Factor Graph and Low Density Parity Check Codes.

Laura Cottatellucci

EURECOM

laura.cottatellucci@eurecom.fr

Outlines

I. Factor Graph

- **Motivations**
- **Definition**
- **Tanner Graphs**
- **Efficient Determination of Marginals from p.m.f.**
- **Message Passing Algorithm**
- **Bitwise MAP Decoding via Message Passing**
- **Blockwise MAP Decoding via Message Passing**

II. Low Density Parity Check Codes

- **Definition and Notation**
- **Message Passing Decoder**
- **Performance Analysis**

III. References

Why to Study Factor Graph for Channel Coding?

A bottleneck in the coding problem is the complexity of the decoder.

Let $v \in \mathcal{C}$ be a codeword in $\mathcal{C}$ and let r be the received signal.

The bitwise MAP decoder is defined by

$$\widehat{v}_i^{\text{MAP}} = \underset{v_i \in \{\pm 1\}}{\text{argmax}}\, p_{V_i|\boldsymbol{R}}(v_i|\boldsymbol{r})$$

The wordwise MAP decoder reads

$$\widehat{\boldsymbol{v}}^{\text{MAP}} = \underset{\boldsymbol{v} \in \mathcal{C}}{\text{argmax}}\, p_{\boldsymbol{V}|\boldsymbol{R}}(\boldsymbol{v}|\boldsymbol{r})$$

In both cases the problem can be reduced to determine marginals of probability mass functions (p.m.f.) $p(r_i|\boldsymbol{v})$

Factor graphs provide an efficient approach to determine marginals of p.m.f.

Distributive Law and Complexity

$\forall a, b, c \in \mathbb{F}$ **(field)**

$$ac + bc = (a + b)c,$$

l.h.s. : 1 addition and 2 multiplications — r.h.s. 1 addition and 1 multiplication

$$\sum_{i,j=1}^{N} a_i b_j = (\sum_{i}^{N} a_i)(\sum_{j=1}^{N} b_j),$$

l.h.s. : $(N^2 - 1)$ **additions and** N^2 **multiplications — r.h.s.** $2N - 2$ **additions and** 1 **multiplication**

The application of the distributive law can decrease complexity considerably

Factor graphs provide a framework to take advantage systematically of the distributive law.

Marginals and Distributive Law: Example

Let $f(x_1, x_2, x_3, x_4, x_5, x_6) = f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$ **with** $x_1, x_2, \ldots x_6$ **variables over a finite alphabet** $\mathcal{X}$ **and let** $f(x_1)$, **the marginal of** $f(x_1, \ldots, x_6)$ **w.r.t.** x_1 **be**

$$f(x_1) = \sum_{x_2, x_3, x_4, x_5, x_6} f(x_1, x_2, x_3, x_4, x_5, x_6) = \sum_{\sim x_1} f(x_1, x_2, x_3, x_4, x_5, x_6) \tag{1}$$

Determining $f(x_1)$ **by applying directly (1) has a complexity** $\Theta(|\mathcal{X}|^6)$.

Let us determine $f(x_1)$ **taking advantage of the factorization of** $f(x_1, x_2, x_3, x_4, x_5, x_6)$ **and the distributive law**

$$f(x_1) = \left[\sum_{x_2, x_3} f_1(x_1, x_2, x_3) \right] \left[\sum_{x_4, x_6} f_2(x_1, x_4, x_6) f_3(x_4) \sum_{x_5} f_4(x_4, x_5) \right]$$

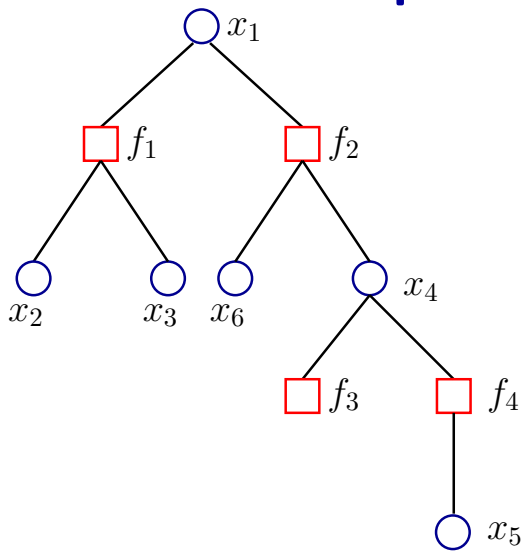
The complexity of this approach is $\Theta(|\mathcal{X}|^3)$ **and requires storage of intermediate values.**

Factorization and Factor Graphs

Let us consider $f(x_1, x_2, x_3, x_4, x_5, x_6) = f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$

- For each variable draw a variable node (circle)
- For each function draw a factor node (square)
- If a variable appears in a factor function draw an edge between the corresponding variable node and factor node.

Factor Graph



A factor graph is a bipartite graph, i.e. the set of vertices is partitioned into two groups (the group of variable nodes and the group of factor nodes) and the edges always connect two nodes belonging to different groups

Factor Graph for Block Codes: Tanner Graph

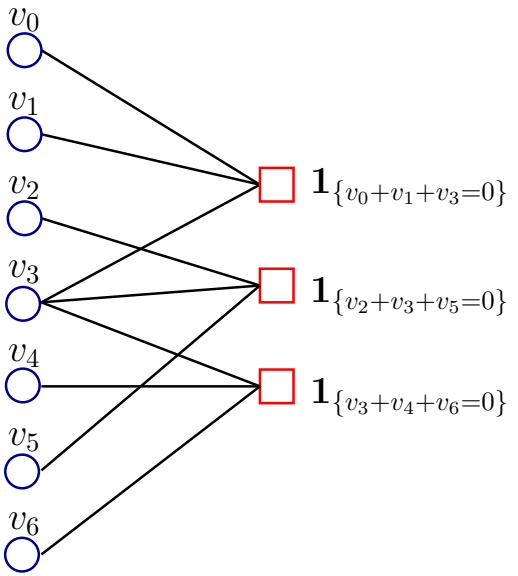
The parity check matrix $\boldsymbol{H}$ of a block code enforces a constraint $f(v_0, v_1, \ldots v_{N-1})$ on the codeword symbols

$$f(\boldsymbol{v}) = \begin{cases} 1 & \boldsymbol{v}\boldsymbol{H}^T = \mathbf{0}, \\ 0 & \text{otherwise.} \end{cases}$$

$f(\boldsymbol{v})$ can be factorized in product of indicator functions* $\mathbf{1}_{\{\boldsymbol{x}\in\mathcal{A}\}}$

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

$$f(\boldsymbol{v}) = \mathbf{1}_{\{v_0+v_1+v_3=0\}}\mathbf{1}_{\{v_2+v_3+v_5=0\}}\mathbf{1}_{\{v_3+v_4+v_6=0\}}$$



The factor graph of a block code is called Tanner graph

* The indicator function $\mathbf{1}_{\{\boldsymbol{x}\in\mathcal{A}\}}$ is equal to 1 if $\boldsymbol{x} \in \mathcal{A}$ and equal to zero otherwise.

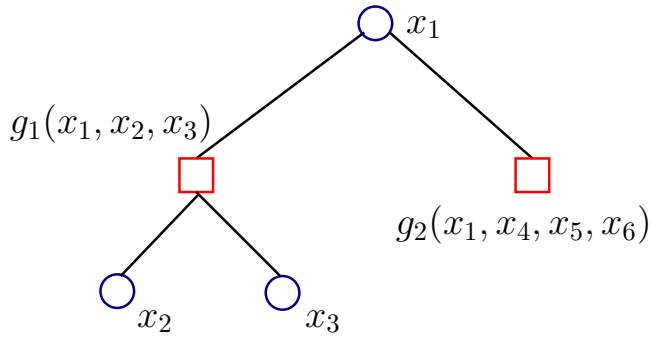
Recursive Determination of Marginals: An Example (1)

$$f(x_1) = \sum_{\sim x_1} f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$$

- Group the functions to get the finer partition of functions containing the same variables except x_1.
- f_1 is the only function containing x_2, x_3 and stays alone.
- f_2, f_3, f_4 are functions of x_4, x_5, x_6: group them.

Set

$$g_1(x_1, x_2, x_3) = f_1(x_1, x_2, x_3)$$
$$g_2(x_1, x_4, x_5, x_6) = f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$$

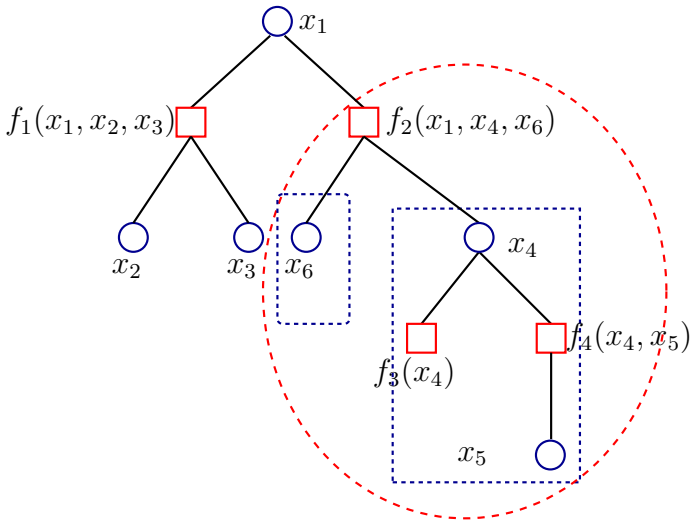


Recursive Determination of Marginals: An Example (2)

Thanks to the assumption that $f(x_1, x_2, x_3, x_4, x_5, x_6)$ can be decomposed in a tree, function $g_2(x_1, x_4, x_5, x_6)$ can be further decomposed in

- kernel function which is the only one depending on x_1.
- product of functions depending all on a common variable.

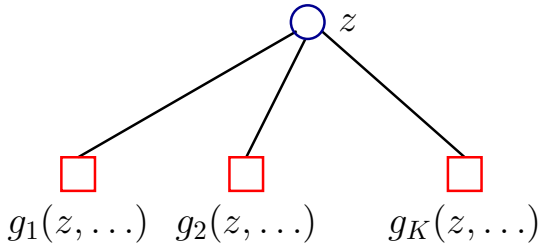
$$g_2(x_1, x_4, x_5, x_6) = \underbrace{f_2(x_1, x_4, x_6)}_{\text{kernel}} \underbrace{f_3(x_4) f_4(x_4, x_5)}_{x_4}$$



Recursive Determination of Marginals

Consider a generic function $g(z, z_1, \ldots, z_J)$ to be marginalized w.r.t. z and whose factor graph is a tree. Then

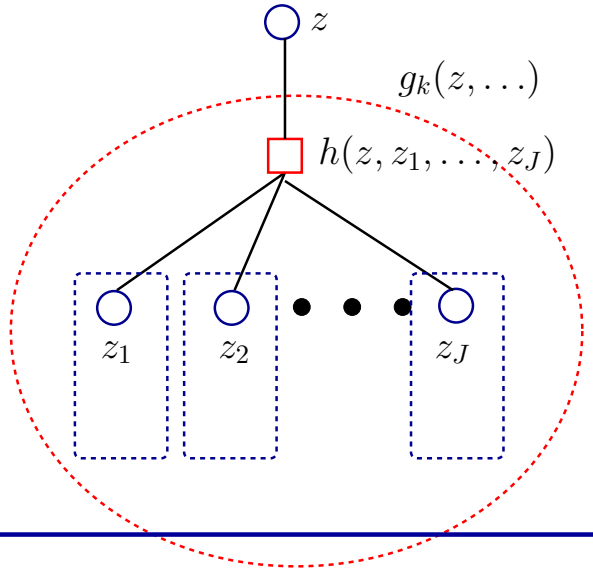
$$g(z, z_1, \ldots, z_K) = \prod_{k=1}^{K} [g_k(z, z_k, ...)]$$



Because of the tree structure

$$g_k(z, \ldots) = \underbrace{h(z, z_{k_1}, \ldots, z_{k_J})}_{\text{Kernel}} \prod_{j=1}^{J} h_j(z_{k_j}, \ldots)$$

where $h_j(z_{k_j}, \ldots)$ is a factor function independent of $z, z_{k_1}, \ldots, z_{k_{j-1}}, z_{k_{j+1}}, \ldots z_{k_J}$



Recursive Determination of Marginals (cntd)

To compute

$$\sum_{\sim z} g_k(z, z_1, \ldots z_K) = \sum_{\sim z} h(z, z_1, \ldots, z_K) \prod_{k=1}^{K} \sum_{\sim z_k} h_k(z_k, \ldots)$$

we need to compute

$$\sum_{\sim z_k} h_k(z_k, \ldots).$$

We can operate on $h_k(z_k, \ldots)$ as we operated on $g(z, \ldots)$ and we obtain a recursive structure.

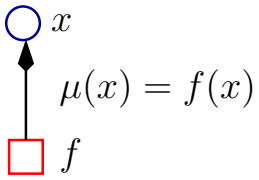
Message Passing Algorithm

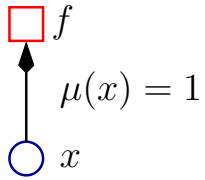
A message passing algorithm is based on factor graph and determine marginals

- **A message is a vector with $|\mathcal{X}|$ elements which are marginals or part of the function;**
- **The algorithm starts from the leave nodes;**
- **A leaf variable node sends a constant message 1. A factor node sends the values of the associated factor;**
- **As soon as a node has received messages from all its children it processes the received messages (multiplication in the variable nodes and saturation sums in the factor nodes) and sends its own message to its parent nodes.**

Message Passing Basic Rules

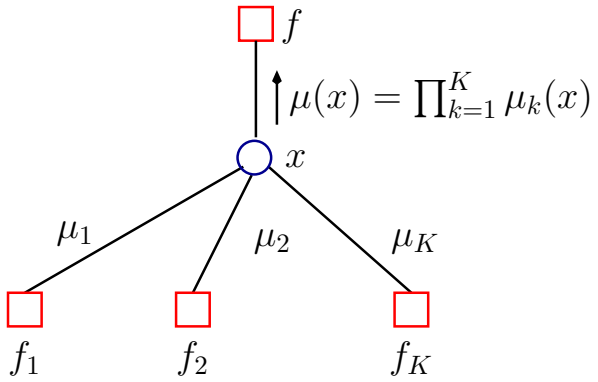
Initialization at Leaf Nodes

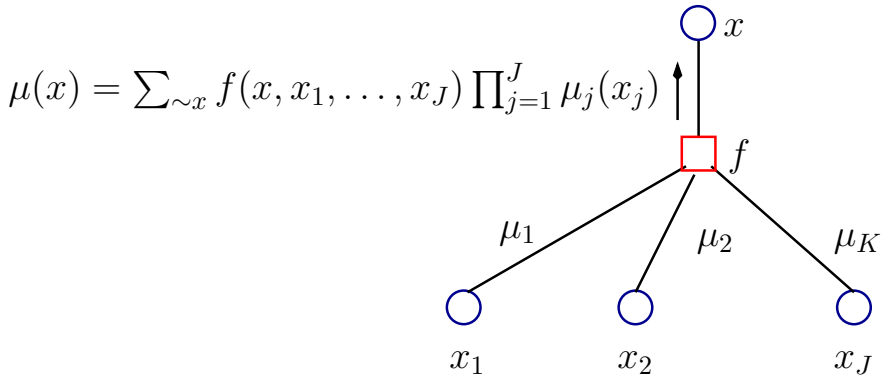




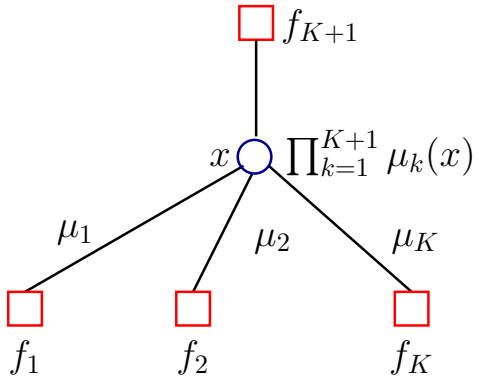
Message Passing Basic Rules

Variable/Factor Node Processing

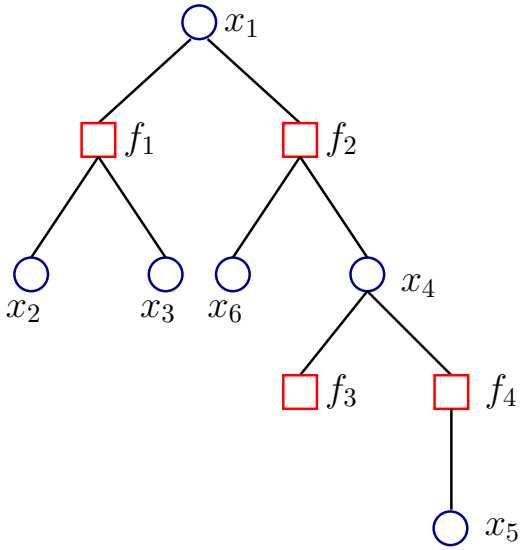


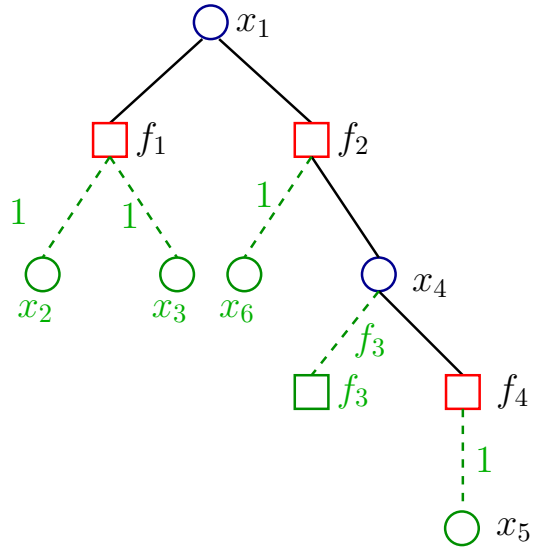


Marginalization

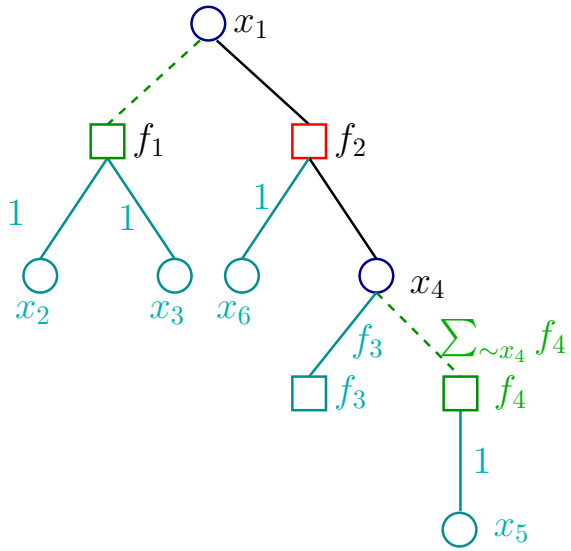


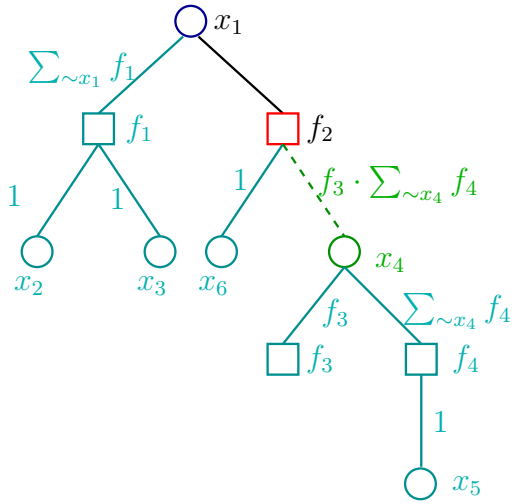
Example: $f(x_1) = \sum_{\sim x_1} f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$

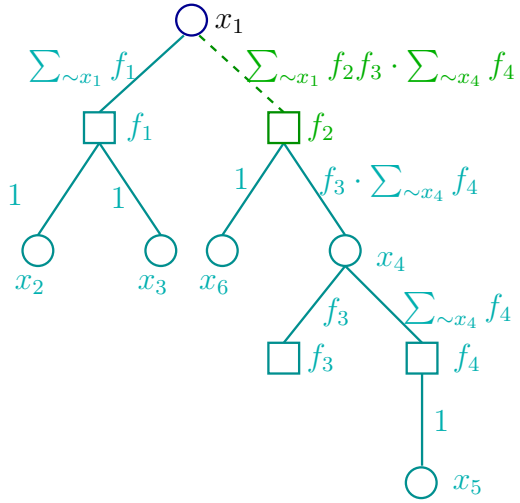




Example (cntd)







Remarks

If we need to compute all marginals we use the same graph

- **A node send a message to another node as soon as it has received messages from all the remaining nodes;**
- **The message passing algorithm ends when a message has been sent in both directions of all edges of the factor graph.**

Exercises

Draw the factor graph of the Markov chain

$$p_{XYZ}(x, y, z) = p_X(x)p_{Y|X}(y|x)p_{Z|Y}(z|y).$$

Compute the messages at each node to compute all marginals.

Consider a quantizer. Let $\mathcal{X}$ be the finite input alphabet and $\mathcal{Y}$ be the finite output alphabet. Let q be the quantization function $q : \mathcal{X} \to \mathcal{Y}$. Draw the corresponding factor graph. Starting from the general message passing rule and assuming that the incoming messages are $\mu_{xq}(x)$ and $\mu_{yq}(y)$, respectively what are the outgoing messages $\mu_{qx}(x)$ and $\mu_{qy}(y)$?

Exercises 2

Assume that the two binary symbols x and y are mapped by a function m into one $4-AM$ symbol, call it z, as in the following table

x	y	$z = m(x,y)$
0	**0**	**1**
1	**0**	**2**
0	**1**	**3**
1	**1**	**4 .**

In more detail, $m : X \times Y \to Z$.
Draw the corresponding factor graph. Starting from the general message passing rule and assuming that the incoming messages are $\mu_{x,m}(x), \mu_{y,m}(y)$, and $\mu_{z,m}(z)$ respectively, what are the outgoing messages $\mu_{m,x}(x), \mu_{m,y}(y)$, and $\mu_{m,z}(z)$?

Bitwise MAP Decoding via Message Passing

$$
\begin{aligned}
\widehat{v}_i^{\mathrm{MAP}}(\boldsymbol{r}) &= \underset{v_i \in \{\pm 1\}}{\operatorname{argmax}} p_{V_i|\boldsymbol{R}}(v_i|\boldsymbol{r}) \\
&= \underset{v_i \in \{\pm 1\}}{\operatorname{argmax}} \sum_{\sim v_i} p_{\boldsymbol{V}|\boldsymbol{R}}(\boldsymbol{v}|\boldsymbol{r}) \\
&= \underset{v_i \in \{\pm 1\}}{\operatorname{argmax}} \sum_{\sim v_i} p_{\boldsymbol{R}|\boldsymbol{V}}(\boldsymbol{r}|\boldsymbol{v}) p_{\boldsymbol{V}}(\boldsymbol{v}) \\
&= \underset{v_i \in \{\pm 1\}}{\operatorname{argmax}} \sum_{\sim v_i} \left(\prod_j p_{R_j|V_j}(r_j|v_j) \right) 1_{\{\boldsymbol{v} \in \mathcal{C}\}}
\end{aligned}
$$

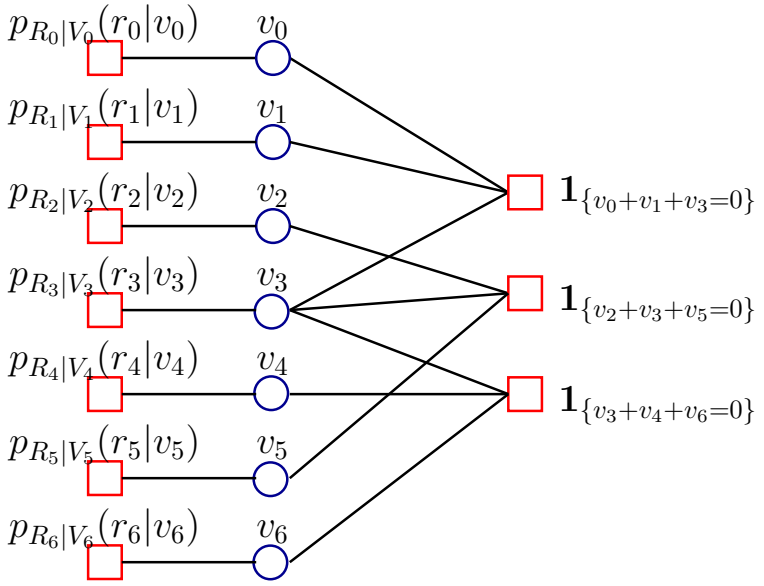
Since $1_{\{\boldsymbol{v} \in \mathcal{C}\}}$ can be factorized the bitwise MAP decoding algorithm is equivalent to determining the marginals of a factorized function and choosing the maximum: We can efficiently use the message passing algorithm.

Bitwise MAP Decoding via Message Passing: an Example

Parity Check Equations

$$v_0 + v_1 + v_3 = 0$$
$$v_2 + v_3 + v_5 = 0$$
$$v_3 + v_4 + v_6 = 0$$

$$\widehat{v}_i^{\text{MAP}} = \underset{v_i \in \{\pm 1\}}{\text{argmax}} \sum_{\sim v_i} \left(\prod_{j=0}^{6} p_{R_j|V_j}(r_j|v_j) \right) \mathbf{1}_{\{v_0+v_1+v_3=0\}} \mathbf{1}_{\{v_2+v_3+v_5=0\}} \mathbf{1}_{\{v_3+v_4+v_6=0\}}$$



Bitwise MAP Decoding for Binary Input Channels

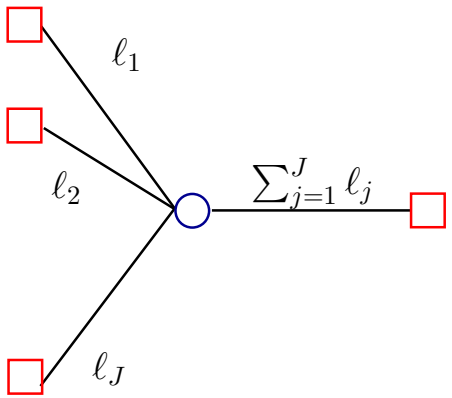
For a binary input channel a message is a vector of two elements $\mu(-1), \mu(1)$. It can be simplified if we consider log-likelihood ratios of the messages

$$\ell_k = \ln \rho_k = \ln \frac{\mu_k(1)}{\mu_k(-1)}$$

with the mapping $\theta : \{0,1\} \to \{\pm 1\},\ \theta(0) = 1, \theta(1) = -1.$

Message Passing Rules

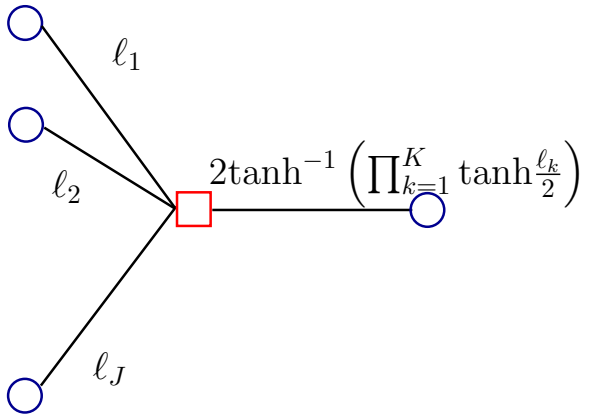
Message passing at variable nodes: the log-likelihood ratios from the children nodes add



Bitwise MAP Decoding for Binary Input Channels (cntd)

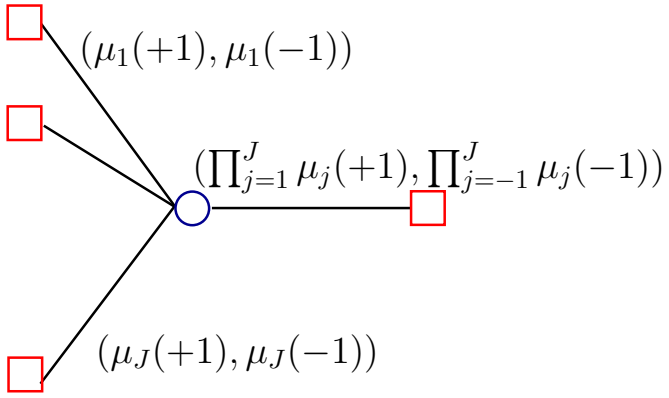
Message passing at check nodes:

$$\ell = 2\tanh^{-1}\left(\prod_{k=1}^{K}\tanh\frac{\ell_k}{2}\right)$$



Bitwise MAP Decoding for Binary Input Channels: Proof

In the general message passing algorithm a variable node multiplies messages from children nodes



In terms of likelihood ratios $\rho_j = \frac{\mu_j(+1)}{\mu_j(-1)}$, the message passing rule reads

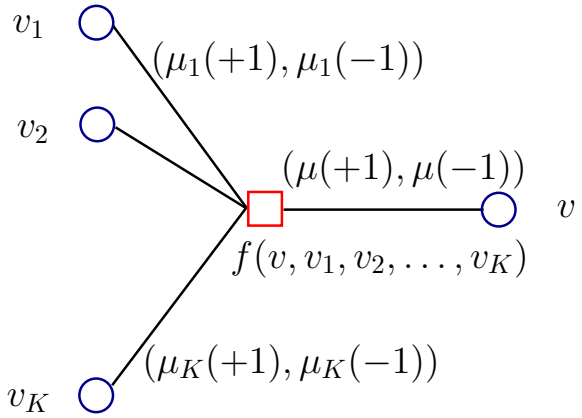
$$\rho = \frac{\prod_{j=1}^{J} \mu_j(+1)}{\prod_{j=1}^{J} \mu_j(-1)} = \prod_{j=1}^{J} \rho_j.$$

In terms of log-likelihood ratios

$$\ell = \ln \rho = \sum_{j=1}^{J} \ln \rho_j = \sum_{j=1}^{J} \ell_j.$$

Bitwise MAP Decoding for Binary Input Channels: Proof (cntd)

Message passage processing at the check node



The message at the output of the check node is $(\mu(+1), \mu(-1))$ **with**

$$\mu(+1) = \sum_{\sim v} f(+1, v_1, v_2, \ldots, v_K) \prod_{k=1}^{K} \mu_k(v_k) \text{ and } \mu(-1) = \sum_{\sim v} f(-1, v_1, v_2, \ldots, v_K) \prod_{k=1}^{K} \mu_k(v_k)$$

and

$$f(v, v_1, v_2, \ldots, v_K) = \mathbf{1}_{\{\prod_{k=1}^{K} v_k = v\}}$$

The likelihood ratio is

$$\rho = \frac{\mu(+1)}{\mu(-1)} = \frac{\sum_{\sim v} f(+1, v_1, v_2, \ldots, v_K) \prod_{k=1}^{K} \mu_k(v_k)}{\sum_{\sim v} f(-1, v_1, v_2, \ldots, v_K) \prod_{k=1}^{K} \mu_k(v_k)}$$

Bitwise MAP Decoding for Binary Input Channels: Proof (cntd)

$$\begin{aligned}\rho &= \frac{\sum_{(v_1,v_2,\ldots v_K):\prod_{k=1}^{K} v_k=+1} \prod_{k=1}^{K} \mu_k(v_k)}{\sum_{(v_1,v_2,\ldots v_K):\prod_{k=1}^{K} v_k=-1} \prod_{k=1}^{K} \mu_k(v_k)} \\ &= \frac{\sum_{(v_1,v_2,\ldots v_K):\prod_{k=1}^{K} v_k=+1} \prod_{k=1}^{K} \frac{\mu_k(v_k)}{\mu_k(-1)}}{\sum_{(v_1,v_2,\ldots v_K):\prod_{k=1}^{K} v_k=-1} \prod_{k=1}^{K} \frac{\mu_k(v_k)}{\mu_k(-1)}} \\ &= \frac{\sum_{(v_1,v_2,\ldots v_K):\prod_{k=1}^{K} v_k=+1} \prod_{k=1}^{K} \rho_k^{(1+v_k)/2}}{\sum_{(v_1,v_2,\ldots v_K):\prod_{k=1}^{K} v_k=-1} \prod_{k=1}^{K} \rho_k^{(1+v_k)/2}}\end{aligned}$$

It can be verified that

$$\prod_{k=1}^{K}(\rho_j+1) \pm \prod_{k=1}^{K}(\rho_j-1) = 2 \sum_{(v_1,v_2,\ldots v_K):\prod_{k=1}^{K} v_k=\pm 1} \prod_{k=1}^{K} \rho_k^{(1+v_k)/2}$$

Then,

$$\rho = \frac{\prod_{k=1}^{K}(\rho_j+1) + \prod_{k=1}^{K}(\rho_j-1)}{\prod_{k=1}^{K}(\rho_j+1) - \prod_{k=1}^{K}(\rho_j-1)} = \frac{1+\prod_{k=1}^{K}\frac{(\rho_j-1)}{(\rho_j+1)}}{1-\prod_{k=1}^{K}\frac{(\rho_j-1)}{(\rho_j+1)}}$$

Bitwise MAP Decoding for Binary Input Channels: Proof (cntd)

Or

$$\frac{\rho-1}{\rho+1}=\prod_{k=1}^{K}\frac{\rho_k-1}{\rho_k+1} \tag{2}$$

Since, $\frac{\rho-1}{\rho+1}=\frac{e^{\ell}-1}{e^{\ell}+1}=\tanh\frac{\ell}{2}$ **we obtain from (2)**

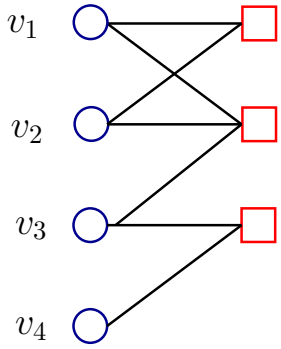
$$\tanh\frac{\ell}{2}=\prod_{k=1}^{K}\tanh\frac{\ell_k}{2}$$

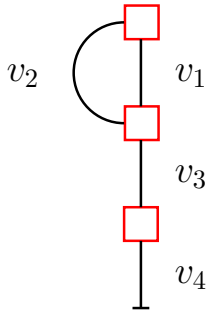
and the processing rule at the check nodes is

$$\ell=2\tanh^{-1}\prod_{k=1}^{K}\tanh\frac{\ell_k}{2}$$

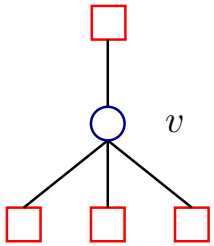
Forney Style Factor Graph (FSFG)

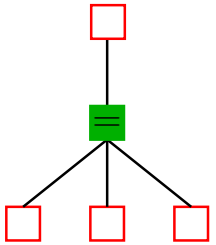
FSFG for factor graphs with variable nodes of degree one or two





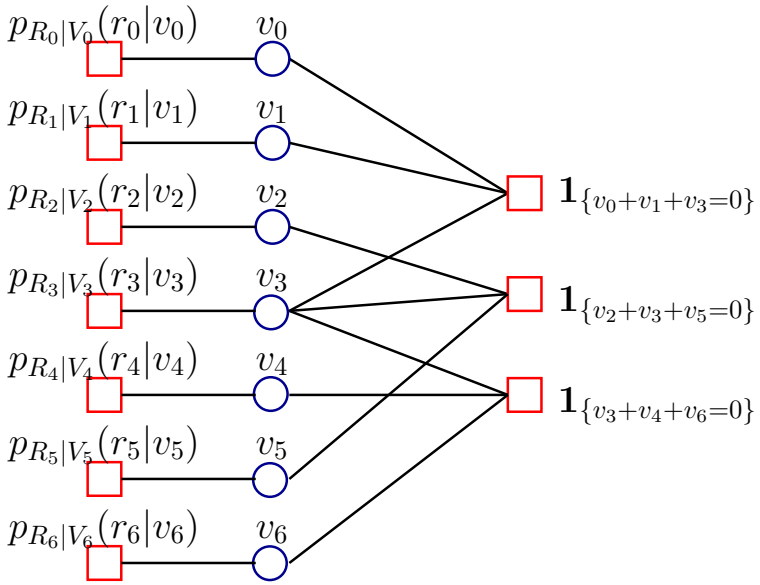
FSFG for factor graphs with variable nodes of degree higher than two

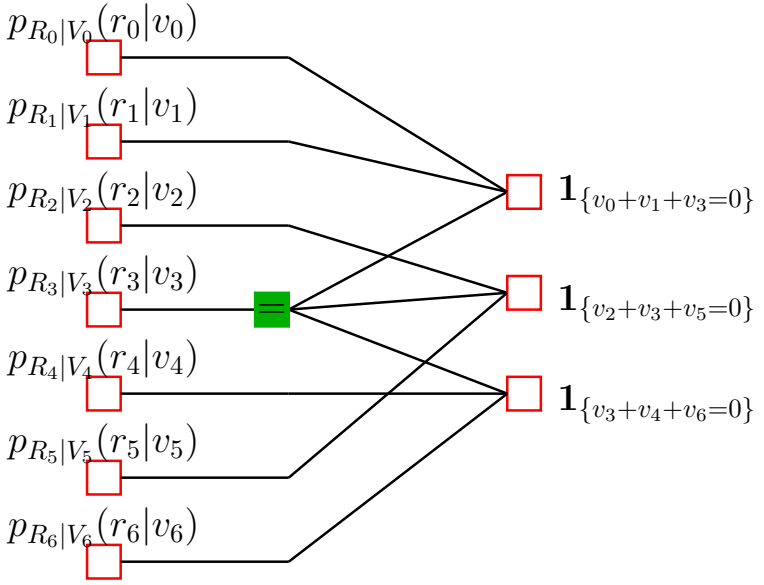




The extension of the processing rules to FSFG is straightforward.

Forney Style Factor Graph (FSFG): an Example





Other Applications of the Distributive Law

min-sum

$$\min(x+y, x+z) = x + \min(y, z) \qquad \min_{ij}(x_i + y_j) = \min_i(x_i) + \min_j(y_j)$$

max-sum

$$\max(x+y, x+z) = x + \max(y, z) \qquad \max_{ij}(x_i + y_j) = \max_i(x_i) + \max_j(y_j)$$

min-product

$$\min(x \cdot y, x \cdot z) = x \cdot \min(y, z) \qquad \min_{ij}(x_i \cdot y_j) = \min_i(x_i) \cdot \min_j(y_j)$$

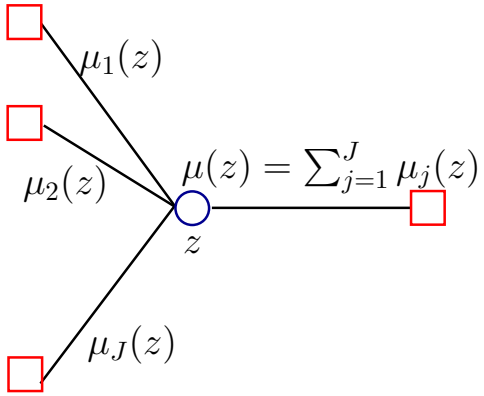
max-product

$$\max(x \cdot y, x \cdot z) = x \cdot \max(y, z) \qquad \max_{ij}(x_i \cdot y_j) = \max_i(x_i) \cdot \max_j(y_j)$$

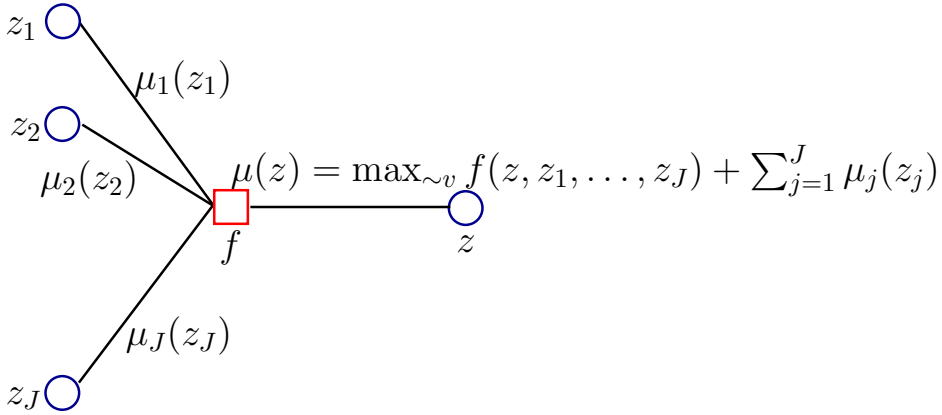
The message passing algorithm can be readily generalized to these cases.

Example: Max-Sum

Processing rule at the variable node



Processing rule at the check node



Word-wise MAP Decoding for Binary Input Channels

$$\widehat{\boldsymbol{v}}^{\mathrm{MAP}}(\boldsymbol{r}) = \underset{\boldsymbol{v}}{\mathrm{argmax}} p_{\boldsymbol{V}|\boldsymbol{R}}(\boldsymbol{v}|\boldsymbol{r}) = \underset{\boldsymbol{v}}{\mathrm{argmax}} p_{\boldsymbol{R}|\boldsymbol{V}}(\boldsymbol{r}|\boldsymbol{v}) p_{\boldsymbol{V}}(\boldsymbol{v})$$

$$= \underset{\boldsymbol{v}}{\mathrm{argmax}} \prod_j p_{R_j|V_j}(r_j|v_j) \qquad \left(\begin{array}{c} v_j \text{ equiprobable} \\ \text{and memoryless channel} \end{array} \right)$$

Component-wise it can expressed as

$$(\widehat{\boldsymbol{v}}^{\mathrm{MAP}}(\boldsymbol{r}))_i = \underset{v_i \in \{\pm 1\}}{\mathrm{argmax}} \max_{\sim v_i} \left(\prod_j p_{V_j|R_j}(v_j|r_j) \right) \mathbf{1}_{\{\boldsymbol{v} \in \mathcal{C}\}}$$

$$= \underset{v_i \in \{\pm 1\}}{\mathrm{argmax}} \left(\max_{\sim v_i} \sum_j \log p_{V_j|R_j}(v_j|r_j) + \log \mathbf{1}_{\{\boldsymbol{v} \in \mathcal{C}\}} \right)$$

Note that if $\boldsymbol{v} \notin \mathcal{C}$ then $\log \mathbf{1}_{\{\boldsymbol{v} \in \mathcal{C}\}} = -\infty$.

Word-wise MAP Decoding for Binary Input Channels (cntd)

Compare the blockwise MAP decoding with the bitwise MAP decoding

$$\widehat{\boldsymbol{v}}_i^{\mathrm{MAP}}(\boldsymbol{r}) = \underset{v_i \in \{\pm 1\}}{\operatorname{argmax}} \sum_{\sim v_i} \prod p_{R_j|V_j}(r_j|v_j) \mathbf{1}_{\{\boldsymbol{v} \in \mathcal{C}\}}$$

We can perform the word-wise MAP decoding via message passing as for the bitwise MAP decoding via message passing by replacing the processing rules for sum-product by the processing rules for max-product (or max-sum if we work with log-likelihood ratios).

Message Passing Algorithm and Channel Decoding

Message passing enables to determine marginals efficiently if the factor graph is a tree...

...However, channel codes whose Tanner graph is a tree are not very powerful!

Proposition: Let $\mathcal{C}$ be a binary linear code with rate R and codeword length n which admits a Tanner graph which is a forest. Then, $\mathcal{C}$ contains at least $\frac{2R-1}{2}n$ codewords of weight 2 (i.e. the minimum distance of the code is 2).

In general cycle-free codes have many low weight codewords and hence an high error probability!

Alternative Approaches

- **Consider alternative graph representations of channel codes (ex state nodes) which are cycle-free: this is the case of convolutional codes (BCJR is a message passing algorithm!).**
- **Consider not binary codes.**
- **Apply the message passing algorithm also in case of Tanner graphs with cycles. Then, the message passage algorithm is suboptimum.**

LDPC Codes: Definition

LDPC codes are linear binary block codes which have at least a sparse Tanner graph, or equivalently their parity check matrix is sparse.

What does it mean sparse?

Consider a parity check matrix H and assume that it is randomly generated with equal probability of having zeros or ones entries. Then the average number of ones is $\frac{n^2}{2}(1-R) \sim n^2$. A matrix H is sparse if the number of its ones is proportional to n.

How to obtain a sparse matrix?

For example, the number of ones in each row or each column is fixed and independent of n.

What is the benefit of having a sparse matrix?

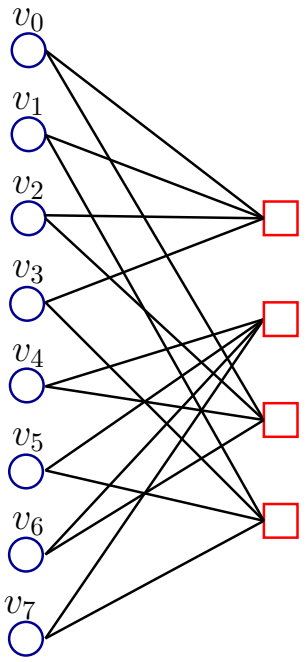
- **They exhibit good performance under message passing decoding.**
- **It is possible to have encoding and decoding algorithms with linear complexity in n, i.e. constant complexity per decoded symbol.**

Regular LDPC Codes

An (l, r)-regular LDPC code is a linear binary block code that has a parity check matrix $\boldsymbol{H}$ with l ones in each column and r ones in each row, where $l \ll n-k$ and $r \ll n$. Equivalently, each variable node in the Tanner graph has degree l and each check node has degree r.

Example

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



Regular LDPC Codes (cntd)

A Tanner graph is regular if all its nodes have the same degree.

Rate of a (d_v, d_c)-regular LDPC code

$$\begin{aligned} n \text{ variable nodes} &\implies nd_v \text{ edges.} \\ m = n - k \text{ check nodes} &\implies md_c \text{ edges.} \end{aligned}$$

Then, since $nd_v = md_c$, the code rate$^{(*)}$ is

$$\frac{k}{n} \geq \frac{n-m}{n} = \underbrace{1 - \frac{d_v}{d_c}}_{\text{Design Rate}}$$

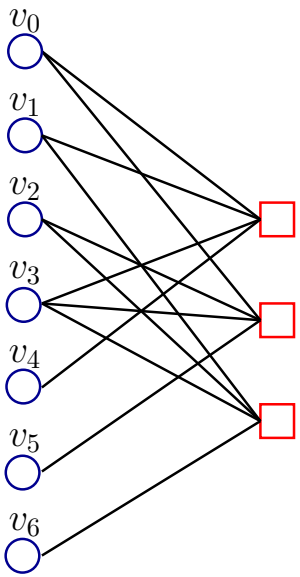
* Inequality because some check equations may be dependent (since randomly generated).

Irregular LDPC Codes

The behavior of LDPC codes can be significantly improved by allowing nodes of different degrees.

An irregular LDPC code is an LDPC code for which the degrees of nodes are chosen according to some distribution.

Example: $(7, 4, 3)$ Hamming Code



- Λ_i **number of variable nodes of degree** i
- P_i **number of check nodes of degree** i

$$\Lambda_1 = 3, \quad \Lambda_2 = 3, \quad \Lambda_3 = 1$$
$$P_4 = 3.$$

Degree Distributions from a Node Perspective

Λ_i number of variable nodes of degree $i \Rightarrow n = \sum \Lambda_i$

P_i number of check nodes of degree $i \Rightarrow m = \sum P_i$

Design Rate

$$r(\Lambda, P) = \frac{n-m}{n} = 1 - \frac{\sum P_i}{\sum \Lambda_i}$$

Since the number of edges must match up

$$\sum_{i=1}^{l_{\max}} i\Lambda_i = \sum_{i=1}^{r_{\max}} iP_i$$

Useful Notation: Variable and Check Degree Distributions from a Node Perspective

$$\Lambda(x) = \sum_{i=1}^{l_{\max}} \Lambda_i x^i \qquad P(x) = \sum_{i=1}^{r_{\max}} P_i x^i$$

Then,

$$\Lambda(1) = n \qquad P(1) = m \qquad r(\Lambda, P) = 1 - \frac{P(1)}{\Lambda(1)} \qquad \Lambda'(1) = P'(1)$$

Degree Distributions from a Node Perspective

Normalized Variable and Check Degree Distributions

$$L(x) = \frac{\Lambda(x)}{\Lambda(1)} \qquad R(x) = \frac{P(x)}{P(1)}$$

Example: $(7, 4, 3)$ Hamming Code

$$\begin{aligned} \Lambda(x) &= 3x + 3x^2 + x^3 \\ L(x) &= \frac{3}{7}x + \frac{3}{7}x^2 + \frac{1}{7}x^3 \end{aligned}$$

$$\begin{aligned} P(x) &= 3x^4 \\ R(x) &= x^4 \end{aligned}$$

Degree Distributions from an Edge Perspective

Fraction of edges connected to variable nodes of degree i: $\lambda_i = \dfrac{i\Lambda_i}{\#\text{of edges}}$

Fraction of edges connected to check nodes of degree i: $\rho_i = \dfrac{iP_i}{\#\text{ of edges}}$

$$\#\text{ of edges} = \sum i\Lambda_i = \sum iP_i$$

Variable Degree Distribution

$$\lambda(x) = \sum \lambda_i x^{i-1} = \frac{\sum i\Lambda_i x^{i-1}}{\sum i\Lambda_i} = \frac{\Lambda'(x)}{\Lambda'(1)} = \frac{L'(x)}{L'(1)}$$

Check Degree Distribution

$$\rho(x) = \sum \rho_i x^{i-1} = \frac{\sum iP_i x^{i-1}}{\sum iP_i} = \frac{P'(x)}{P'(1)} = \frac{R'(x)}{R'(1)}$$

Degree Distributions from an Edge and Node Perspective: Relations

$$\frac{\Lambda(x)}{n} = L(x) = \frac{\int_0^x \lambda(z)\mathrm{d}z}{\int_0^1 \lambda(z)\mathrm{d}z} \qquad \frac{P(x)}{m} = R(x) = \frac{\int_0^x \rho(z)\mathrm{d}z}{\int_0^1 \rho(z)\mathrm{d}z}$$

Degree Distributions from an Edge Perspective (cntd)

Average number of edges per variable node

$$\mathfrak{l}_{\text{avg}} = \frac{\sum i\Lambda_i}{\Lambda(1)} = L'(1) = \frac{1}{\int_0^1 \lambda(x)\mathrm{d}x}$$

Proof:

$$\Lambda_i = \#\text{ of edges } \frac{\lambda_i}{i} \qquad n = \#\text{ of edges } \sum_i \frac{\lambda_i}{i}$$

$$\mathfrak{l}_{\text{avg}} = \frac{\#\text{ of edges } \sum_i \lambda_i}{n} = \frac{\sum_i \lambda_i}{\sum_i \frac{\lambda_i}{i}} = \frac{1}{\int_0^1 \lambda(x)\mathrm{d}x}$$

Average number of edges per check node

$$\mathfrak{r}_{\text{avg}} = \frac{\sum iP_i}{P(1)} = R'(1) = \frac{1}{\int_0^1 \rho(x)\mathrm{d}x}$$

Proof:

$$P_i = \#\text{ of edges } \frac{\rho_i}{i} \qquad m = \#\text{ of edges } \sum_i \frac{\rho_i}{i}$$

$$\mathfrak{r}_{\text{avg}} = \frac{\#\text{ of edges } \sum_i \rho_i}{m} = \frac{\sum_i \rho_i}{\sum_i \frac{\rho_i}{i}} = \frac{1}{\int_0^1 \rho(x)\mathrm{d}x}$$

Design Rate

$$r(\lambda,\rho) = 1 - \frac{\mathfrak{l}_{\text{avg}}}{\mathfrak{r}_{\text{avg}}} = 1 - \frac{L'(1)}{R'(1)} = 1 - \frac{\int_0^1 \rho(x)\mathrm{d}x}{\int_0^1 \lambda(x)\mathrm{d}x}$$

Examples

Example 1: $(7, 4, 3)$ Hamming Code

$$L(x) = \frac{3}{7}x + \frac{3}{7}x^2 + \frac{1}{7}x^3 \qquad R(x) = x^4$$

$$\lambda(x) = \frac{L'(x)}{L'(1)} = \frac{1}{4} + \frac{2}{4}x + \frac{1}{4}x^2 \qquad \rho(x) = \frac{R'(x)}{R'(1)} = x^3$$

Interpretation of $\lambda(x)$:

- $\frac{1}{4}$ **of the edges are connected to variable nodes of degree 1.**
- $\frac{2}{4}$ **of the edges are connected to variable nodes of degree 2.**
- $\frac{1}{4}$ **of the edges are connected to variable nodes of degree 3.**

Interpretation of $\rho(x)$:

- **All edges are connected to check nodes of degree 4.**

Example 2

$$\Lambda(x) = 613x^2 + 202x^3 + 57x^4 + 84x^7 + 44x^8 \qquad P(x) = 500x^6$$

Number of Variable Nodes: $\Lambda(1) = 1000$ **Number of Check Nodes:** $P(1) = 500$

Number of Edges: $\Lambda'(1) = P'(1) = 3000$

$$L(x) = \frac{\Lambda(x)}{\Lambda(1)} = \frac{613}{1000}x^2 + \frac{202}{1000}x^3 + \frac{57}{1000}x^4 + \frac{84}{1000}x^7 + \frac{44}{1000}x^8 \qquad R(x) = \frac{P'(x)}{P'(1)}x^5$$

$$\lambda(x) = \frac{\Lambda'(x)}{\Lambda'(1)} = \frac{1226}{3000}x + \frac{606}{3000}x^2 + \frac{228}{3000}x^3 + \frac{588}{3000}x^6 + \frac{352}{3000}x^7 \qquad \rho(x) = \frac{P'(x)}{P'(1)} = x^5$$

Interpretation of $\lambda(x)$:

- **$\frac{1226}{3000}, \frac{606}{3000}, \frac{228}{3000}, \frac{588}{3000}, \frac{352}{3000}$ of the edges are connected to variable nodes of degree 2,3,4,7, and 8, respectively.**

Interpretation of $\rho(x)$:

- **All edges are connected to check nodes of degree 6.**

Ensemble LDPC(Λ, P) of Bipartite Graphs

Given an degree distribution pair (Λ, P), define an ensemble of bipartite graphs LDPC(Λ, P) in the following way.

- **Each graph in LDPC(Λ, P) has $\Lambda(1)$ variable nodes and $P(1)$ check nodes.**
- **Λ_i variable nodes and P_i check nodes have degree i.**
- **A variable/check node of degree i has i sockets from which the i edges emanate (in total there are $\Lambda'(1) = P'(1)$ sockets in each side).**
- **The sockets in each side are labelled with the set $[\Lambda'(1)] = \{1, \ldots \Lambda'(1)\}$ in an arbitrary but fixed way.**
- **Given a permutation σ of $\{1, 2 \ldots \Lambda'(1)\}$ associate to σ a bipartite graph connecting the i socket on the variable side to the $\sigma(i)$-th socket in the check node side.**
- **Varying σ over the set of permutations on $[\Lambda'(1)]$ we generate the set of bipartite graphs.**
- **Define a probability distribution over the set of graphs by placing the uniform probability distribution on the set of permutations.**

Standard Ensemble LDPC(Λ, P)

Given an ensemble of bipartite graphs LDPC(Λ, P) we can associate a parity check matrix to each bipartite graph in the following way.

- **The parity check matrix has a nonzero entry at row i and column j if and only if the i-th check node is connected to the j-th variable node an odd number of times.**

Remarks

- **Since each bipartite graph in the ensemble LDPC(Λ, P) is associate to a parity check matrix we can interchange the terms and refer to a bipartite graph or a code in LDPC(Λ, P).**
- **The bipartite graphs are uniformly distributed in the standard ensemble LDPC(Λ, P) while the codes are not uniformly distributed.**
- **To pick at random a code from LDPC(Λ, P) means to pick at random a bipartite graph.**
- **The bipartite graphs in the ensemble LDPC(Λ, P) have all the same number of variable and check nodes. We can consider also the ensemble with normalized variable and check degree distributions $L(x)$ and $R(x)$. This is equivalent to consider the ensemble obtained by the union of all standard ensembles LDPC(nL, mR) with $n \in \mathbb{N}$.**
- **We consider only ensembles without degree one nodes. In fact, degree one nodes imply that there is nonzero probability of having codewords with weight 2.**

Rate and Design Rate

Lemma: Consider the ensemble LDPC(n, λ, ρ) $\stackrel{\circ}{=}$**LDPC**(n, L, R). **Let** $r(\lambda, \rho)$ **denotes the design rate of the ensemble and let** $r(\mathcal{C})$ **denotes the actual rate of the code** $\mathcal{C}, \mathcal{C} \in$ LDPC(n, λ, ρ). **Consider the function** $\psi(y)$.

$$\psi(y) = -L'(1)\log_2\left[\frac{(1-yz)}{1+z}\right] + \sum_i L_i \log_2\left[\frac{1+y^i}{2}\right] + \frac{L'(1)}{R'(1)}\sum_j R_j \log_2\left[1+\left(\frac{1-z}{1+z}\right)^j\right]$$

with

$$z = \left(\sum_i \frac{\lambda_i y^{i-1}}{1+y^i}\right) / \left(\sum_i \frac{\lambda_i}{1+y^i}\right).$$

If for $y \geq 0$ $\psi(y) \leq 0$ **with equality only at** $y = 1$ **then for** $\xi > 0$ **and any** $n > n(\xi)$

$$\Pr\{r(\mathcal{C})n - r(\lambda, \rho)n < \xi\} \leq \mathrm{e}^{-n\xi\ln(2)/2}$$

Rate and Design Rate for Regular Codes

In case of regular LDPC codes the rate design always converges to the design rate, eventually up to a fraction $\frac{1}{n}$, as $n \to \infty$.

Corollary: Consider the regular ensemble LDPC$(nx^{\mathfrak{l}}, n\frac{\mathfrak{l}}{\mathfrak{r}}x^{\mathfrak{r}})$ with $2 \leq \mathfrak{l} \leq \mathfrak{r}$. Let $r(\lambda, \rho) = 1 - \frac{\mathfrak{l}}{\mathfrak{r}}$ denotes the design rate of the ensemble and let $r(\mathcal{C})$ denotes the actual rate of the code $\mathcal{C}$, $\mathcal{C} \in$ LDPC$(nx^{\mathfrak{l}}, n\frac{\mathfrak{l}}{\mathfrak{r}}x^{\mathfrak{r}})$. Then

$$\Pr\{r(\mathcal{C})n = r(\lambda, \rho)n + \nu\} \to 1 - o_n(1)$$

where $\nu = 1$ if all variable nodes have even degree and $\nu = 0$ otherwise.

Remarks

$\nu = 1$ if all variable nodes have even degree because in such a case the sum of all constraints is zero. Therefore, the constraints are not independent.

Decoding of Linear Block Codes for BEC

Consider the parity check matrix H and the received signal r.

Let $\mathcal{E} \subseteq \{1, 2, \ldots n\}$ be the set of indices of r such that $r_i = \Delta$, for $i \in \mathcal{E}$

Let $\mathcal{U} = \overline{\mathcal{E}}$.

Let $\boldsymbol{H}(\mathcal{E})\,(\boldsymbol{r}(\mathcal{E}))$ be the submatrix (subvector) of $\boldsymbol{H}\,(r)$ obtained by considering the columns (elements) with indices in $\mathcal{E}$ and let $\boldsymbol{H}(\mathcal{U})\,(\boldsymbol{r}(\mathcal{U}))$ be the submatrix (subvector) of $\boldsymbol{H}\,(r)$ obtained by considering the columns (elements) with indices in $\mathcal{U}$

Then, we have the system of equations

$$\boldsymbol{r}(\mathcal{E})\boldsymbol{H}^T(\mathcal{E}) = \boldsymbol{r}(\mathcal{U})\boldsymbol{H}^T(\mathcal{U}).$$

The system has unique solution if $\operatorname{rank}(\boldsymbol{H}(\mathcal{E})) = |\mathcal{E}|$.
Since $\min \operatorname{rank}(\boldsymbol{H}(\mathcal{E})) = d - 1$, being d the minimum Hamming distance, it follows:

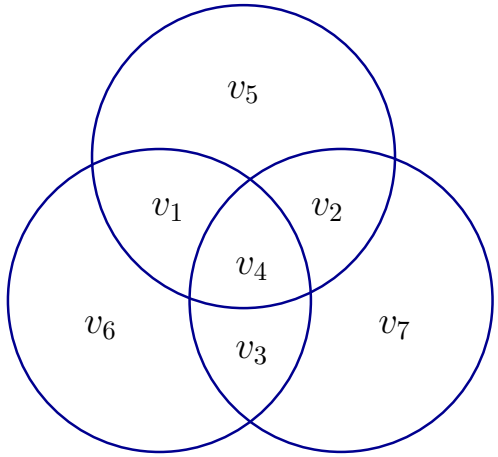
A linear binary (n, k, d) code is able to correct all patterns of $e \leq d - 1$ erasures

First Tour of Iterative Decoding

Consider the $(7, 4, 3)$ Hamming code defined by the parity check equations

$$\begin{aligned} v_1 + v_2 + v_4 &= v_5 \\ v_1 + v_3 + v_4 &= v_6 \\ v_2 + v_3 + v_4 &= v_7 \end{aligned}$$

Let us represent these constraints by a Venn diagram

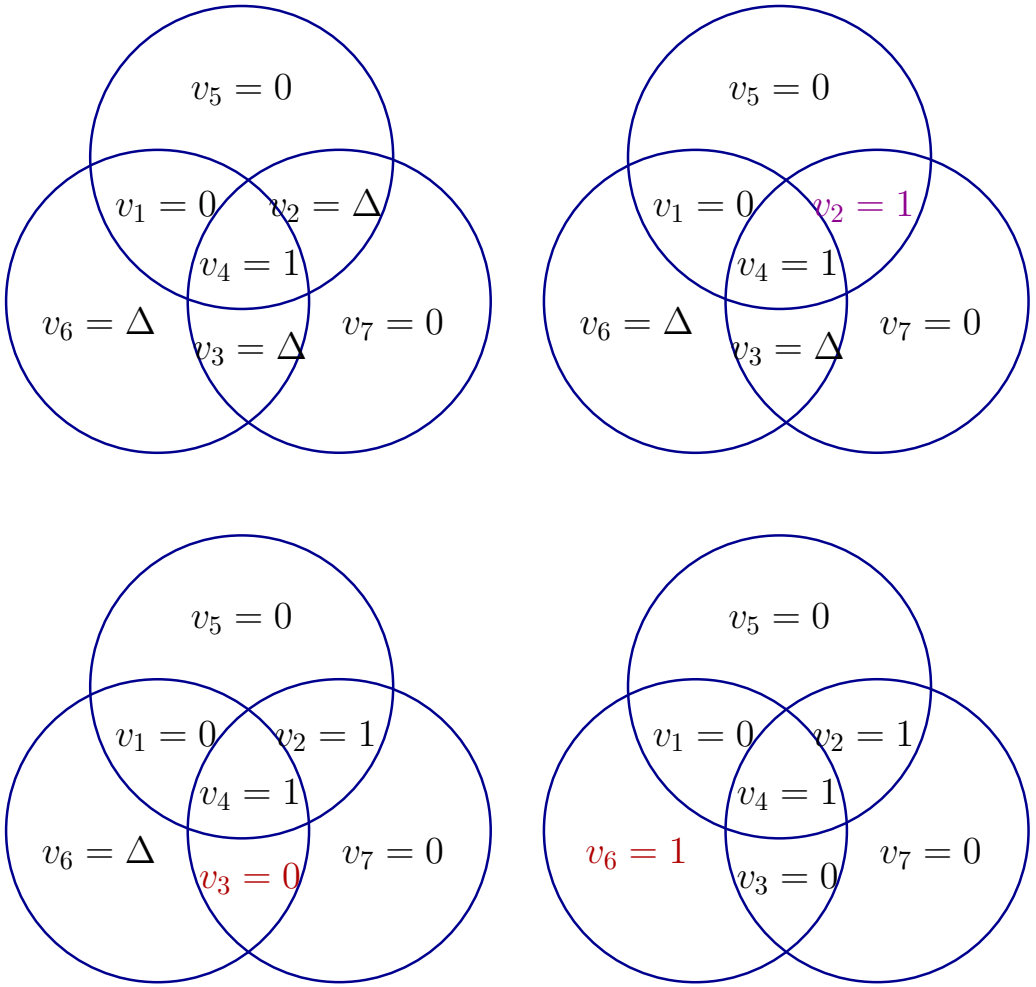


Encoding Problem

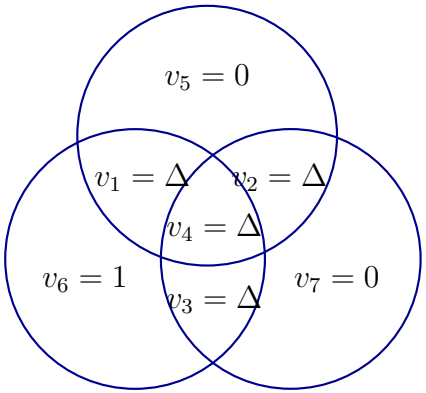
Given (v_1, v_2, v_3, v_4) it is straightforward to determine (v_5, v_6, v_7) :
The number of ones in each circle has to be even.

First Tour of Iterative Decoding (ctnd)

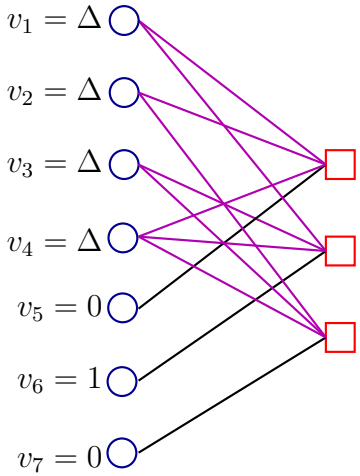
Example: $\boldsymbol{r} = (0, \Delta, \Delta, 1, 0, \Delta, 0)$



Example 2: $\boldsymbol{r} = (\Delta, \Delta, \Delta, \Delta, 0, 1, 0)$



The ambiguity cannot be solved!
The set $\mathcal{S} = (v_1, v_2, v_4)$ is a stopping set !



Tanner Graph

Characteristics: All check nodes connected to the nodes in the stopping set $\mathcal{S} = (v_1, v_2, v_4)$ are connected to at least two variable nodes!

The stopping sets characterize the performance of iterative APP decoders for finite length LDPC codes on BECs.

Message Passing Algorithms for Binary Erasure Channel

Scheduling:

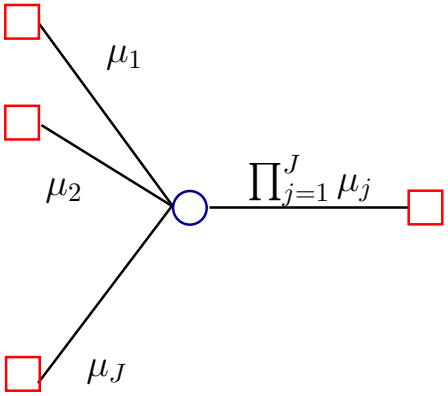
- **If the Tanner graph is a tree we have a natural scheduling starting from the leaf nodes and encoding a message once all incoming messages required for the computation are available.**
- **If the Tanner graph is not a tree we can define different scheduling.**
  Our convention is to proceed in rounds.
  A round consists of
  - **Processing the incoming messages of the check nodes and then send the resulting outgoing messages to variable nodes along all edges.**
  - **Processing the incoming messages at the variable nodes and then send the outgoing message to the variable nodes along all edges.**

Remarks

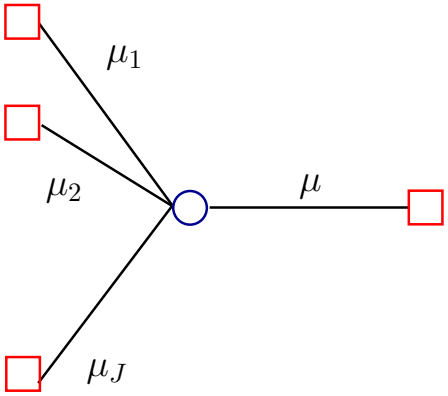
- **The scheduling for a Tanner graph consists of activating in successive time units all check nodes and then all variable nodes.**
- **In the initial round the variable nodes simply send the message received from the channel to their neighboring check nodes.**

Message Passing Rules at Variable Nodes

General Rule



Rule for BEC



The message μ is an erasure if all incoming messages are erasures. Otherwise it is equal to zero or equal to one.

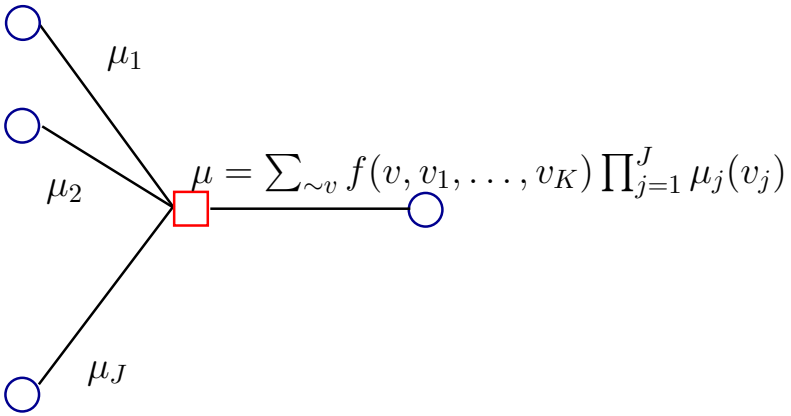
Rationale: If some incoming messages are not erasures they should be or all equal to zero or all equal to one since the channel does not introduce errors. Their value is also the value of the message at the variable node.

This rule can be formally obtained from the general rule with the following mapping:

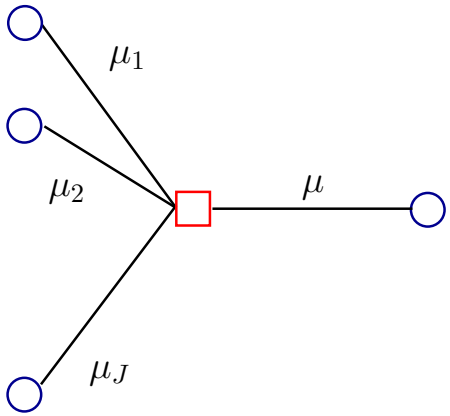
$$\begin{aligned} 0 \rightarrow \quad & (p(0|0), p(0|1)) = (1-\varepsilon, 0) \rightarrow \quad (1,0) \\ \Delta \rightarrow \quad & (p(\Delta|0), p(\Delta|1)) = (\varepsilon, \varepsilon) \rightarrow \quad (1,1) \\ 1 \rightarrow \quad & (p(1|0), p(1|1)) = (0, 1-\varepsilon) \rightarrow \quad (0,1) \end{aligned}$$

Message Passing Rules at Check Nodes

General Rule



Rule for BEC



The message μ is an erasure if any of the incoming messages is an erasures. If all incoming messages are zero or one then the outgoing message μ is the mod-2 sum of the incoming messages.

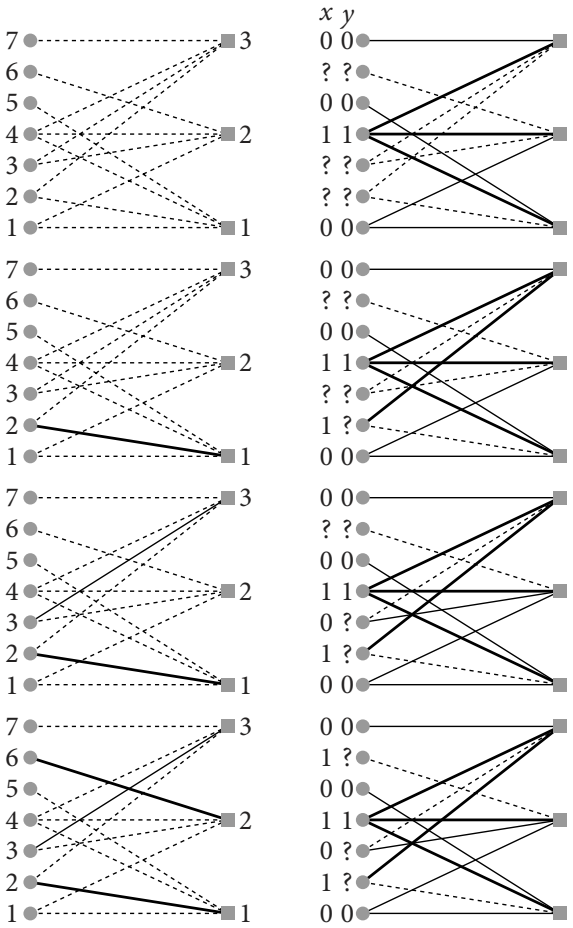
Formally, it can be seen for $J = 2$:

$$
\begin{aligned}
(\mu(0), \mu(1)) &= \left(\sum_{v_1, v_2} \mathbf{1}_{v_1+v_2=0} \mu_1(v_1)\mu_2(v_2), \sum_{v_1, v_2} \mathbf{1}_{v_1+v_2=1} \mu_1(v_1)\mu_2(v_2) \right) \\
&= (\mu_1(0)\mu_2(0) + \mu_1(1)\mu_2(1), \mu_1(0)\mu_2(1) + \mu_1(1)\mu_2(0))
\end{aligned}
$$

Let us assume that $v_1 = \Delta$, i.e. $(\mu_1(0), \mu_1(1)) = (1, 1)$.

It can be verified that $(\mu(0), \mu(1)) = (1, 1)$ for any value of v_2.

Example



y: received word $\quad\quad$ $\widehat{x}$: current estimate of the transmitted word

Iterative Decoding

- **In case of BEC any meaningful message passing algorithm is equivalent.**

- **A special class of message passing algorithms is the class of BELIEVE PROPAGATION. In the believe propagation algorithm the messages represent probabilities or beliefs.**

- **The believe propagation algorithm plays a relevant role for general channels.**

Simplifying Analysis and Design of LDPC Codes

- **The performance of LDPC codes is independent of the transmitted codeword and depends only on the erasure pattern.**

$$\Downarrow$$

We can focus on the all zeros transmitted codeword.

- **Theorem: Let the code $\mathcal{C}$, chosen uniformly at random from LDPC(n, λ, ρ), be used for the transmission over a BEC(ε). Assume that the decoder performs ℓ rounds of message passing decoding and let $P_b^{\mathrm{BP}}(\mathcal{C}, \varepsilon, \ell)$ denote the resulting bit erasure probability.**
  Then, for λ fixed and for any given $\delta > 0$, there exists an $\alpha > 0, \alpha = \alpha(\lambda, \rho, \varepsilon, \delta, \ell)$ such that

$$\Pr\{|P_b^{\mathrm{BP}}(\mathcal{C}, \varepsilon, \ell) - \mathrm{E}_{\mathcal{C}' \in \mathrm{LDPC}(n,\lambda,\rho)}(P_b^{\mathrm{BP}}(\mathcal{C}', \varepsilon, \ell))| > \delta\} \leq \mathrm{e}^{-\alpha n}$$

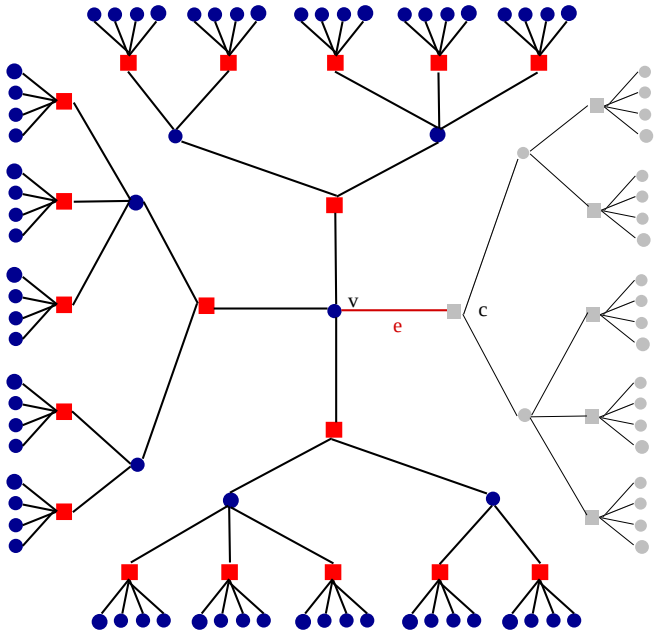
$$\Downarrow$$

Interpretation: Individual elements of an ensemble behave with high probability close to the ensemble average. We can focus on the design and construction of ensembles whose average performance approaches the Shannon limit.

Average Performance Analysis: Preliminary Definition

Definition of Directed Neighborhood: **Let $e = (v \to c)$ be the directed edge in the Tanner graph of a code $\mathcal{C}$ connecting the variable node v to the check node c. The directed neighborhood of e of depth d, denoted by $\mathcal{N}_e^{(d)}$, is the subgraph including all paths of length d starting from the variable node v and not including the edge e.**

Example



Average Performance Analysis of the Decoding Algorithm

Object: Compute the average probability $z^{(k)}$ that a message output of a variable node at the k-th decoder iteration is an erasure when the average is taken over

- **the codes of the ensemble LDPC(n, λ, ρ),**
- **the ensemble of possible output $\boldsymbol{r}$,**

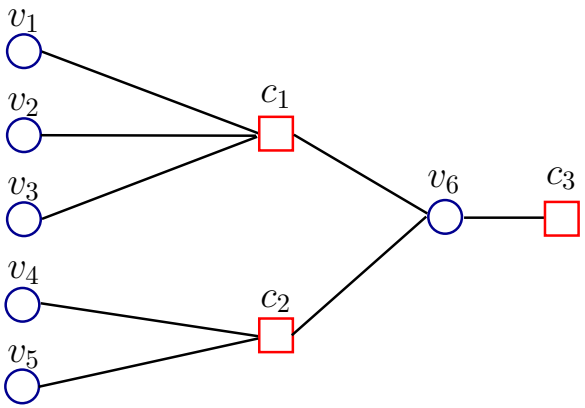
and it is conditioned with respect to the cycle-free assumption

$$z^{(k)} = \sum_{\mathcal{C}\in \mathrm{LDPC}(n,\lambda,\rho), \boldsymbol{r}\in\{0,1,\Delta\}^n} \Pr\{\mu_e = \Delta | \mathcal{N}_e^{2k+1} \text{ is cycle-free}, \mathcal{C}, \boldsymbol{r}\} \Pr(\mathcal{C}) \Pr(\boldsymbol{r})$$

Computation of $z^{(k)}$

If $\mathcal{N}_\ell^{(2k+1)}$ is cycle-free then the messages ingoing at each node of $\mathcal{N}_\ell^{(2k+1)}$ are statistically independent.

Assume $z^{(k-1)}$ known (in the example we know the average probability $v_1, v_2, \ldots, v_5$.)



$$\Pr\{\text{output of } c_1 \neq \Delta\} = (1 - z^{(k-1)})^3$$

For any check node of degree j

$$\Pr\{\text{output of } c \neq \Delta\} = (1 - z^{(k-1)})^{j-1}$$

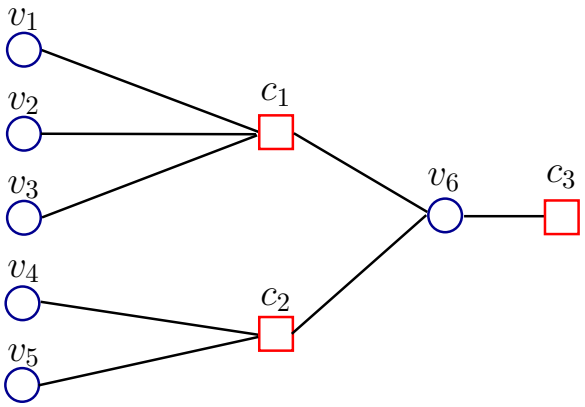
$$q_j \equiv \Pr\{\text{output of } c = \Delta\} = 1 - (1 - z^{(k-1)})^{j-1}$$

Average probability that the incoming message to v_6 is an erasure

$$\rho_4(1 - (1 - z^{(k-1)})^3) + \rho_3(1 - (1 - z^{(k-1)})^2)$$

For any variable node v the average probability that the ingoing message to v is an erasure is

$$q = \sum_j \rho_j q_j = \sum_j \rho_j (1 - (1 - z^{(k-1)})^{j-1}) = 1 - \rho(1 - z^{(k-1)})$$



For any check node of degree j

$$\Pr\{\text{output of } v_6 = \Delta\} = \varepsilon q^2$$

ε takes into account the probability that the channel symbol is an erasure

For a variable node v of degree i the average probability that the outgoing message to v is an erasure is

$$\Pr\{\text{output of } v = \Delta\} = \varepsilon q^{i-1}$$

Average over all variable nodes

$$z^{(k)} = \sum_i \lambda_i \varepsilon q^{i-1} = \varepsilon \lambda(q) = \varepsilon \lambda(1 - \rho(1 - z^{(k-1)}))$$

The recursion

$$z^{(k)} = \varepsilon \lambda(1 - \rho(1 - z^{(k-1)}))$$

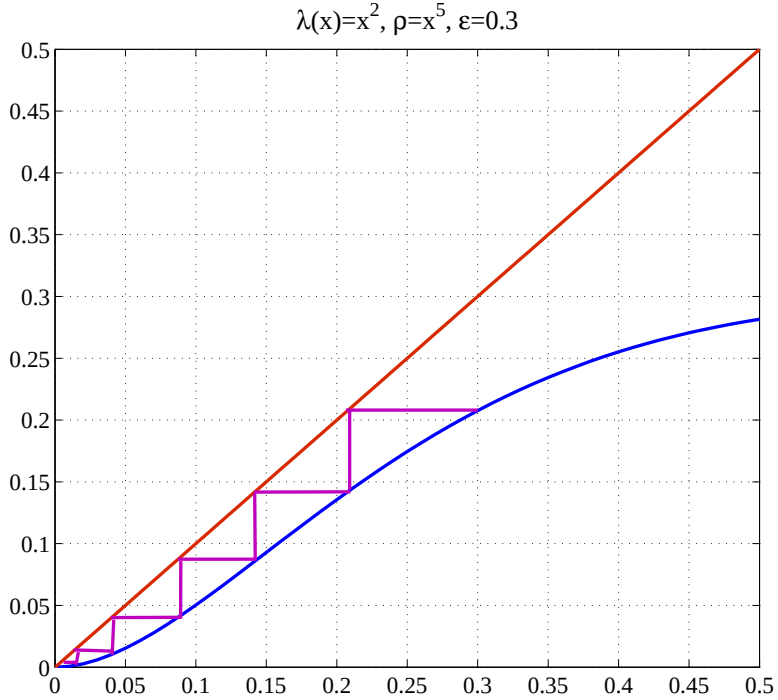
with $z^{(0)} = \varepsilon$ describes the evolution of the average erasure message probability.

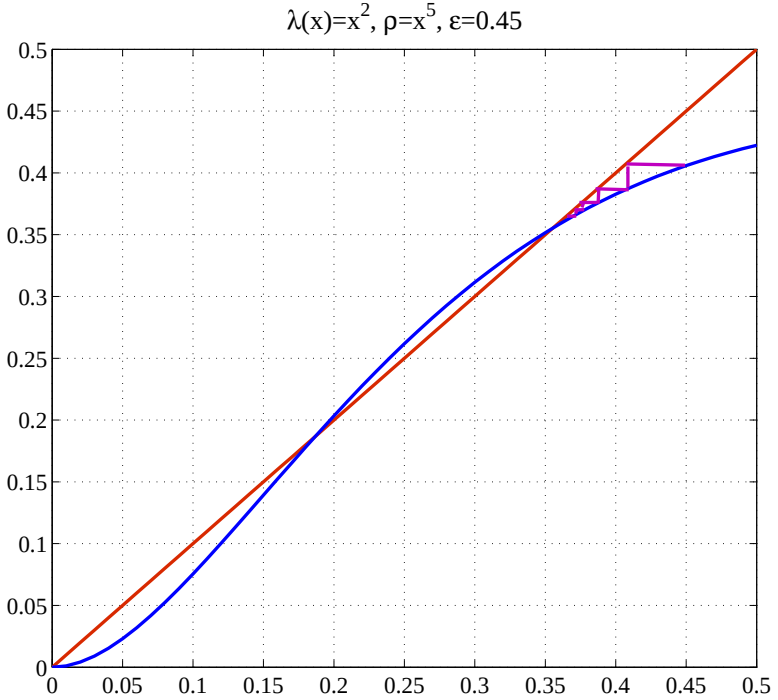
Examples of Density Evolution

Density evolution of the regular (3,6)-LDPC ensemble $\lambda(x) = x^2,\ \rho(x) = x^5$.

$$z^{(k)} = \varepsilon\left(1 - \left(1 - z^{(k-1)}\right)^5\right)^2$$

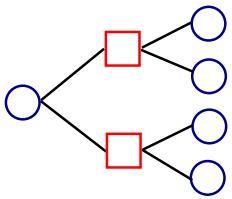
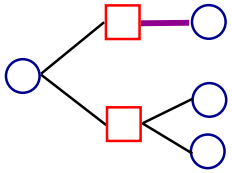
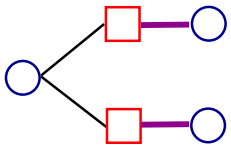
with $z^{(0)} = \varepsilon$.

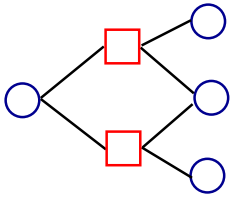
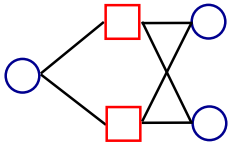
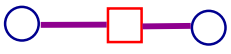




Extension to Graphs with Cycles

Consider for example the ensemble LDPC($n, \lambda(x) = x, \rho(x) = x^2$)
The directed neighborhood of an edge of depth $d = 2$ can be one of the following types.

$\mathcal{N}_e^{(2)}$	$\Pr\{\mathcal{N}_e^{(2)}\}$ as $n \to \infty$	Erasure probability
	$\Pr\{\mathcal{N}_e^{(2)}\} \to \text{const}$	$\varepsilon(1-(1-\varepsilon)^2)^2$
	$\Pr\{\mathcal{N}_e^{(2)}\} \to o(n^{-1})$	$\varepsilon^2(1-(1-\varepsilon)^2)$
	$\Pr\{\mathcal{N}_e^{(2)}\} \to o(n^{-2})$	ε^3

$\mathcal{N}_e^{(2)}$	$\Pr\{\mathcal{N}_e^{(2)}\}$ as $n \to \infty$	Erasure probability
	$\Pr\{\mathcal{N}_e^{(2)}\} \to o(n^{-1})$	$\varepsilon^2+\varepsilon^3(1-\varepsilon)$
	$\Pr\{\mathcal{N}_e^{(2)}\} \to o(n^{-2})$	$\varepsilon(1-(1-\varepsilon)^2)$
	$\Pr\{\mathcal{N}_e^{(2)}\} \to o(n^{-1})$	ε^2

Extension to Graphs with Cycles (cntd)

For a fixed number of iterations and increasing blocklength n fewer and fewer cycles occur in the corresponding computation graphs.
In the limit of infinitely long blocklengths the computation graph becomes a tree with probability one.

Theorem: For a given degree distribution pair (λ, ρ) consider the sequence of associated ensembles LDPC(n, λ, ρ) for increasing blocklength n under ℓ rounds of BP decoding. Then, the average probability of erasure at the k-th iteration $\mathrm{E}_{\mathrm{LDPC}(n,\lambda,\rho)}\{\mathrm{P}_e(\mathcal{C}, \varepsilon, k)\}$ converges to $z^{(k)} = z^{(k)}(\varepsilon)$ (average erasure probability for cycle free neighborhood) as $n \to \infty$, i.e.

$$\lim_{n\to\infty} \mathrm{E}_{\mathrm{LDPC}(n,\lambda,\rho)}(\mathrm{P}_e(\mathcal{C}, \varepsilon, k)) = z^{(k)}.$$

Density Evolution Properties

Lemma (monotonicity): For a given degree distribution pair (λ, ρ) **define** $f(\varepsilon, z) = \varepsilon\lambda(1 - \rho(1 - z))$. **Then,** $f(\varepsilon, z)$ **is increasing in both its arguments for** $z \in [0, 1]$

Lemma (monotonicity with respect to the channel): Let (λ, ρ) **be a degree distribution pair,** $\varepsilon \in [0, 1]$ **and** $z^{(k)}_{(\lambda,\rho)}(\varepsilon) = f(\varepsilon, z^{(k)}_{(\lambda,\rho)}(\varepsilon))$. **If** $z^{(k)}_{(\lambda,\rho)}(\varepsilon) \to 0$ **for** $k \to \infty$ **then**

$$z^{(k)}_{(\lambda,\rho)}(\varepsilon)) \to 0 \qquad \text{for all } \varepsilon' < \varepsilon$$

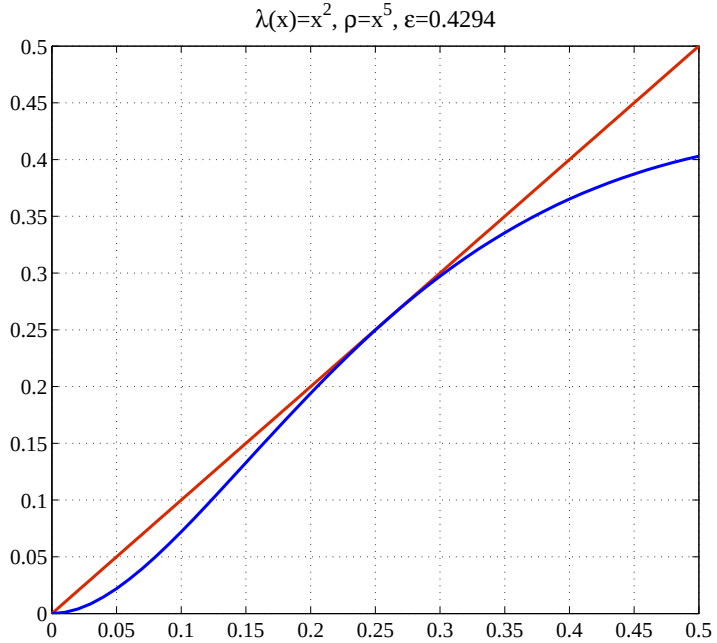
Extreme values

$$z^{(k)}_{(\lambda,\rho)}(\varepsilon = 0) = 0 \qquad \text{and} \qquad z^{(k)}_{(\lambda,\rho)}(\varepsilon = 1) = 1$$

Definition: The erasure probability threshold $\varepsilon^*(\lambda, \rho)$ **associated with the degree distribution pair** (λ, ρ) **is defined as**

$$\varepsilon^*(\lambda, \rho) = \sup\{\varepsilon \in [0, 1] : \lim_{k\to\infty} z^{(k)}_{(\lambda,\rho)}(\varepsilon) = 0\}$$

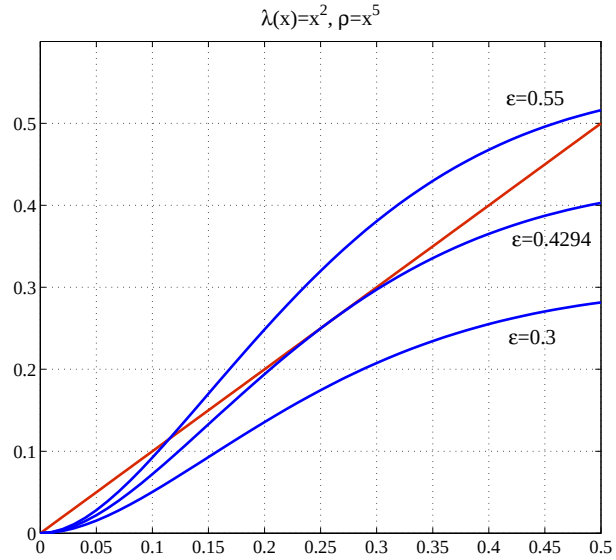
Density Evolution Properties (cntd)



Interpretation: Using an ensemble LDPC(λ, ρ) of sufficient long length we can transmit reliably over the channel BEC(ε) if $\epsilon < \epsilon^*(\lambda, \rho)$ while for $\epsilon > \epsilon^*(\lambda, \rho)$ no reliable transmission is possible.

Graphical Determination of the Threshold

Theorem: For a given degree distribution pair (λ, ρ) **and** $\varepsilon \in [0,1]$ **let** $f(\epsilon, z) = \epsilon\lambda(1-\rho(1-z))$. **The threshold can be characterized by the fixed point equation** $z = f(\varepsilon, z)$ **as follows**



$$(i) \qquad \varepsilon^*(\lambda,\rho) = \sup\{\varepsilon \in [0,1] : \ z = f(\varepsilon, z) \text{ has no solution in } [0,1]\}$$

$$(ii) \qquad \varepsilon^*(\lambda,\rho) = \inf\{\varepsilon \in [0,1] : \ z = f(\varepsilon, z) \text{ has a solution in } [0,1]\}$$

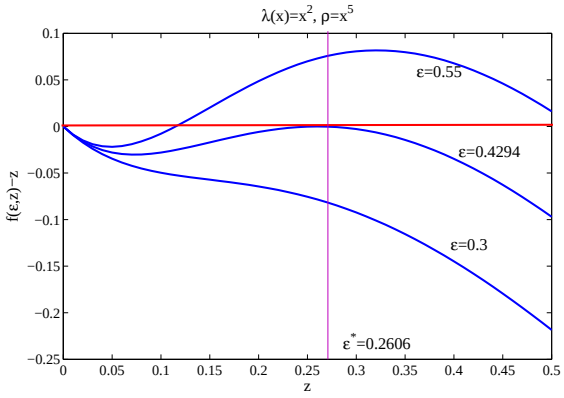
Graphical Determination of the Threshold: Critical Point

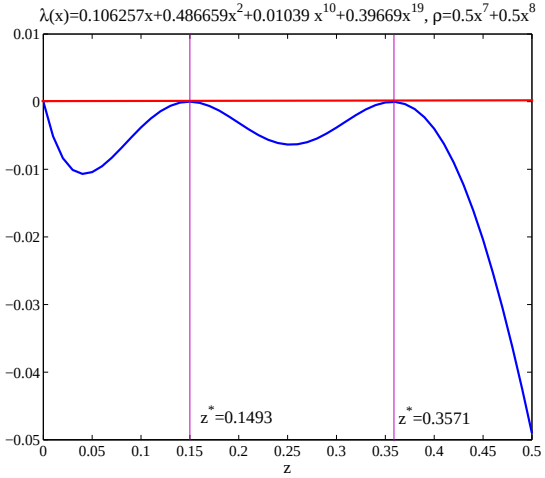
Given a degree distribution pair (λ, ρ) which has threshold $\varepsilon^* = \varepsilon^*(\lambda, \rho)$ we say that z^* is a critical point if

$$f(\varepsilon^*, z^*) = z^* \text{ and } \left.\frac{\partial f(\varepsilon^*, z)}{\partial z}\right|_{z=z^*} = 1 \qquad (3)$$

i.e. z^* is the point or one of the points at which $f(\varepsilon^*, z) - z$ tangently touches the horizontal axis.

- **(3) is a system of two equations in two unknowns;**
- **The solution (ε^*, z^*) of (3) is not unique;**
- **Capacity is achieved when z is a critical point in the whole range $z \in [0, \varepsilon^*]$ (matching condition).**
- **The closer we get to the capacity the more critical points we expect to see.**





Exit Charts

Given a degree distribution pair (λ, ρ) and the BEC(ε), $f(\varepsilon, z) = \varepsilon\lambda(1 - \rho(1 - z))$ describes the evolution of the fraction of erased messages emitted by a variable node.

Let us represent $f(\varepsilon, z)$ as the composition of two functions:

- **a function which represents the action of variable nodes**
- **a function which represents the action of check nodes**

Define $\nu_\varepsilon(z) = \varepsilon\lambda(z)$ $\qquad$ $c(z) = 1 - \rho(1 - z)$

Then $f(\varepsilon, z) = \nu_\varepsilon(c(z))$

Convergence condition $f(\varepsilon, z) < z \quad \Rightarrow \quad \nu_\varepsilon(c(z)) < z$

Note that $\nu_\varepsilon(z)$ has an inverse for $z > 0$ since $\lambda(x)$ is a polynomial with positive coefficients. The convergence condition yields

$$c(z) < \nu_\varepsilon^{-1}(z) \qquad z \in [0, 1] \tag{4}$$

Interpretation:
- **$c(z)$ has to lie strictly below $\nu_\varepsilon^{-1}(z)$**
- **The threshold ε^* is the supremum of all $\varepsilon \in [0, 1]$ for which (4) is satisfied.**

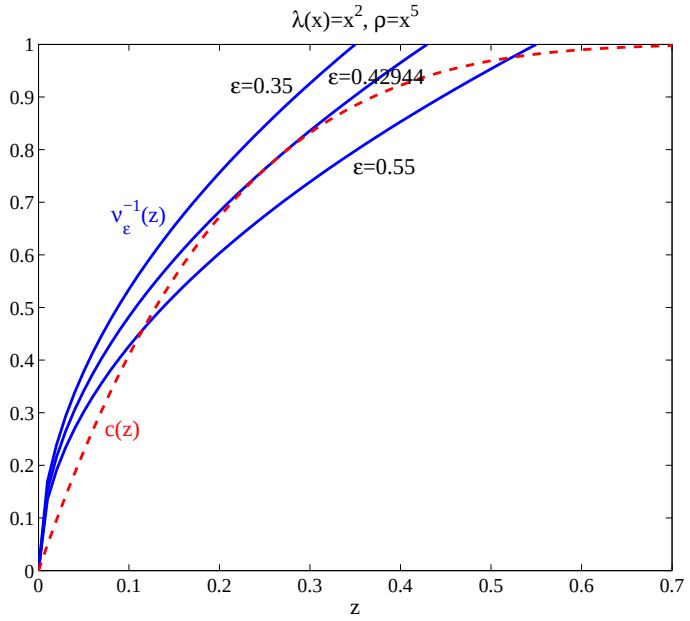
Exit Charts: Examples

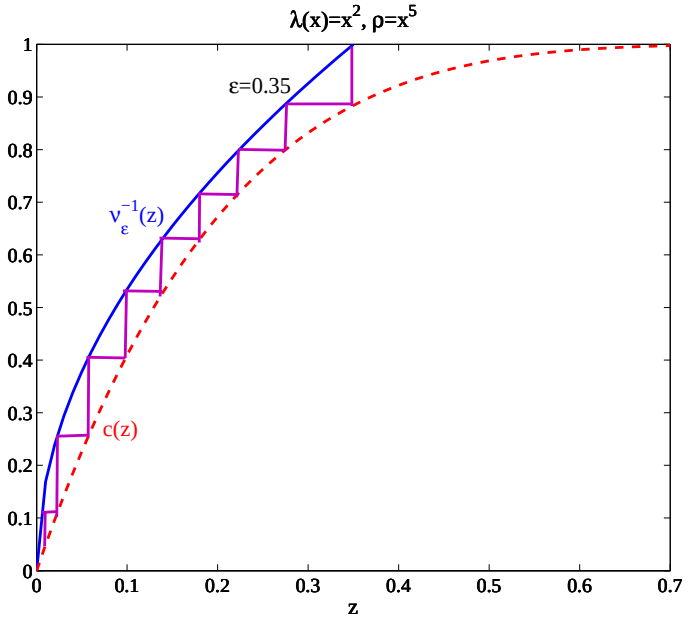
Assume $\lambda(x) = x^2$ and $\rho(x) = x^5$

Inverse of $\nu_\varepsilon(z) = \varepsilon z^2$: $z = \sqrt{\frac{\nu_\varepsilon(z)}{\varepsilon}} \Rightarrow \nu_\varepsilon^{-1}(z) = \sqrt{\frac{z}{\varepsilon}}$

$$c(z) = 1 - (1-z)^5$$

The convergence condition is $1 - (1-z)^5 < \sqrt{\frac{z}{\varepsilon}}$





LDPC Ensemble Optimization by Linear Programming

- **Problem 1: For a given code rate determine the degree distribution pair (λ, ρ) that maximizes the threshold ε^*.**
- **Problem 2: For a given threshold ε^* find (λ, ρ) that maximizes the design rate.**

Solution of Problem 2

- **Fix $\rho(x)$. Since the design rate is given by $r(\lambda, \rho) = 1 - \frac{\int_0^1 \rho(x)\mathrm{d}x}{\int_0^1 \lambda(x)\mathrm{d}x}$ to maximize $r(\lambda, \rho)$ we have to maximize**

$$\int_0^1 \lambda(x)\mathrm{d}x = \sum_i \frac{\lambda_i}{i} \tag{5}$$

LDPC Ensemble Optimization by Linear Programming (cntd)

- **For given ε^* (5) has to be maximized subject to the constraint**

$$\sum_i \lambda_i (1-\rho(1-z))^{i-1} < \frac{z}{\varepsilon^*} \qquad \forall z \in z \in [0, \varepsilon^*] \tag{6}$$

Given q values of z, $0 \le z_1 \le z_2 \le \ldots \le z_q \le \varepsilon^*$ (6) yields q constraints

$$\sum_i \lambda_i (1-\rho(1-z_j))^{i-1} \le \frac{z_j}{\varepsilon^*} \qquad j = 1, \ldots q.$$

- **Enforce the additional constraints $\lambda_i > 0$ for all i.**
- **Enforce the property $\sum_i \lambda_i = 1$**

The optimization of the LDPC ensemble can be solved by linear programming!

References

- T. Richardson and R. Urbanke, Modern Coding Theory, free at \lthcwww.epfl.ch\mct\index.php
- G. Caire, Lecture notes for the course on Information Theory and Coding, 2003.