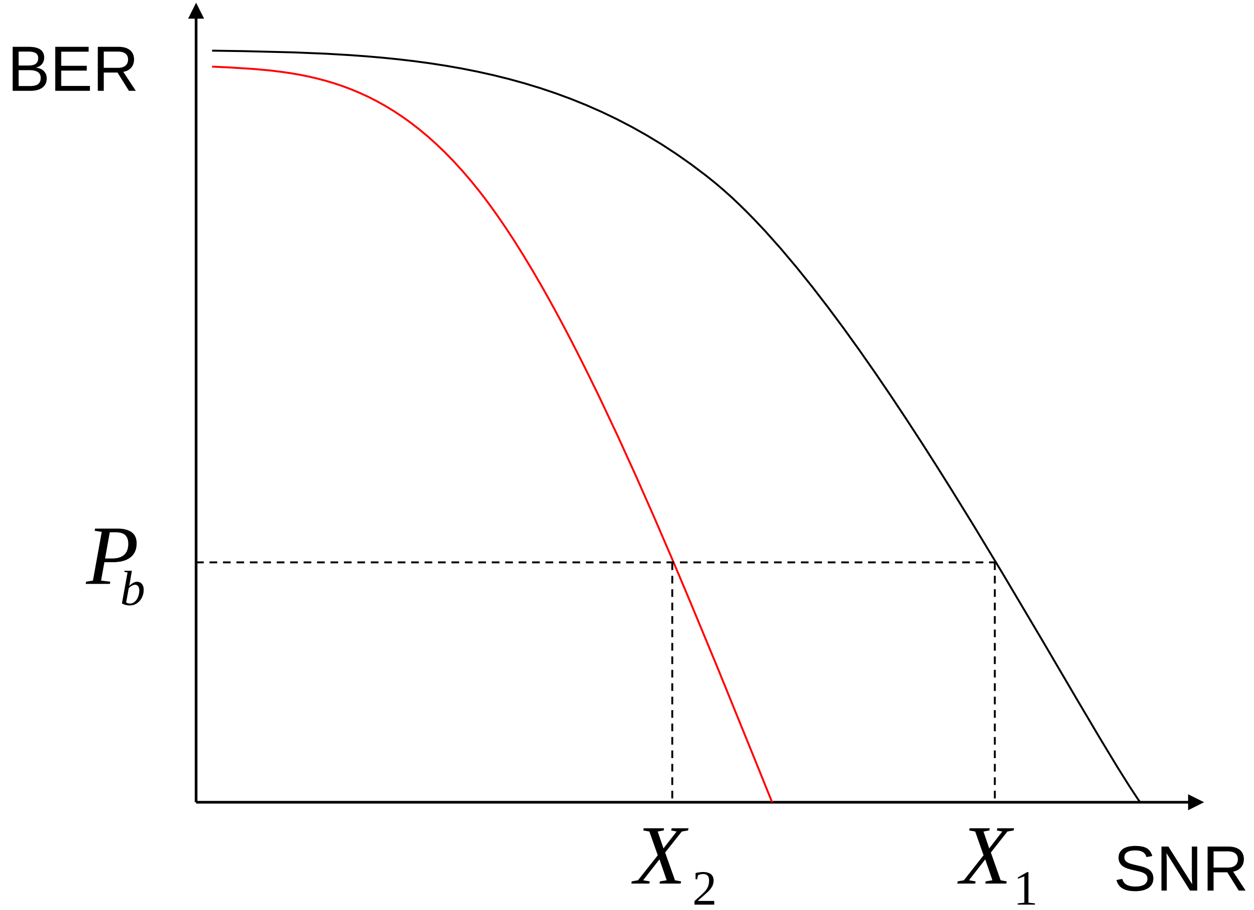
Part II: Experiment

- ❑ Introduction to Matlab/Simulink
- ❑ Introduction to Zedboard
- ❑ Hamming code Enc / Dec experiment
- ❑ LDPC code Enc / Dec experiment
- ❑ Polar code Enc / Dec experiment
- ❑ QPSK transceiver experiment with Enc / Dec capability

Introduction to channel coding theory

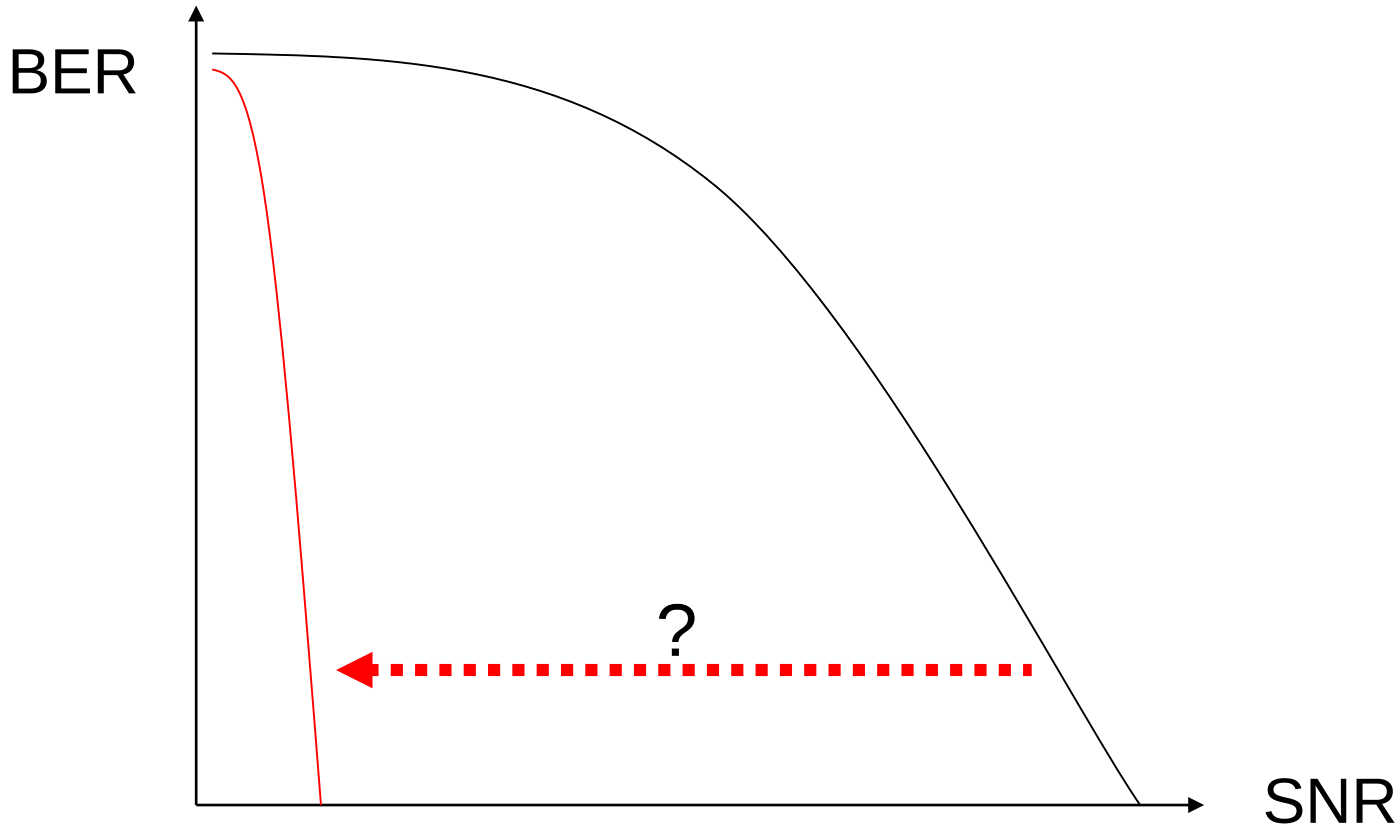


Why channel coding?



BSC: P=0.1

What's the performance limit?



What “A Mathematical Theory of Communication” Told Us in 1948:

- Associated with a communication channel - typically defined in probabilistic terms - is a number called the *capacity* of the channel.
- Significance: If the capacity of a channel is C bits, then:
  - It is possible to convey information reliably over the channel at a rate of up to C bits per channel use. (“Reliably” meaning with error rates as low as you want them.)
  - It is impossible to convey information reliably over the channel at rates greater than C bits per channel use.

Claude E. Shannon (1916-2001)

- Examples of capacity:
  - The capacity of a binary symmetric channel with crossover probability p is $C_{\mathrm{BSC}} = 1 - h(p)$, where $h(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ is the binary entropy function.

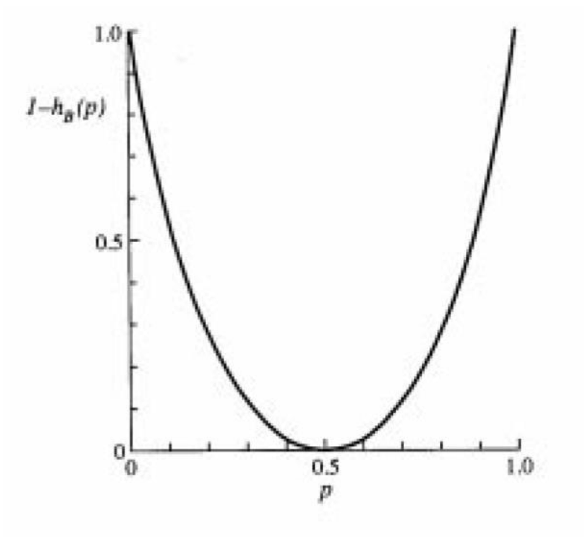


Figure 1: Capacity of a binary symmetric channel as a function of the crossover probability.

- The capacity of the complex-input additive white Gaussian noise channel (with only a power constraint on the input) is $C_{\text{AWGN}} = \log_2(1 + \text{SNR})$.
- For channels using particular signal constellations - e.g., QPSK, 16-QAM, etc. - the capacity can be computed numerically based on the *mutual information* between the channel's input and output.
  * If the 2-D signal constellation has M signals, then the capacity approaches $\log_2(M)$ bits/channel use at high SNR.

Unfortunately:

- Shannon didn't tell us how to construct channel codes that obtain the performance promised by his information theory results.

But he did give us these lessons:

- Good codes should have very long codewords.
- Good codes should appear random.

Linear Block Codes

Introduction

A binary block code of length n with 2^k codewords is called an (n, k) linear block code iff its 2^k codewords form a k dimensional subspace of the vector space V of all the n-tuples over $GF(2)$.

For a binary (n, k) linear block code C, there exists k linear independent codewords $\mathbf{g_0}, \mathbf{g_1}, \ldots, \mathbf{g_{k-1}}$ or basis such that every codeword $\mathbf{v}$ in C is a linear combination of these k linearly independent codewords.

Let $\mathbf{u} = (u_0, u_1, ..., u_{k-1})$ be the message to be encoded. The codeword $\mathbf{v} = (v_0, v_1, ..., v_{n-1})$ for this message is given by

$$\mathbf{v} = u_0\mathbf{g_0} + u_1\mathbf{g_1} + \cdots + u_{k-1}\mathbf{g_{k-1}}$$

$$= \mathbf{u} \cdot \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \mathbf{u} \cdot \mathbf{G}$$

where $\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix}$

is a generator matrix of $\mathbf{C}$.

$$
\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{n-k-1} \end{bmatrix}
= \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots & h_{0,n-1} \\ h_{1,0} & h_{1,1} & \cdots & h_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k-1,0} & h_{n-k-1,1} & \cdots & h_{n-k-1,n-1} \end{bmatrix}
$$

Then $\mathbf{H}$ is a generator matrix of C_d.

Cyclic codes

Let $\mathbf{v} = (v_0, v_1, v_2, , \ldots, v_{n-1})$ be an n-tuple over $GF(2)$.
If we shift every component of $\mathbf{v}$ cyclically one place to the right, we obtain $\mathbf{v}^{(1)} = (v_{n-1}, v_0, v_1, \ldots, v_{n-2})$ which is called right cyclic-shift of $\mathbf{v}$.

An (n, k) linear block code $\mathbf{C}$ is said to be cyclic if the cyclic-shift of each codeword in $\mathbf{C}$ is also a codeword in $\mathbf{C}$.

Polynomial representation

1. Code polunomial of $\mathbf{v}$:
   $\mathbf{v}(X) = v_0 + v_1 X + v_2 X^2 + \cdots + v_{n-1} X^{n-1}$.

2. In an (n, k) cyclic code $\mathbf{C}$, there exists one and only one code polynomial $\mathbf{g}(X)$ of dgree $n-k$ of the following form
   $\mathbf{g}(X) = 1 + g_1 X + g_2 X^2 + \cdots + g_{n-k-1} X^{n-k-1} + X^{n-k}$
   This unique nonzero code polynomial $\mathbf{g}(X)$ of minimum degree in C is called the generator polynomial of the (n, k) cyclic code C.

3. $\mathbf{g}(X)$ divides $X^n + 1$. Consequently, $X^n + 1$ can be expressed as
   $X^n + 1 = g(X)\mathbf{f}(X)$, where $\mathbf{f}(X) = 1 + f_1 X + \cdots + f_{k-1} X^{k-1} + X^k$
   Let $\mathbf{h}(X) = X^k \mathbf{f}(X^{-1}) = 1 + h_1 X + \cdots + h_{k-1} X^{k-1} + X^k$ be the reciprocal polynomial of $\mathbf{f}(X)$. Then

$$\mathbf{H} = \begin{bmatrix} 1 & h_1 & h_2 & \dots & h_{k-1} & 1 & 0 & \dots & 0 \\ 0 & 1 & h_1 & \dots & h_{k-2} & h_{k-1} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & h_1 & h_2 & \dots & h_{k-1} \end{bmatrix}$$

is a PCM of the (n, k) cyclic code C. The polynomial $\mathbf{h}(\mathbf{X})$ is called the parity-check polynomial of C. In fact, $\mathbf{h}(\mathbf{X})$ is the generator polynomial of the dual code, an $(n, n-k)$ cyclic code, of the (n, k) cyclic code C.

Low-Density Parity-Check Codes

Outline

- ❑ Introduction
- ❑ Decoding Algorithm of LDPC Codes
- ❑ Classifications of LDPC Codes
- ❑ Analysis of LDPC Codes
- ❑ LDPC Codes in Practical System

Introduction

- Low-density parity-check (LDPC) codes are a class of linear block codes which provide near-capacity performance on many channels.
- We only consider binary LDPC codes in this lecture.

Introduction

- History:
  - LDPC codes were invented by Gallager in his 1960 doctoral dissertation.
  - In 1981, Tanner generalized LDPC codes and introduced a graphical representation of LDPC codes, now called a Tanner graph.
  - The study of LDPC codes was resurrected in the mid 1990s with the work of MacKay, Luby, and others

Introduction

- Advantages over turbo codes:
  - Do not require a long interleaver for near capacity performances.
  - Lower error floor value.
  - More simple decoding architecture.
  - More flexible for code design.

Matrix Representation

- An LDPC code is a linear block code given by the null space of an $m \times n$ parity-check matrix $\boldsymbol{H}$ that has a low density of 1's
- A density of 0.01 or lower can be called low density
  - But no stringent restriction
- A *regular LDPC code* is a linear block code whose $\boldsymbol{H}$ has column weight g and row weight r, where $r = g(n/m)$ and $g \ll m$. Otherwise, it is an *irregular LDPC code*

- Almost all LDPC code constructions impose the following additional structure property on $\boldsymbol{H}$: no two rows(or two columns) have more than one position in common that contains a nonzero element. This is called *row-column constraint* (RC constraint).
- The low density aspect of LDPC codes accommodates iterative decoding which has near-maximum-likelihood performance at error rates of interest for many applications.

- The code rate R for a regular LDPC code is bounded as

$$R \geq 1 - \frac{m}{n} = 1 - \frac{g}{r},$$

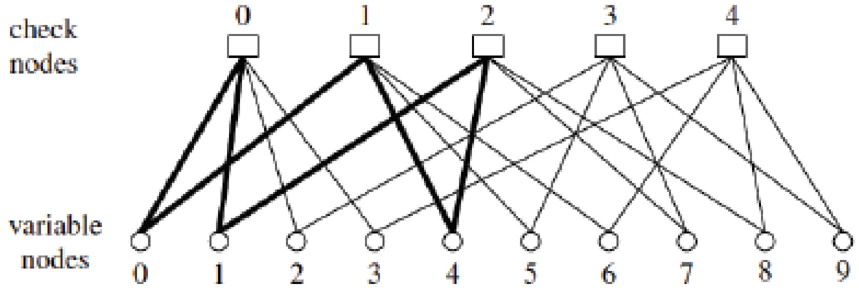
with equality when $\boldsymbol{H}$ is full rank.

- Graphical Representation
  - The *Tanner graph* of an LDPC code is analogous to the trellis of a convolutional code.
  - A Tanner graph is a *bipartite graph* whose nodes may be separated into two types, with edges connecting only nodes of different types.

- The two types in a Tanner graph are the *variable nodes* (VNs) (or *code-bit nodes*) and the *check nodes*(CNs) (or *constraint nodes*).
- The Tanner graph of a code is drawn as follows: CN *i* is connected to VN *j* whenever element h_{ij} in $\boldsymbol{H}$ is a 1.
- There are *m* CNs in a Tanner graph, one for each check equation (row of $\boldsymbol{H}$), and *n* VNs, one for each code bit (column of $\boldsymbol{H}$)
- The allowable *n*-bit words represented by the *n* VNs are the codewords in the code.

- Example of Tanner Graph: A (10,5) code with $g = w_c = 2, r = w_r = 4$, and the following $\boldsymbol{H}$ matrix:

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



- A sequence of edges forms a closed path in a Tanner graph is called a *cycle*.
- Cycles force the decoder to operate locally in some portions of the graph so that a globally optimal solution is impossible.
- At high densities, many short cycles will exist, thus precluding the use of an iterative decoder.
- The length of a cycle is equal to the number of edges in the cycle.

- The minimum cycle length in a given bipartite graph is called the graph's *girth*.
- The shortest possible cycle in a bipartite graph is a length-4 cycle.
- Such cycles manifest themselves in the $\boldsymbol{H}$ matrix as four 1s that lie on the four corners of a rectangular submatrix of $\boldsymbol{H}$
- RC constraint eliminates length-4 cycles
- The number of edges in a Tanner graph is $mr = ng$

- The original LDPC codes are random in the sense that their parity-check matrices possess little structure.
- Both encoding and decoding become quite complex when the code possesses no structure beyond being a linear code.
- The nominal parity-check matrix $\boldsymbol{H}$ of a cyclic code is a $n \times n$ circulant; that is, each row is a cyclic-shift of the one above it, with the first row a cyclic-shift of the last row.

- The implication of a sparse circulant matrix $\boldsymbol{H}$ for LDPC decoder complexity is substantial.
- Beside being regular, a drawback of cyclic LDPC codes is that the nominal $\boldsymbol{H}$ matrix is $n \times n$, independently of the code rate, implying a more complex decoder.
- Another drawback is that the known cyclic LDPC codes tend to have large row weights, which makes decoder implementation tricky.

- Quasi-cyclic (QC) codes also possess tremendous structure, leading to simplified encoder and decoder designs.
- They permit more flexibility in code design, particularly irregularity, and, hence, lead to improved codes relative to cyclic LDPC codes.

- The $\boldsymbol{H}$ matrix of a QC code is generally represented as an array of circulants, e.g.,

$$\boldsymbol{H} = \begin{pmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{M1} & \cdots & A_{MN} \end{pmatrix},$$

where each matrix A_{rc} is a $Q \times Q$ circulant

- To affect irregularity, some of the circulants may be the all-zeros $Q \times Q$ matrix using a technique called masking
- For irregular LDPC codes, it is usual to specify the VN and CN degree-distribution polynomials, denoted by $\lambda(\mathrm{X})$ and $\rho(X)$, respectively.
  - In the polynomial,

$$\lambda(\mathrm{X}) = \sum_{\mathrm{d=1}}^{\mathrm{dv}} \lambda_d X^{d-1} \; [\rho(X) = \sum_{d=1}^{d_c} \rho_d X^{d-1}]$$

$\lambda_{\mathrm{d}}(\rho_{\mathrm{d}})$: the fraction of all edges connected to degree-d $VNs(CNs)$

$d_v(d_c)$: the maximum $VN(CN)$ degree

- For (2,4)-regular LDPC code, we have $\lambda(X) = X, \rho(X) = X^3$
- Let us denote the number of VNs (CNs) of degree d by $N_v(d)$ $(N_c(d))$. Let E denote the number of edges in the graph.

$$E = \frac{n}{\int_0^1 \lambda(X)dX} = \frac{m}{\int_0^1 \rho(X)dX}$$

$$N_v(d) = \frac{E\lambda_d}{d} = \frac{\frac{n\lambda_d}{d}}{\int_0^1 \lambda(X)dX}$$

$$N_c(d) = \frac{E\rho_d}{d} = \frac{\frac{m\rho_d}{d}}{\int_0^1 \rho(X)dX}$$

- The code rate R is bounded as

$$R \geq \left(1 - \frac{m}{n}\right) = 1 - \frac{\int_0^1 \rho(X)dX}{\int_0^1 \lambda(X)dX}$$

The polynomials $\lambda(\mathrm{X})$ and $\rho(X)$ represent a Tanner graph's degree

distributions from an "edge perspective"

- The degree distributions may also be represented from a "node perspective" using the notation $\tilde{\lambda}(\mathrm{X})(\tilde{\rho}(\mathrm{X}))$, where the coefficient $\tilde{\lambda}_d$ $(\tilde{\rho}_d)$ is the fraction of all VDs (CNs)that have degree d

$$\tilde{\lambda}_d = \frac{N_v(d)}{n} = \frac{n\lambda_d/d}{n\int_0^1 \lambda(X)\,dx} = \frac{\lambda_d/d}{\int_0^1 \lambda(X)\,dx}$$

$$\tilde{\rho}_d = \frac{\rho_d/d}{\int_0^1 \rho(X)\,dx}$$

- In addition to partitioning LDPC codes into three classes, i.e., cyclic, quasi-cyclic, and random (but linear), the LDPC code-construction techniques can be partitioned as well

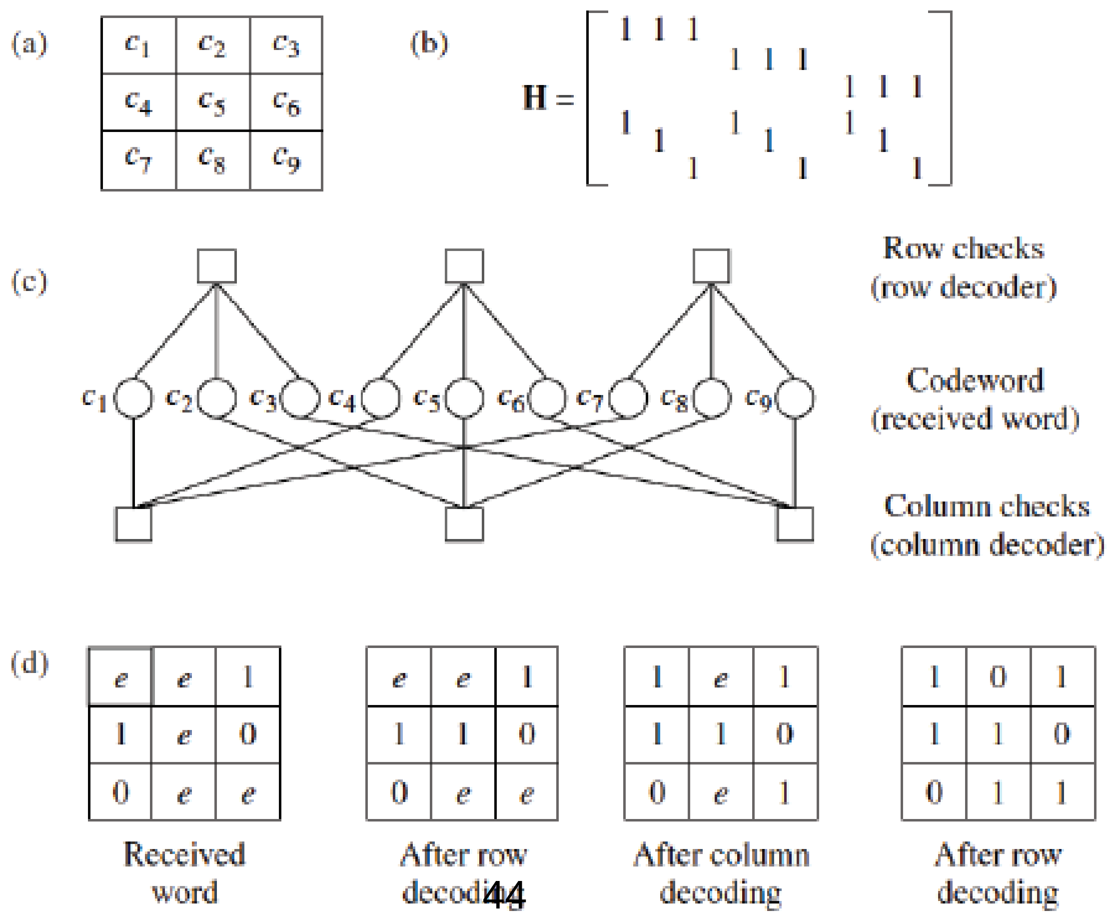
- The first class of construction techniques can be described as algorithmic or computer-based.
  - The computer-based construction techniques can lead to either random or structured LDPC codes.
- The second class of construction techniques consists of those based on finite mathematics, including algebra, combinatorics, and graph theory.
  - The mathematical construction techniques generally lead to structured LDPC codes, although exceptions exist.

Decoding Algorithm of LDPC Codes

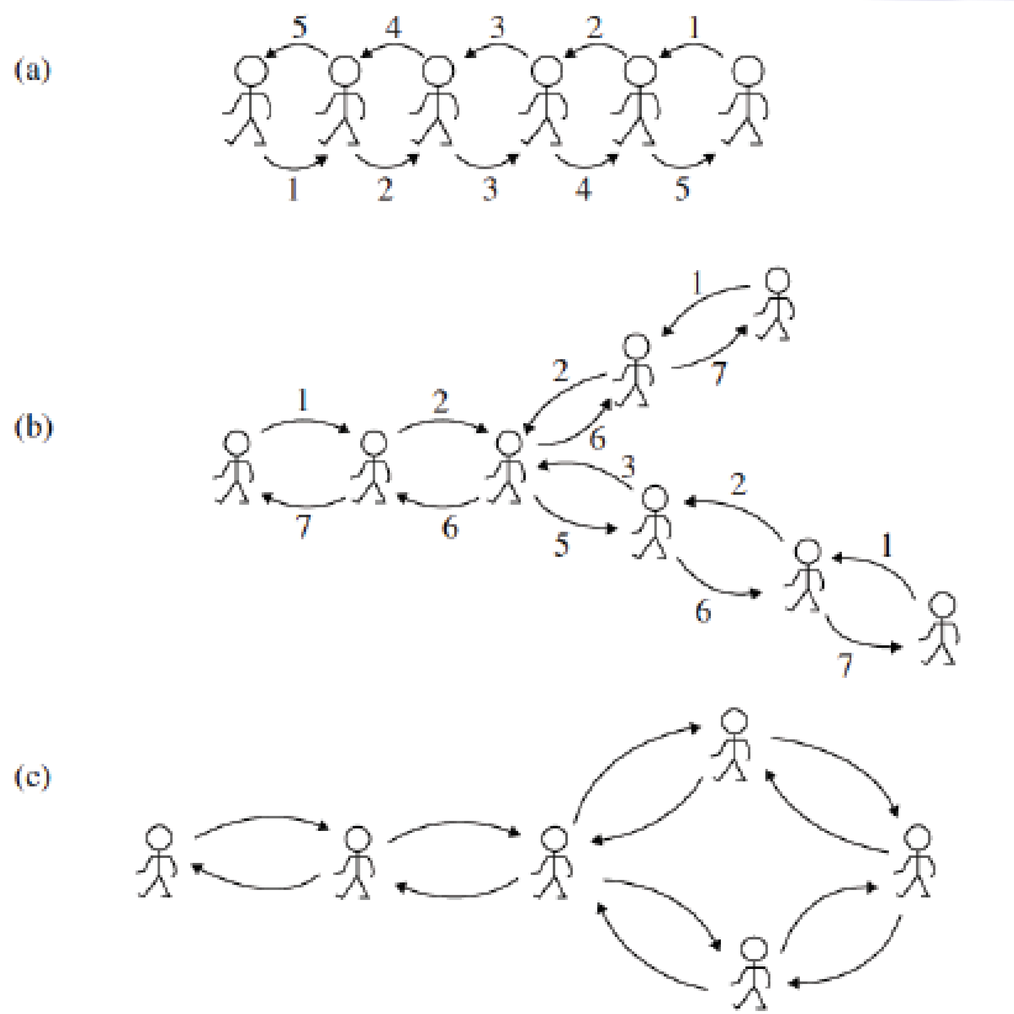
- Message Passing and the Turbo Principle
  - The key innovation behind LDPC codes is the low-density nature of the parity- check matrix, which facilitates iterative decoding.
  - *Sum-product algorithm* (SPA) is a general algorithm that provides near-optimal performance across a broad class of channels.
  - *Message-passing decoding* refers to a collection of low-complexity decoders working in a distributed fashion to decode a received codeword in a concatenated coding scheme.

- We can consider an LDPC code to be a generalized concatenation of many single parity-check (SPC) codes.
- A message-passing decoder for an LDPC code employs an individual decoder for each SPC code and these decoders operate cooperatively in a distributed fashion to determine the correct code bit values.

- Example: $(3,2) \times (3,2)$ SPC product code as an LDPC code



- Message-passing decoding for a collection of constituent decoders arranged in a graph is optimal provided that the graph contains no cycles, but it is not optimal for graphs with cycles.
- Consider the figure in next slide, which depicts six soldiers in a linear formation. The goal is for each of the soldiers to learn the total number of soldiers present by counting in a distributed fashion.



Distributed soldier counting. (a) Soldiers in a line. (b) Soldiers in a tree formation. (c) Soldiers in a formation containing a cycle

- Consider (*b*). The message that an arbitrary soldier X passes to arbitrary neighboring soldier Y is equal to the sum of all incoming messages, plus one for soldier X, minus the message that soldier Y had just sent to soldier X
- This message-passing rule introduces the concept of *extrinsic information*.

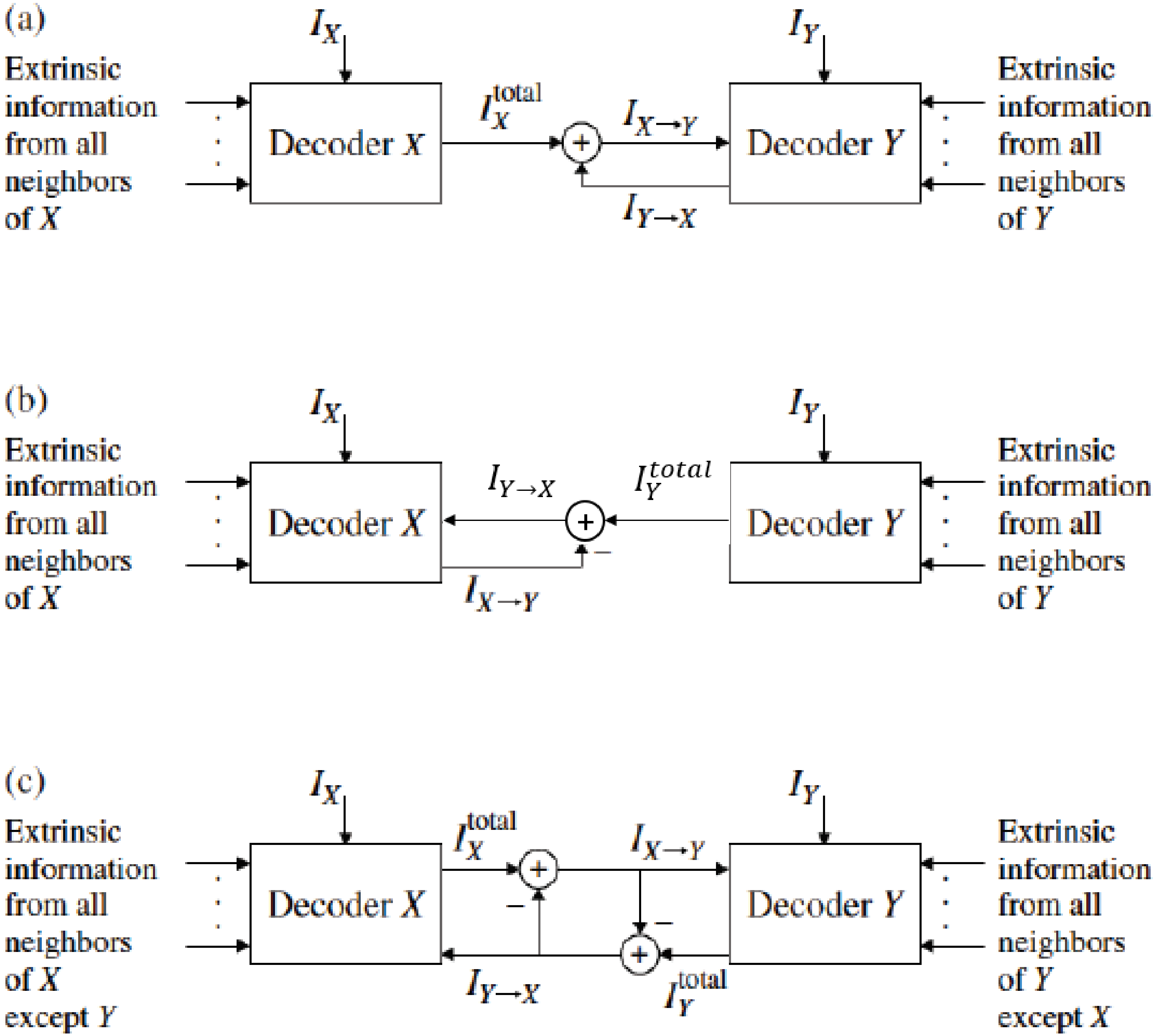
- The idea is that a soldier does not pass to a neighboring soldier any information that the neighboring soldier already has, that is, only extrinsic information is passed.
- We say that soldier X passes to soldier Y only extrinsic information, which may be computed as

$$
\begin{aligned}
I_{X\to Y} &= \sum_{Z\in N(X)} I_{Z\to X} - I_{Y\to X} + I_X \\
&= \sum_{Z\in N(X)-\{Y\}} I_{Z\to X} + I_X
\end{aligned}
$$

where $N(X)$ is the set of neighbors of soldier X, $I_{X \to Y}$ is the extrinsic information sent from solider X to solider Y

- I_X is the “one" that a soldier counts for himself and I_X is called the *intrinsic information*.
- Consider (*c*). There is a cycle and the situation is unstable.
- While most practical codes contain cycles, it is well known that message-passing decoding performs very well for properly designed codes for most error-rate ranges of interest

- The notion of extrinsic-information passing described above has been called the *turbo principle* in the context of the iterative decoding of concatenated codes in communication channel.
- A depiction of the turbo principle is contained in next slide.


(a)
I_X
I_Y
Extrinsic information from all neighbors of X
Decoder X
I_X^{total}
$I_{X\to Y}$
$I_{Y\to X}$
Decoder Y
Extrinsic information from all neighbors of Y
(b)
I_X
I_Y
Extrinsic information from all neighbors of X
Decoder X
$I_{Y\to X}$
I_Y^{total}
$I_{X\to Y}$
Decoder Y
Extrinsic information from all neighbors of Y
(c)
I_X
I_Y
Extrinsic information from all neighbors of X except Y
Decoder X
I_X^{total}
$I_{X\to Y}$
$I_{Y\to X}$
I_Y^{total}
Decoder Y
Extrinsic information from all neighbors of Y except X

- The Sum-Product Algorithm (SPA)
  - We derive the sum-product algorithm for general memoryless binary-input channels, applying the turbo principle in our development.
  - The optimality criterion underlying the development of the SPA decoder is symbol-wise *maximum a posteriori* (MAP).

- We are interested in computing the *a posteriori probability* (APP) that a specific bit in the transmitted codeword

  $v = (v_0, v_1, \dots, v_{n-1})$

  equals 1, given the received word

  $y = (y_0, y_1, \dots, y_{n-1})$.
- Without loss of generality, we focus on the decoding of bit v_j

  and calculate $\Pr(v_j|y)$.

- The APP ratio and log-likelihood ratio (LRR) are

$$\ell(v_j|y) = \frac{\Pr(v_j=0|y)}{\Pr(v_j=1|y)}$$

and

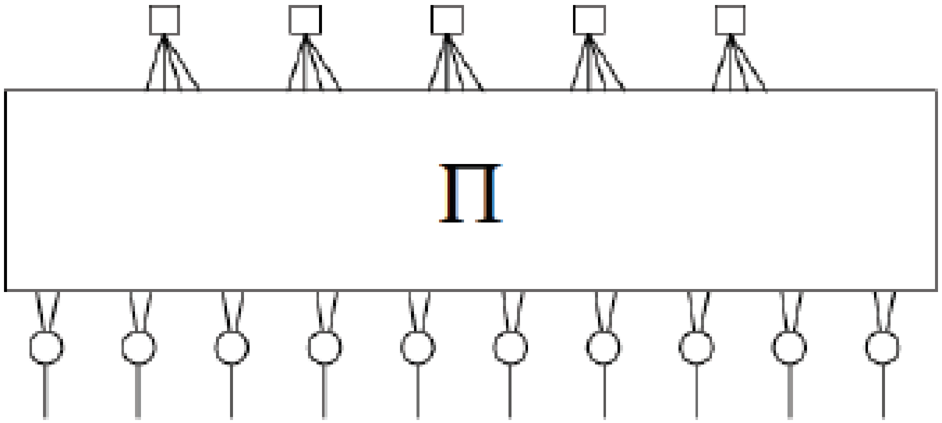
$$\mathrm{L}(v_j|y) = \log\left(\frac{\Pr(v_j = 0|y)}{\Pr(v_j = 1|y)}\right)$$

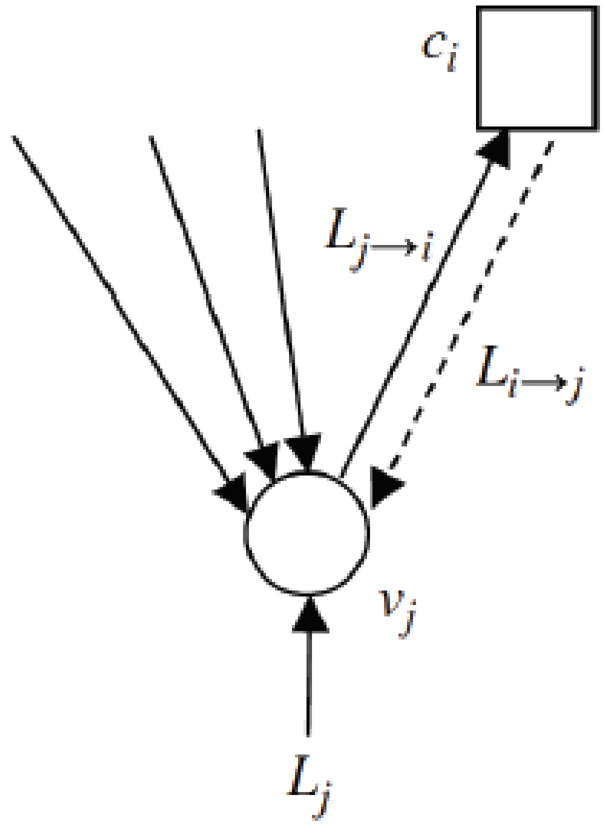
respectively.

- The natural logarithm is assumed for LLRs.

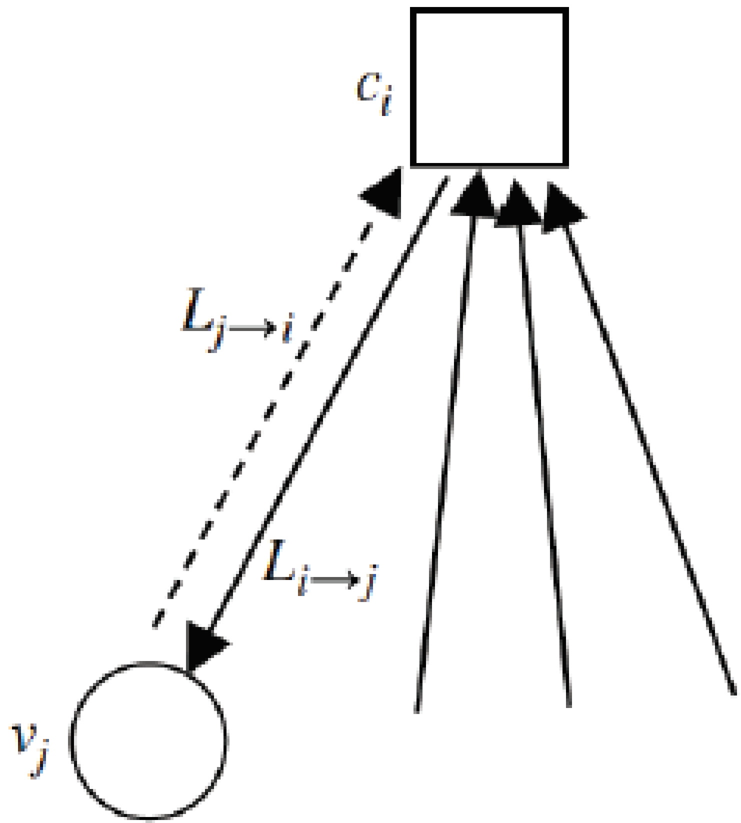
- The SPA for the computation of $\Pr(v_j = 1 \mid \boldsymbol{y})$, $\ell(v_j \mid \boldsymbol{y})$, or $L(v_j \mid \boldsymbol{y})$ is a distributed algorithm that is an application of the turbo principle to a code's Tanner graph.
- An LDPC code can be deemed a collection of SPC codes concatenated through an interleaver to a collection of repetition (REP) codes.

- The SPC codes are treated as outer codes, that is, they are not connected to the channel.
- The following is a graphical representation of an LDPC code as a concatenation of SPC and REP codes. "∏ " represents an interleaver.





- The figure depicts the REP (VN) decoder .
- The VN j decoder receives LLR information both from the channel and from its neighbors.



- In the computation of the extrinsic information $L_{j\to i}$, VN j need not receive $L_{i\to j}$ from CN i since it would be subtracted out anyway.
- The above figure depicts the SPC (CN) decoder situation.

- The VN and CN decoders work cooperatively and iteratively to estimate $L(v_j|y)$ for $j = 0,1,\dots,n-1$.
- Assume that the *flooding schedule* is employed.
- According to this schedule, all VNs process their inputs and pass extrinsic information up to their neighboring CNs; the CNs then process their inputs and pass extrinsic information down to their neighboring VNs; and the procedure repeats, starting with the VNs.

- After a preset maximum number of repetitions (or iterations) of this VN/CN decoding round, or after some stopping criterion has been met, the decoder computes (estimates) the LLRs $L(v_j|\mathrm{y})$ from which decisions on the bits v_j are made.
- When the cycles are large, the estimates will be very accurate and the decoder will have near-optimal (MAP) performance.

Repetition Code MAP Decoder and APP Processor

- At this point, we need to develop the detailed operations within each constituent (CN and VN) decoder.
- Consider a REP code in which the binary code symbol $c \in \{0,1\}$ is transmitted over a memoryless channel d times so that the d-vector r is received.

- The MAP decoder computes the log-APP ratio

$$\mathrm{L}(c|r) = \log\left(\frac{\Pr(c = 0|r)}{\Pr(c = 1|r)}\right)$$

- which is equal to

$$\mathrm{L}(c|r) = \log\left(\frac{\Pr(r|c = 0)}{\Pr(r|c = 1)}\right)$$

- under an equally likely assumption for the value of *c*.

- This simplifies as

$$\mathrm{L}(c|r) = \log\left(\frac{\prod_{l=0}^{d-1}\Pr(r_l|c=0)}{\prod_{l=0}^{d-1}\Pr(r_l|c=1)}\right)$$

$$= \sum_{l=0}^{d-1} log\left(\frac{\Pr(r_l|c=0)}{\Pr(r_l|c=1)}\right)$$

$$= \sum_{l=0}^{d-1} L(\, r_l|c) = \sum_{l=0}^{d-1} L(\, c|r_l)$$

where $L(r_l|c)$ and $L(c|r_l)$ are obviously defined.

- The MAP receiver for a REP code computes the LLRs for each channel output r_l and adds them. The MAP decision is $\hat{c} = 0$ if $L(c|r) \geq 0$ and $\hat{c} = 1$ otherwise.

- In the context of LDPC decoding, the above expression is adapted to compute the extrinsic information to be sent from VN j to CN i,

$$L_{j\to i} = L_i + \sum_{i' \in N(j)-\{i\}} L_{i'\to j}$$

- The quantity L_j in this expression is the LLR value computed from the channel sample y_i ,

$$L_j = L(c_j | y_j).$$

- In the context of LDPC decoding, we call the VN an APP processor instead of a MAP decoder. At the last iteration, VN j produces a decision based on

$$L_j^{\text{total}} = L_j + \sum_{i \in N(j)} L_{i \to j}$$

- Single-Parity-Check Code MAP Decoder and APP Processor
    - To develop the MAP decoder for an SPC code we first need the following result due to Gallager.

- Consider a vector of d independent binary random variables $a = (a_0, a_1, \ldots, a_{d-1})$ in which $\Pr(a_l = 1) = p_1^{(l)}$ and $\Pr(a_l = 0) = p_0^{(l)}$. Then the probability that a contains an even number of 1s is

$$\frac{1}{2} + \frac{1}{2}\prod_{l=0}^{d-1}\left(1 - 2p_1^{(l)}\right)$$

and the probability that a contains an odd number of 1's is

$$\frac{1}{2} - \frac{1}{2}\prod_{l=0}^{d-1}\left(1 - 2p_1^{(l)}\right)$$

- (Partial Proof)
- Assume that the above equations are true for $d = k$. Then the probability that a contains an even number of 1s for $d = k + 1$ is

$$\left(\frac{1}{2}+\frac{1}{2}\prod_{l=0}^{k-1}\left(1-2p_1^{(l)}\right)\right)\left(1-p_1^{(k)}\right)+\left(\frac{1}{2}-\frac{1}{2}\prod_{l=0}^{k-1}\left(1-2p_1^{(l)}\right)\right)\cdot p_1^{(k)}$$

$$= \frac{1}{2}+\frac{1}{2}\left\{\prod_{l=0}^{k-1}\left(1-2p_1^{(l)}\right)\left[\left(1-p_1^{(k)}\right)-p_1^{(k)}\right]\right\}$$

$$= \frac{1}{2}+\frac{1}{2}\prod_{l=0}^{(k+1)-1}(1-2p_1^{(l)})$$

- Consider the transmission of a length-d SPC codeword c over a memoryless channel whose output is $\boldsymbol{r}$.
- The bits c_l in the codeword c have a single constraint: there must be an even number of 1s in c. Without loss of generality, we focus on bit c_0, for which the MAP decision rule is

$$\widehat{c_0} = \arg \max_{b \in \{0,1\}} \Pr(c_0 = b | r, \mathrm{SPC}),$$

where the conditioning on SPC is a reminder that

there is an SPC constraint imposed on c.

- $\Pr(c_0 = 0|r, \mathrm{SPC})$
  $= \Pr(c_1, c_2, \ldots, c_{d-1} \text{ has an even no. of 1s}|r)$
  $= \frac{1}{2} + \frac{1}{2}\prod_{l=1}^{d-1}(1 - 2\Pr(c_l = 1|r_l))$.
- Rearranging gives

$$1 - 2\Pr(c_0 = 1|r, \mathrm{SPC}) = \prod_{l=1}^{d-1}(1 - 2\Pr(c_l = 1|r_l)),$$

where we used

$$\Pr(c_0 = 1|r, \mathrm{SPC}) = 1 - \Pr(c_0 = 1|r, \mathrm{SPC}).$$

- We can change this to an LLR representation using the easily proven relation for a generic binary random variable with probabilities p_1 and p_0,

$$1 - 2p_1 = \tanh\left(\frac{1}{2}\log\left(\frac{p_0}{p_1}\right)\right) = \tanh\left(\frac{1}{2}\mathrm{LLR}\right),$$

where $\mathrm{LLR} = \log(p_0/p_1)$ and $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$ is the hyperbolic tangent function.

- Applying this relation to (1) gives

$$\tanh\left(\frac{1}{2}L(c_0|r,\mathrm{SPC})\right) = \prod_{l=1}^{d-1}\tanh\left(\frac{1}{2}L(c_l|r_l)\right)$$

or

$$L(c_0|r,\mathrm{SPC}) = 2\tanh^{-1}\left(\prod_{l=1}^{d-1}\tanh\left(\frac{1}{2}L(c_l|r_l)\right)\right).$$

- The MAP decoder for bit c_0 in a length-d SPC code makes the decision $\widehat{c_0} = 0$ if $L(c_0|r, SPC) \geq 0$ and $\widehat{c_0} = 1$ otherwise.
- In the context of LDPC decoding, when the CNs function as APP processors instead of MAP decoders, CN i computes the extrinsic information

$$L_{i\to j} = 2\tanh^{-1}\left(\prod_{j'\in N(i)-\{j\}} \tanh\left(\frac{1}{2}L_{j'\to i}\right)\right)$$

and transmits it to VN j.

- Because the product is over the set $N(i) - \{j\}$, the message $L_{j \to i}$ has in effect been subtracted out to obtain the extrinsic information $L_{i \to j}$.

- The Gallager SPA Decoder
  - The information $L_{j\to i}$ that VN j sends to CN i at each iteration is the best (extrinsic) estimate of the value of v_j (the sign bit of $L_{j\to i}$) and the confidence or reliability level of that estimate (the magnitude of $L_{j\to i}$).
  - This information is based on the REP constraint for VN j and all inputs from the neighbors of VN j, excluding CN i.

- Similarly, the information $L_{i\to j}$ that CN i sends to VN j at each iteration is the best (extrinsic) estimate of the value of v_j (sign bit of $L_{i\to j}$) and the confidence or reliability level of that estimate (magnitude of $L_{i\to j}$).
- This information is based on the SPC constraint for CN i and all inputs from the neighbors of CN i , excluding VN j.

- The decoder is initialized by setting all VN messages equal to

$$L_j = L(v_j | y_j) = \log\left(\frac{\Pr(v_j = 0 | y_j)}{\Pr(v_j = 1 | y_j)}\right)$$
$$= \log\left(\frac{\Pr(y_j | v_j = 0)}{\Pr(y_j | v_j = 1)}\right)$$

- As mentioned, the SPA assumes that the messages passed are statistically independent throughout the decoding process.

- When the y_j are independent, this independence assumption would hold true if the Tanner graph possessed no cycles. The SPA would yield exact LLRs in this case.
- For a graph of girth γ, the independence assumption is true only up to the $(\gamma/2)$th iteration, after which messages start to loop back on themselves in the graph's various cycles.

$L(v_j|y_j)$ for Binary Symmetric Channel

- In this case, $y_j \in \{0,1\}$ and we define

  $\varepsilon = \Pr(y_j = b^c|v_j = b)$ to be the error probability.

  Then it is obvious that

$$\Pr(v_j = b|y_j) = \begin{cases} 1-\varepsilon & \text{when } y_j = b, \\ \varepsilon & \text{when } y_j = b^c \end{cases}$$

- $L(v_j|y_j) = (-1)^{y_j}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$.

- $L(v_j|y_j)$ for Additive White Gaussian Noise Channel
  - We only consider binary-input additive white Gaussian noise channel (BI-AWGNC)
  - We first let $x_j = (-1)^{v_j}$ be the jth transmitted binary value
  - Note $x_j = +1(-1)$ when $v_j = 0(1)$. We shall use x_j and v_j interchangeably hereafter

- The jth received sample is $y_j = x_j + n_j$, where the n_j are independent and normally distributed as $N(0, \sigma^2)$. Then it is easy to show that

$$\Pr\left(y_j \middle| x_j = x\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_j - x)^2}{2\sigma^2}\right),$$

where $x \in \{-1,1\}$ and, from this, that

$$L\left(v_j | y_j\right) = 2y_j/\sigma^2.$$

- In practice, an estimate of σ^2 is necessary.

❑ The Gallager Sum-Product Algorithm

- 1.**Initialization** : For all j, initialize L_j for appropriate channel model. Then, for all i, j for which $h_{ij} = 1$, set $L_{j\to i} = L_j$.

- 2.**CN update** : Compute outgoing CN messages $L_{i\to j}$ for each CN using

$$L_{i\to j} = 2\tanh^{-1}\left(\prod_{j'\in N(i)-\{j\}} \tanh\left(\frac{1}{2}\mathrm{L}_{j'\to i}\right)\right)$$

and then transmit to the VNs.

- 3.**VN update** : Compute outgoing VN messages $L_{j\to i}$ for each VN   using

$$L_{j\to i} = L_j + \sum_{i' \in N(j)-\{i\}} L_{i'\to j}$$

and then transmit to the CNs.

- 4.**LLR total** : For $j = 0,1,2,\ldots,n-1$ compute

$$L_j^{total} = L_j + \sum_{i \in N(j)} L_{i\to j}$$

- 5.**Stopping criteria** : For $j = 0,1,2,\ldots,n-1$, set

$$\hat{v}_j = \begin{cases} 1 & if\ L_j^{total} < 0, \\ 0 & else \end{cases}$$

to obtain $\hat{v}$. If $\hat{v}\boldsymbol{H}^T = 0$ or the number of iteration equals the maximum limit, stop; else, go to Step 2.

- Reduction on tanh and $\tanh^{-1}$ Functions
  - The update equation (2) is numerically challenging due to the presence of the product and the tanh and $\tanh^{-1}$ functions.
  - Following Gallager, we can improve the situation as follows. First, factor $L_{j\to i}$ into its sign and magnitude (or bit value and bit reliability):

$$L_{j\to i} = \alpha_{ji}\beta_{ji},$$

$$\alpha_{ji} = sign(L_{j\to i}),$$

$$\beta_{ji} = \left|L_{j\to i}\right|,$$

such that

$$\prod_{j'\in N(i)-\{j\}} \tanh\left(\frac{1}{2}L_{i\to j}\right) = \prod_{j'\in N(i)-\{j\}} \alpha_{j'i} \prod_{j'\in N(i)-\{j\}} \tanh\left(\frac{1}{2}\beta_{j'i}\right)$$

- We then have

$$
\begin{aligned}
L_{i\to j} &= \prod_{j'\in N(i)-\{j\}} \alpha_{j'i} \cdot 2tanh^{-1}\left(\prod_{j'\in N(i)-\{j\}} tanh\left(\frac{1}{2}\beta_{j'i}\right)\right) \\
&= \prod_{j'\in N(i)-\{j\}} \alpha_{j'i} \cdot 2tanh^{-1}log^{-1}log\left(\prod_{j'\in N(i)-\{j\}} tanh\left(\frac{1}{2}\beta_{j'i}\right)\right) \\
&= \prod_{j'\in N(i)-\{j\}} \alpha_{j'i} \cdot 2tanh^{-1}log^{-1}\left(\sum_{j'\in N(i)-\{j\}} log\left(tanh\left(\frac{1}{2}\beta_{j'i}\right)\right)\right)
\end{aligned}
$$

- CN update :

$$L_{i\to j} = \prod_{j'\in N(i)-\{j\}} \alpha_{j'i} \cdot \phi^{-1}\left(\sum_{j'\in N(i)-\{j\}} \phi\left(\beta_{j'i}\right)\right)$$

where we have defined

$$\phi(x) = -\log\left[\tanh\left(\frac{x}{2}\right)\right] = log\left(\frac{e^x+1}{e^x-1}\right)$$

and used the fact that $\phi^{-1}(x) = \phi(x)$ when $x > 0$.