Low-Density Parity-Check Codes [1]

Yunghsiang S. Han

Department of Electrical Engineering,
National Taiwan University of Science and Technology
Taiwan

E-mail: yshan@mail.ntust.edu.tw

Description of Low-Density Parity-Check Codes

- Low-density parity-check (LDPC) codes are a class of linear block codes which provide near-capacity performance on many channels.
- They were invented by Gallager in his 1960 doctoral dissertation.
- The study of LDPC codes was resurrected in the mid 1990s with the work of MacKay, Luby, and others.
- We only consider binary LDPC codes in this lecture.

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.
- 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

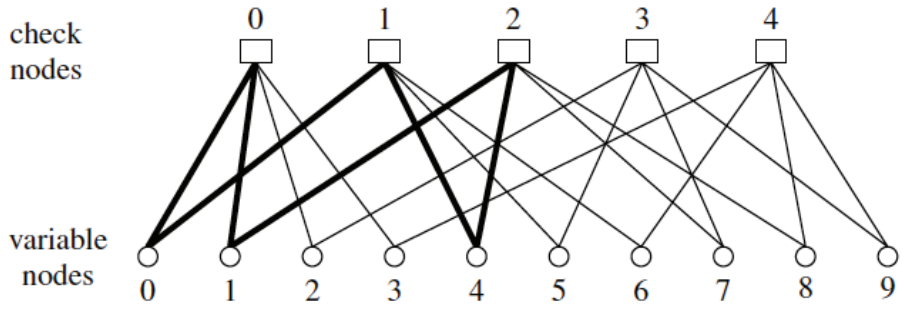
- The *Tanner graph* of an LDPC code is analogous to the trellis of a convolutional code.
- A Tanner graph is a *bipartite graph* whose nodes may be separated into two types, with edges connecting only nodes of different types.
- The two types in a Tanner graph are the *variable nodes* (VNs) (or *code-bit 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:

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



Graphical Representation

- 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 $mr = ng$.

Classifications of LDPC codes

- 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 an $n \times n$ 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.,

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{A}_{11} & \cdots & \boldsymbol{A}_{1N} \\ \vdots & & \vdots \\ \boldsymbol{A}_{M1} & \cdots & \boldsymbol{A}_{MN} \end{bmatrix},$$

where each matrix $\boldsymbol{A}_{rc}$ is a $Q \times Q$ circulant.

- To effect irregularity, some of the circulants may be the all-zeros $Q \times Q$ matrix using a technique called masking.
- In addition to partitioning LDPC codes into three classes – 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.

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

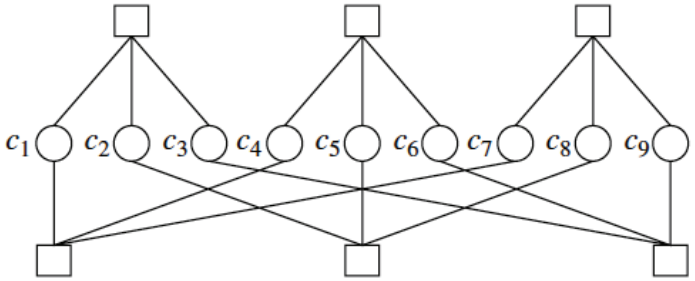
(a)

c_1	c_2	c_3
c_4	c_5	c_6
c_7	c_8	c_9

(b)

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

(c)



Row checks (row decoder)

Codeword (received word)

Column checks (column decoder)

(d)

e	e	1
1	e	0
0	e	e

Received word

e	e	1
1	1	0
0	e	e

After row decoding

1	e	1
1	1	0
0	e	1

After column decoding

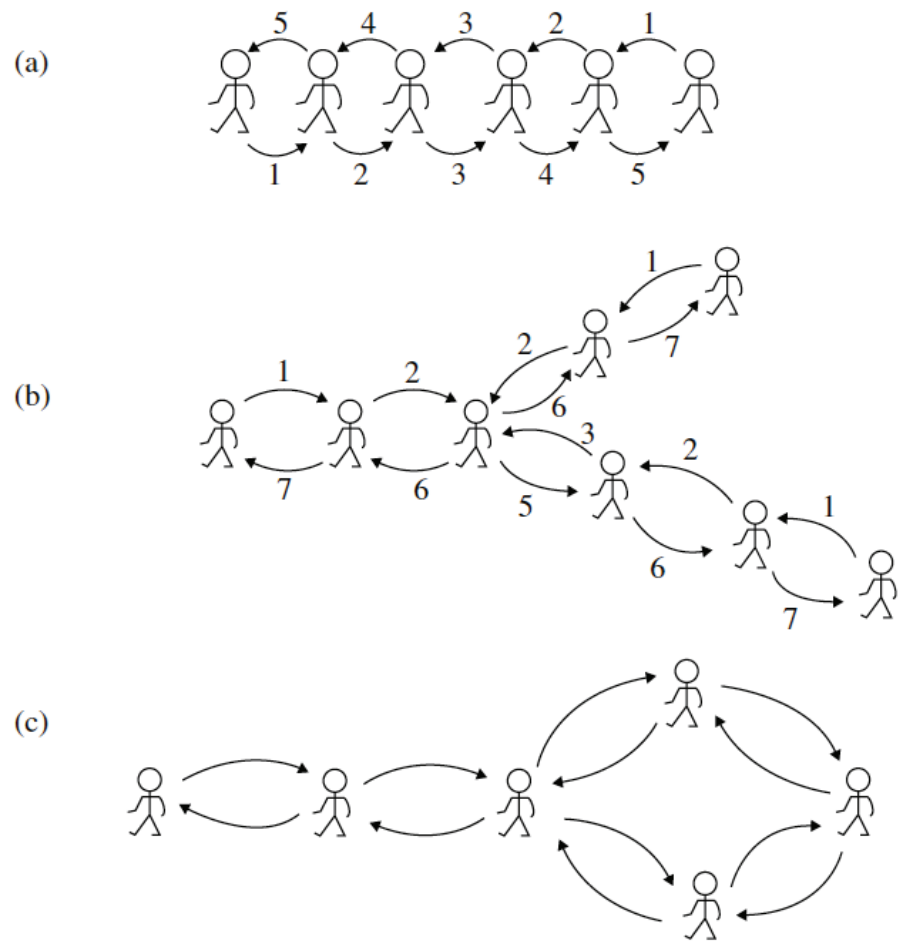
1	0	1
1	1	0
0	1	1

After row decoding

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

Message Passing and the Turbo Principle

- 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.

Message Passing and the Turbo Principle

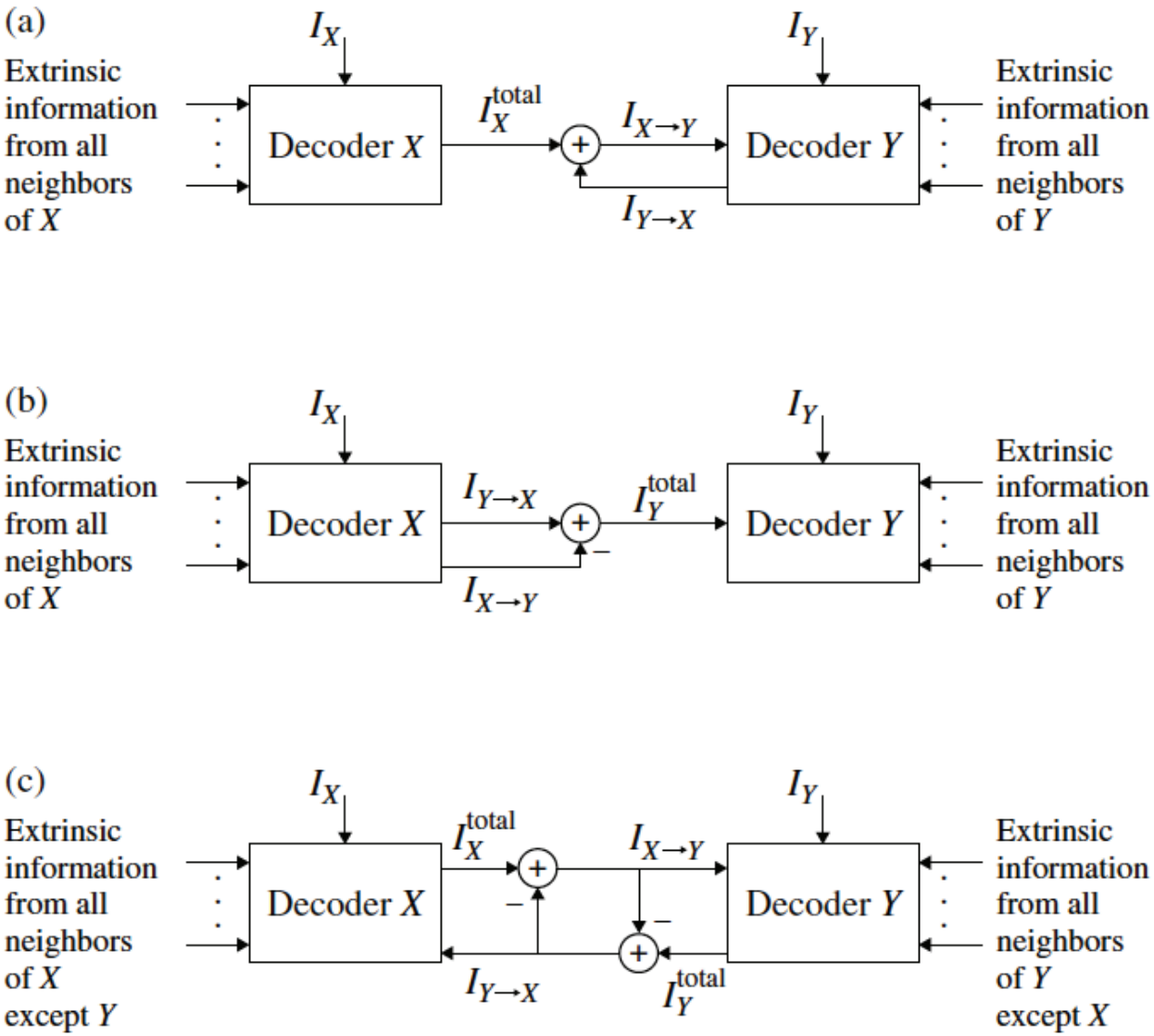
- Consider (b). The message that an arbitrary soldier X passes to arbitrary neighboring soldier Y is equal to the sum of all incoming messages, plus one for soldier X, minus the message that soldier Y had just sent to soldier X.
- This message-passing rule introduces the concept of *extrinsic information*.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

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

$$= \sum_{Z \in N(X) - \{Y\}} I_{Z \to X} + I_X,$$

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 untenable.
- 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)
Extrinsic information from all neighbors of X
I_X
Decoder X
I_X^{total}
$I_{X\to Y}$
$I_{Y\to X}$
I_Y
Decoder Y
Extrinsic information from all neighbors of Y
(b)
Extrinsic information from all neighbors of X
I_X
Decoder X
$I_{Y\to X}$
$I_{X\to Y}$
I_Y^{total}
I_Y
Decoder Y
Extrinsic information from all neighbors of Y
(c)
Extrinsic information from all neighbors of X except Y
I_X
Decoder X
I_X^{total}
$I_{Y\to X}$
$I_{X\to Y}$
I_Y^{total}
I_Y
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 $\boldsymbol{v} = (v_0, v_1, \ldots, v_{n1})$ equals 1, given the received word $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n1})$.
- Without loss of generality, we focus on the decoding of bit v_j and calculate $\Pr(v_j|\boldsymbol{y})$.

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

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

  and

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

  respectively.

- The natural logarithm is assumed for LLRs.

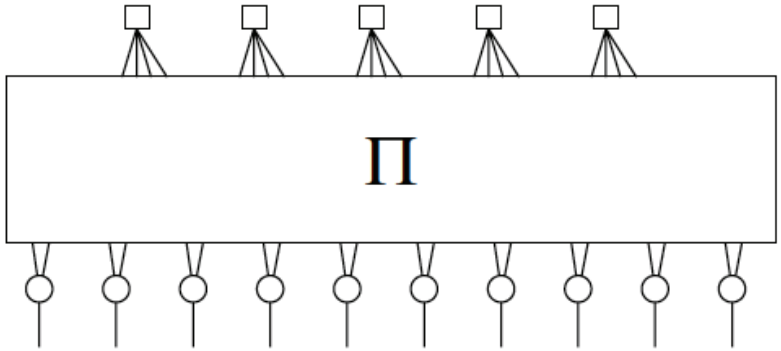
- The SPA for the computation of $\Pr(v_j = 1|\boldsymbol{y}), \ell(v_j|\boldsymbol{y})$, or $L(v_j|\boldsymbol{y})$ is a distributed algorithm that is an application of the turbo principle to a code's Tanner graph.

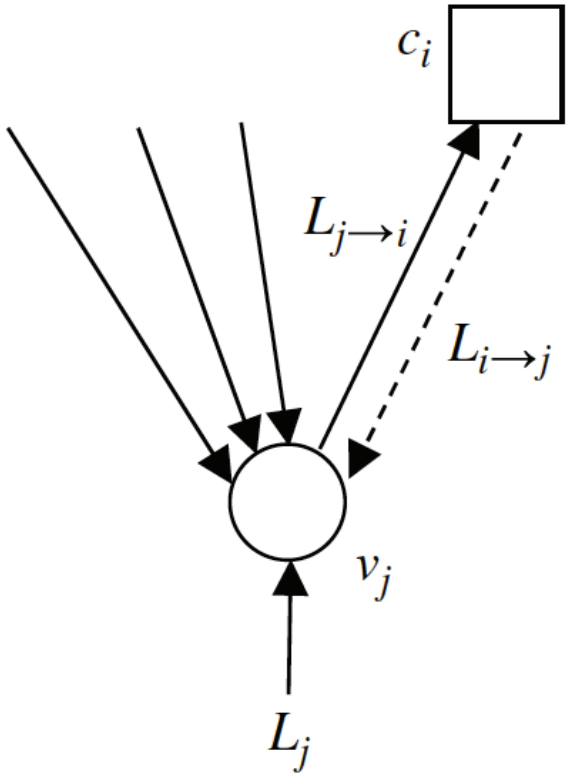
- An LDPC code can be deemed a collection of SPC codes concatenated through an interleaver to a collection of repetition (REP) codes.

- The SPC codes are treated as outer codes, that is, they are not

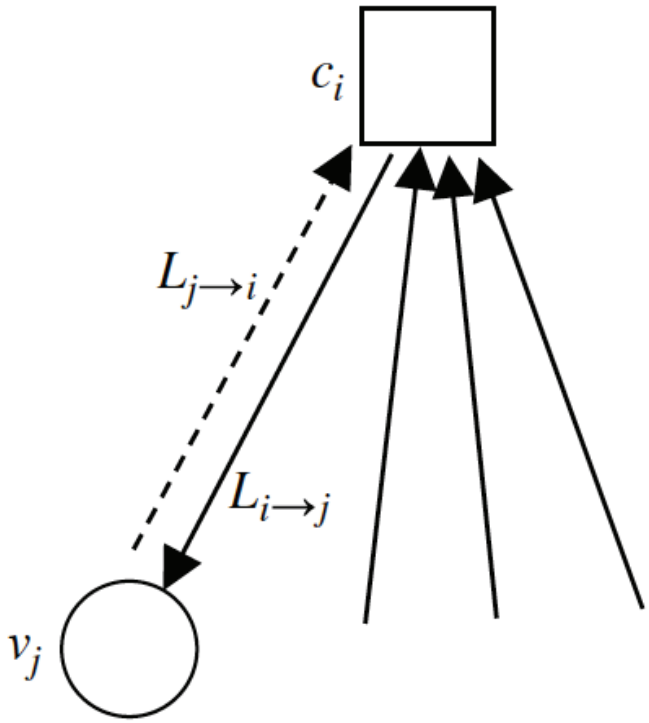
connected to the channel.

- The following is a graphical representation of an LDPC code as a concatenation of SPC and REP codes. "Π" represents an interleaver





- The figure depicts the REP (VN) decoder situation for a VN with degree greater than the degree-2 VNs of that in previous slide.
- 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 $L(v_j|\boldsymbol{y})$ for $j = 0,\ 1,\ \ldots,\ n-1$.
- 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 $L(v_j|\boldsymbol{y})$ 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 $\boldsymbol{r}$ is received.
- The MAP decoder computes the log-APP ratio

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

which is equal to

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

under an equally likely assumption for the value of c.

- This simplifies as

$$\begin{aligned} L(c|\boldsymbol{r}) &= \log\left(\frac{\prod_{\ell=0}^{d-1}\Pr(r_\ell|c=0)}{\prod_{\ell=0}^{d-1}\Pr(r_\ell|c=1)}\right) \\ &= \sum_{\ell=0}^{d-1}\log\left(\frac{\Pr(r_\ell|c=0)}{\Pr(r_\ell|c=1)}\right) \\ &= \sum_{\ell=0}^{d-1}L(r_\ell|c) = \sum_{\ell=0}^{d-1}L(c|r_\ell), \end{aligned}$$

  where $L(r_\ell|c)$ and $L(c|r_\ell)$ 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 $L(c|\boldsymbol{r})\geq 0$ 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,

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

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

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

- 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 $\boldsymbol{a} = (a_0, a_1, \ldots, a_{d-1})$ in which $\Pr(a_\ell = 1) = p_1^{(\ell)}$ and $\Pr(a_\ell = 0) = p_0^{(\ell)}$. Then the probability that $\boldsymbol{a}$ contains an even number of 1s is

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

and the probability that $\boldsymbol{a}$ contains an odd number of 1's is

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

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

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

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

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

- Consider the transmission of a length-d SPC codeword $\boldsymbol{c}$ over a memoryless channel whose output is $\boldsymbol{r}$.

- The bits c_ℓ in the codeword $\boldsymbol{c}$ have a single constraint: there must be an even number of 1s in $\boldsymbol{c}$. Without loss of generality,

we focus on bit c_0, for which the MAP decision rule is

$$\hat{c}_0 = \arg \max_{b \in \{0,1\}} \Pr(c_0 = b | \boldsymbol{r}, \text{SPC}),$$

where the conditioning on SPC is a reminder that there is an SPC constraint imposed on $\boldsymbol{c}$.

•

$$\begin{aligned} \Pr(c_0 = 0 | \boldsymbol{r}, \text{SPC}) &= \Pr(c_1, c_2, \ldots, c_{d-1} \text{ has an even no. of 1s} | \boldsymbol{r}) \\ &= \frac{1}{2} + \frac{1}{2} \prod_{\ell=1}^{d-1} (1 - 2 \Pr(c_\ell = 1 | r_\ell)) . \end{aligned}$$

• Rearranging gives

$$1 - 2 \Pr(c_0 = 1 | \boldsymbol{r}, \text{SPC}) = \prod_{\ell=1}^{d-1} (1 - 2 \Pr(c_\ell = 1 | r_\ell)) , \qquad (1)$$

where we used $\Pr(c_0 = 0 | \boldsymbol{r}, \text{SPC}) = 1 - \Pr(c_0 = 1 | \boldsymbol{r}, \text{SPC})$.

- 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

$$\tanh\left(\frac{1}{2}L(c_0|\boldsymbol{r}, \mathrm{SPC})\right) = \prod_{\ell=1}^{d-1} \tanh\left(\frac{1}{2}L(c_\ell|r_\ell)\right)$$

  or

$$L(c_0|\boldsymbol{r}, \mathrm{SPC}) = 2\tanh^{-1}\left(\prod_{\ell=1}^{d-1} \tanh\left(\frac{1}{2}L(c_\ell|r_\ell)\right)\right).$$

- The MAP decoder for bit c_0 in a length-d SPC code makes the decision $\hat{c}_0 = 0$ if $L(c_0|\boldsymbol{r}, \text{SPC}) \geq 0$ and $\hat{c}_0 = 1$ otherwise.
- In the context of LDPC decoding, when the CNs function as APP processors instead of MAP decoders, CN i computes the extrinsic information

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

  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_i (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

$L_{j\to i}$ equal to

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

for all j, i for which $h_{ij} = 1$.

- 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 $\mathcal{N}(0, \sigma^2)$. Then it is easy to show that

$$\Pr(x_j = x|y_j) = [1 + \exp(-2y_j x/\sigma^2)]^{-1},$$

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

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

- 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

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

   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

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

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

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

to obtain $\hat{\boldsymbol{v}}$. If $\hat{\boldsymbol{v}}\boldsymbol{H}^T = \mathbf{0}$ 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):

$$\begin{aligned} L_{j\to i} &= \alpha_{ji}\beta_{ji}, \\ \alpha_{ji} &= \text{sign}(L_{j\to i}), \\ \beta_{ji} &= |L_{j\to i}|, \end{aligned}$$

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

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

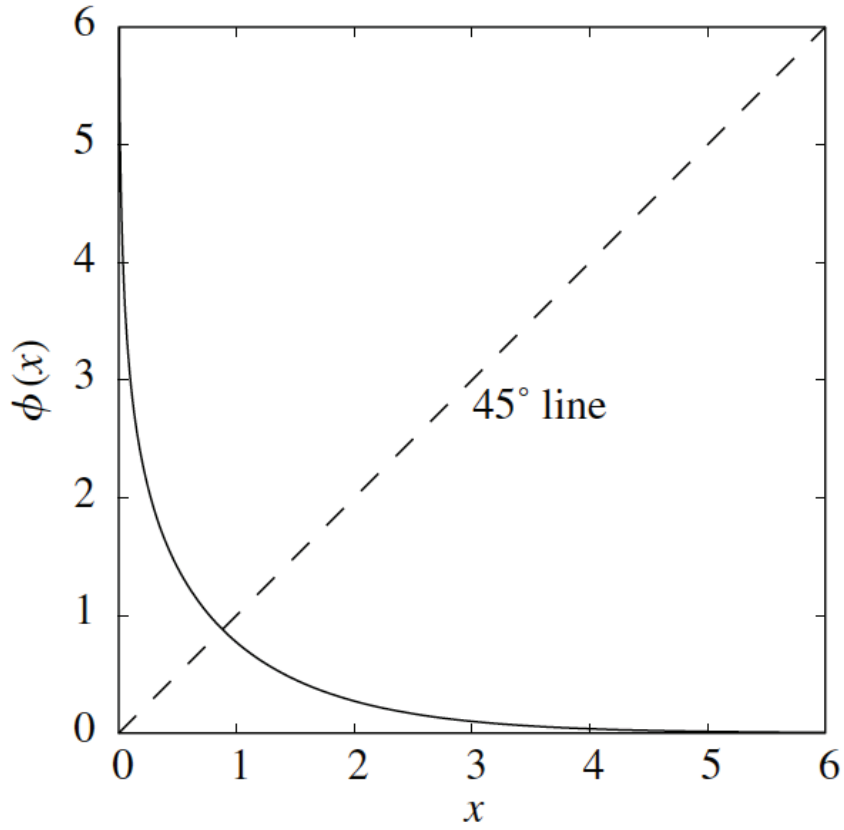
- CN update: $L_{i\to j} = \prod_{j'\in N(i)-\{j\}} \alpha_{j'i} \cdot \phi\left(\sum_{j'\in N(i)-\{j\}} \phi(\beta_{j'i})\right)$,

  where we have defined

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

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

- The function $\phi(x)$, shown in the figure, may be implemented by use of a look-up table.



Performance of the SPA Decoder

- In contrast with the error-rate curves for classical codes – e.g., Reed-Solomon codes with an algebraic decoder or convolutional codes with a Viterbi decoder – the error-rate curves for iteratively decoded codes generally have a region in which the slope decreases as the channel signal to noise ratio (SNR) increases.
- The region of the curve just before the slope transition region is called the *waterfall region* of the error-rate curve and the region of the curve with the reduced slope is called the *error-rate floor region*, or simply the *floor region*.
- A floor seen in the performance curve of an iteratively decoded code is occasionally attributable to a small minimum distance, particularly when the code is a parallel turbo code.
- it is possible to have a floor in the error-rate curve of an LDPC

code (or serial turbo code) with a large minimum distance.

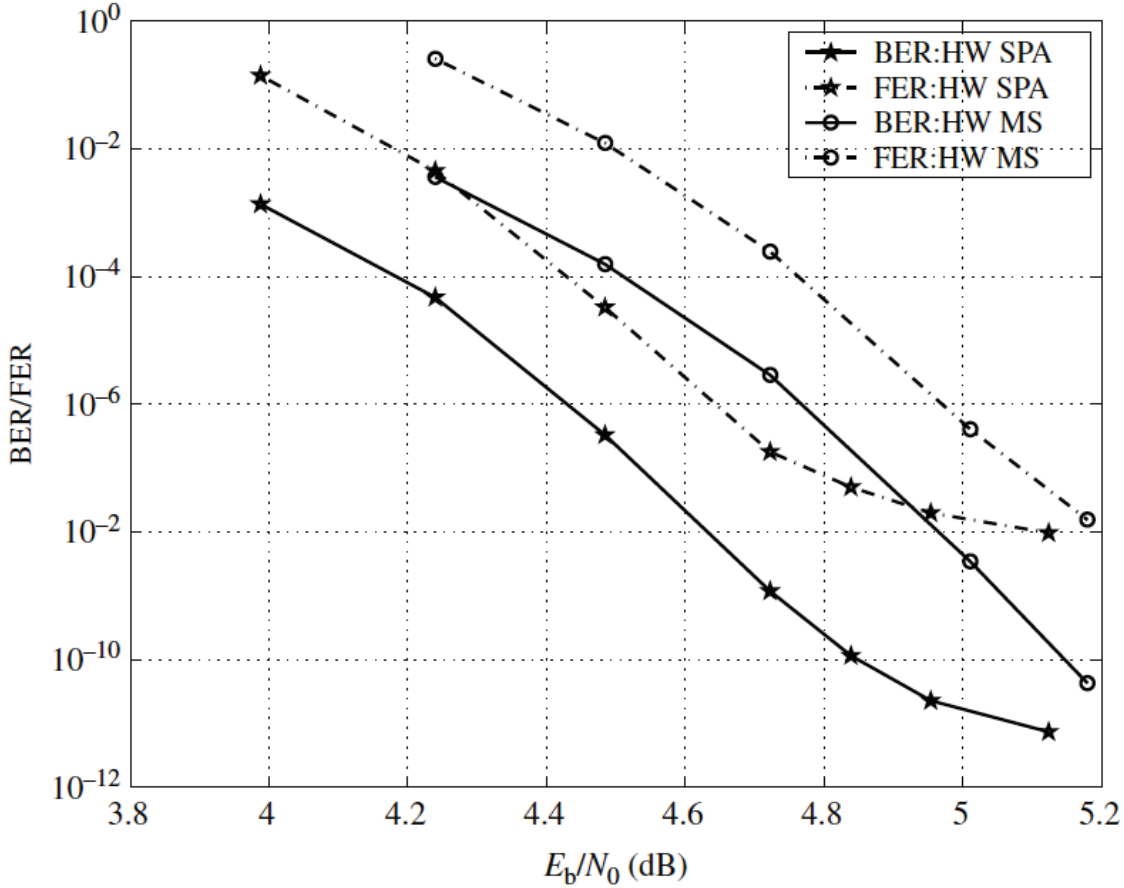
- In this case, the floor is attributable to so-called trapping sets.
- An (ω, ν) *trapping set* is a set of ω VNs that induce a subgraph with ν odd-degree checks so that, when the ω bits are all in error, there will be ν failed parity checks.
- An essentially equivalent notion is a near-codeword.
- An (ω, ν) *near-codeword* is a length-n error pattern of weight ω that results in ν check failures (i.e., the syndrome weight is ν), where ω and ν are "small."
- Near-codewords tend to lead to error situations from which the SPA decoder (and its approximations) cannot escape.
- The implication of a small ω is that the error pattern is more likely.
- The implication of a small ν is that only a few check equations

are affected by the pattern, making it more likely to escape the notice of the iterative decoder.

- Iterative decoders are susceptible to trapping sets (near-codewords) since an iterative decoder works locally in a distributed-processing fashion, unlike an ML decoder, which finds the globally optimum solution.

Performance of a $0.9(4550, 4096)$ Quasi-Cyclic Irregular Repeat-Accumulate (IRA) Code with SPA and Min-Sum Decoders

"HW" represents "hardware."



The Min-Sum Decoders

- Note from the shape of $\phi(x)$ that the largest term in the sum corresponds to the smallest β_{ji} so that, assuming that this term dominates the sum,

$$\begin{aligned}\phi\left(\sum_{j'\in N(i)-\{j\}}\phi(\beta_{j'i})\right) &\simeq \phi\left(\phi\left(\min_{j'\in N(i)-\{j\}}\beta_{j'i}\right)\right)\\ &= \min_{j'\in N(i)-\{j\}}\beta_{j'i}.\end{aligned}$$

- Thus, the min-sum algorithm is simply the log-domain SPA with update equation replaced by

$$\text{CN update: } L_{i\to j} = \prod_{j'\in N(i)-\{j\}}\alpha_{j'i}\cdot\min_{j'\in N(i)-\{j\}}\beta_{j'i}.$$

- It can also be shown that, in the AWGNC case, the initialization $L_{j\to i} = 2y_j/\sigma^2$ may be replaced by $L_{j\to i} = y_j$

when the min-sum algorithm is employed.

- We observe that, while the SPA decoder is superior in the waterfall region by about 0.3 dB, it suffers from an error-rate floor.
- This floor is attributable to trapping sets seen by the SPA decoder which are apparently transparent to the min-sum decoder.
- In fact the floor is a characteristic of iteratively decodable codes and was first seen with turbo codes.
- If one sees an iterative-decoder error-rate curve without a floor, then, in all likelihood, the simulations have not been run at a high enough SNR.
- The floor likely exists, but it requires more time and effort to find it, since it might be out of the reach of standard computer simulations.

- The floor can be due to trapping sets (as is usually the case for LDPC codes and serial turbo codes) or to a small minimum distance (as is usually the case for parallel turbo codes).

The Bit-Flipping Algorithm for the BSC

- The bit-flipping algorithm first evaluates all parity-check equations in $\boldsymbol{H}$ and then "flips" (complements) any bits in the received word that are involved in more than some fixed number of failed parity checks.
- This step is then repeated with the modified received word until all parity checks are satisfied or until some maximum number of iterations has been executed.
- Noting that failed parity-check equations are indicated by the elements of the syndrome, $\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^T$.
- The number of failed parity checks for each code bit is contained in the n-vector $\boldsymbol{f} = \boldsymbol{s}\boldsymbol{H}$ (operations over the integers).

The Bit-Flipping Algorithm

1. Compute $\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^T$ (operations over $GF(2)$). If $\boldsymbol{s} = 0$, stop, since $\boldsymbol{r}$ is a codeword.
2. Compute $\boldsymbol{f} = \boldsymbol{s}\boldsymbol{H}$ (operations over $\mathbb{Z}$).
3. Identify the elements of $\boldsymbol{f}$ greater than some preset threshold, and then flip all the bits in $\boldsymbol{r}$ corresponding to those elements.
4. If you have not reached the maximum number of iterations, go to Step 1 using the updated $\boldsymbol{r}$.

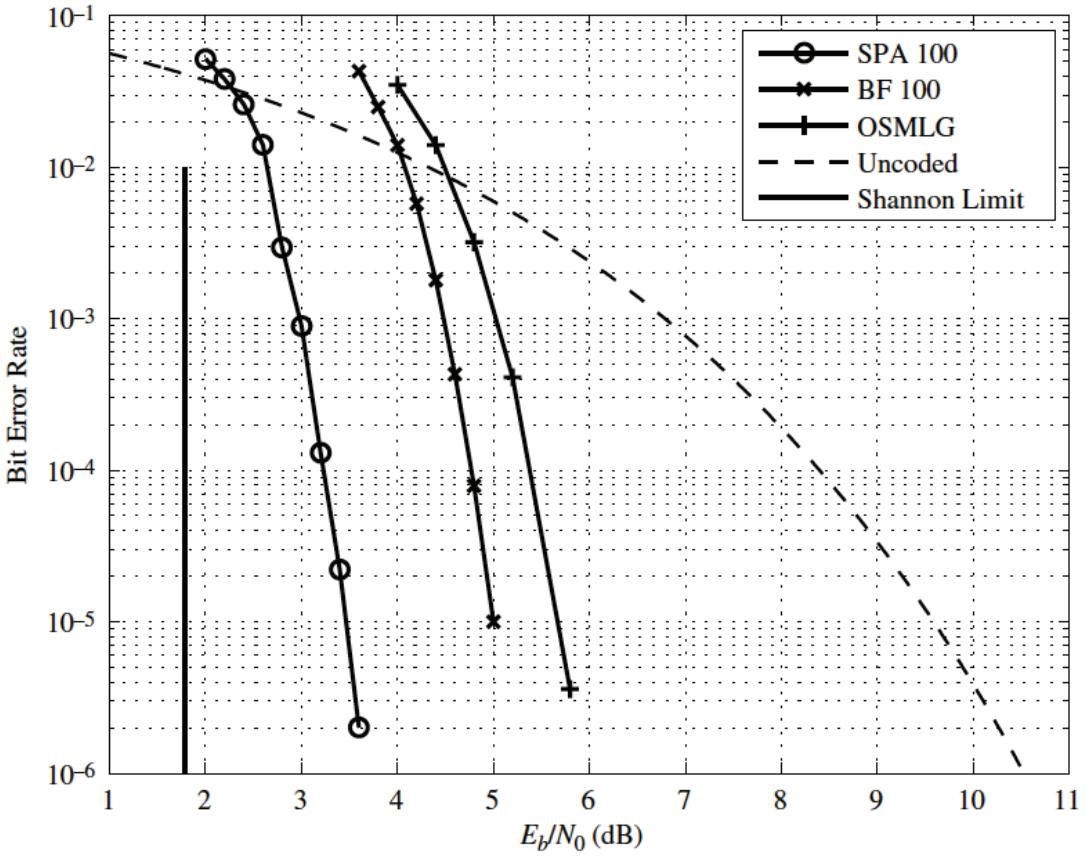
- The threshold will depend on the channel conditions and Gallager has derived the optimum threshold for regular LDPC codes.
- A particularly convenient way to obtain a threshold that effectively adapts to the channel quality:

  $3'$. Identify the elements of $\boldsymbol{f}$ equal to $\max f_j$ and then flip all

the bits in $\boldsymbol{r}$ corresponding to those elements.

The Performance of Bit-Flipping (BF) Algorithm

- The code: $0.823(4161, 3431)$ cyclic PG-LDPC code
- OSMLG: one-step majority-logic decoding algorithm



References

[1] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*, New York , NY: Cambridge University Press, 2009.