Graph Covers and Iterative Decoding of Finite-Length Codes

Pascal O. Vontobel (CSL, UIUC)

Ralf Koetter (CSL, UIUC)

Talk at DIMACS, Piscataway, NJ, USA Dec. 15, 2003

Table of Contents

- MAP/ML decoding vs. message-passing decoding
- A simple example
- Graph covers
- Pseudo-codewords, pseudo-weight
- Bounds on the minimum pseudo-weight
- Conclusions/Outlook

MAP/ML Decoding Algorithm (Part 1)

Assume that the codeword $\mathbf{x} \in C$ was sent, the word $\mathbf{y} \in \mathcal{Y}^n$ was received, and based on $\mathbf{y}$ we would like to find the "most likely" transmitted codeword $\hat{\mathbf{x}}$. An algorithm that performs the above task is called a decoding algorithm.

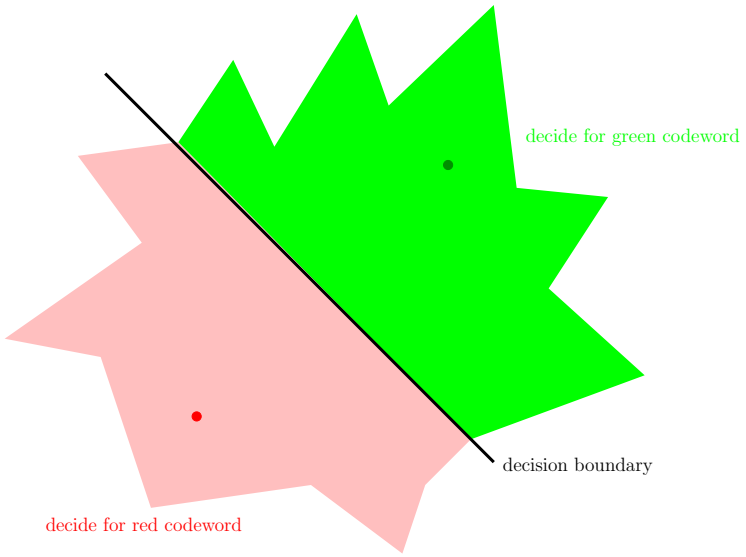
- Symbol-wise MAP decoding gives

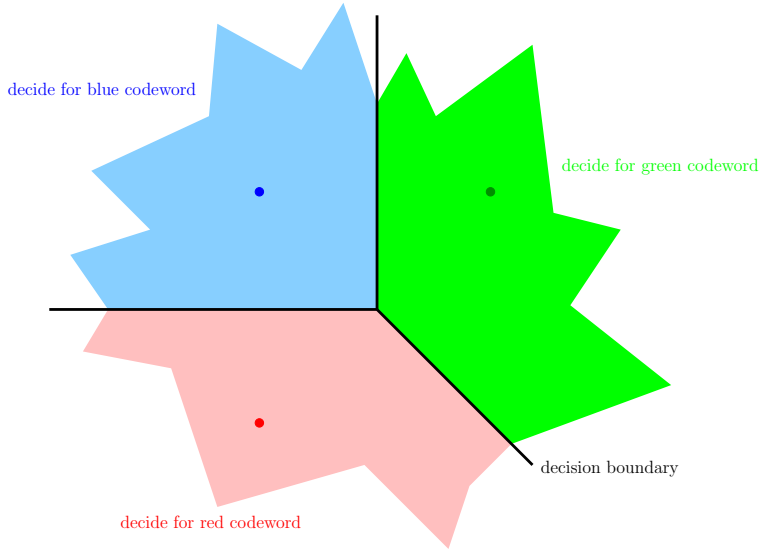
$$\hat{x}_i(\mathbf{y}) = \underset{x_i \in \mathcal{X}}{\operatorname{argmax}} P_{X_i|\mathbf{Y}}(x_i|\mathbf{y}) \quad \text{(for each } i = 1, \ldots, n\text{)}.$$

- Block-wise MAP decoding gives

$$\hat{\mathbf{x}}(\mathbf{y}) = \underset{\mathbf{x} \in C}{\operatorname{argmax}} P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}).$$

MAP/ML Decoding Algorithm (Part 2)





Left-hand side: MAP (ML) decision regions for a codebook with two codewords.

Right-hand side: MAP (ML) decision regions for a codebook with three codewords.

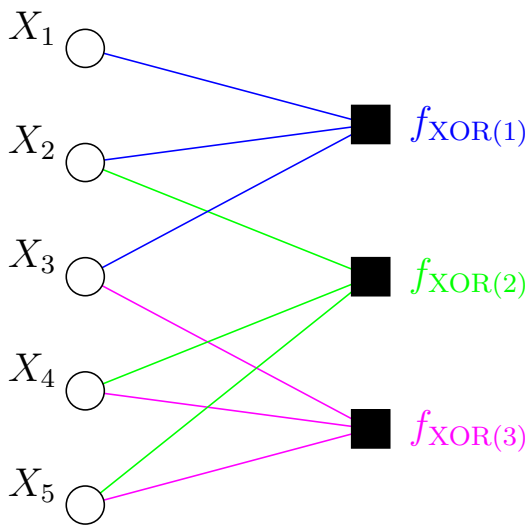
These are decision regions (where the axes are log-likelihoods of the symbols) for block-wise MAP decoding, under the assumption that all codewords are equally likely.

Based on the Hamming distances of the codewords we can calculate the distances to the decision boundaries.

Tanner/Factor Graph of an LDPC Code

Example:

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



This factor/Tanner graph has cylces of length four, six, and eight.

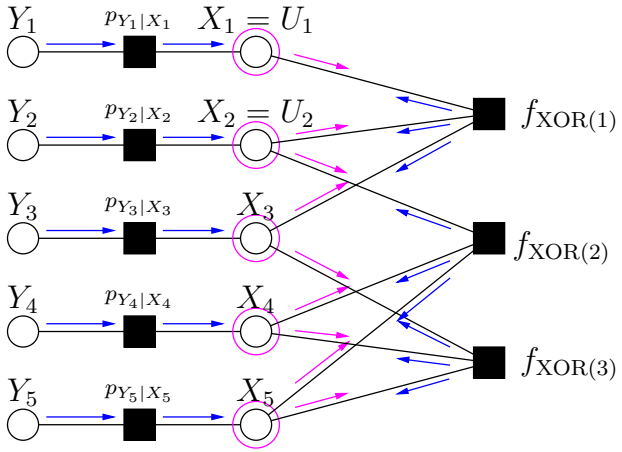
LDPC codes in general:

- An LDPC code has a matrix with very few ones.
- (j,k)-regular LDPC code: all bit nodes have degree j and all check nodes have degree k. Equivalently, $\mathbf{H}$, has uniform column weight j and uniform row weight k.
- One can show that factor/Tanner graphs of good codes have cycles (under the assumption of bounded state-space sizes).

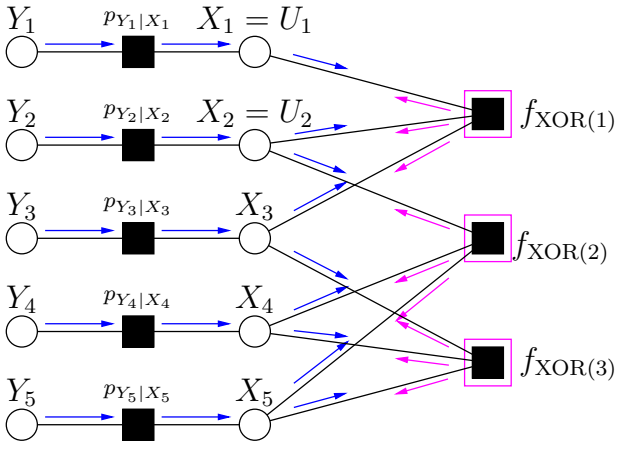
Message-Passing Decoding Algorithms

For interesting code sizes, the above MAP/ML decoding procedures are intractable, therefore we need low-complexity, sub-optimal algorithms: message-passing algorithms are such a class of decoding algorithms.

i-th iteration



$i.5$-th iteration



A message-passing algorithm

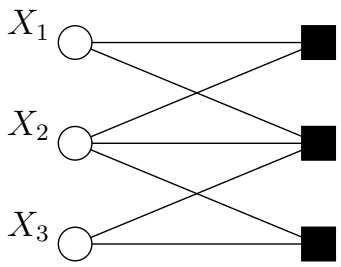
- sends messages along the edges,
- does processing of the messages at the vertices.

Note: all operations are performed locally!

Analysis of Message-Passing Algorithms in finite length graphs

- Wiberg (1996): pseudo-codewords, computation tree, pseudo-weight, deviation set
- Horn (1999): pseudo-codewords, cycle codes
- Forney et al. (2001): extensions to other channels, tail-biting-trellis
- Frey et al. (2001): signal-space interpretation of iterative decoding
- Di et al. (2002): stopping sets, erasure channel
- MacKay et al. (2002): near codewords
- Tian et al. (2002): extrinsic message degree
- Feldman (2003): linear progamming decoding
- Richardson (2003): trapping sets
- enormous anecdotal evidence ...

A Simple Example (Part 1)

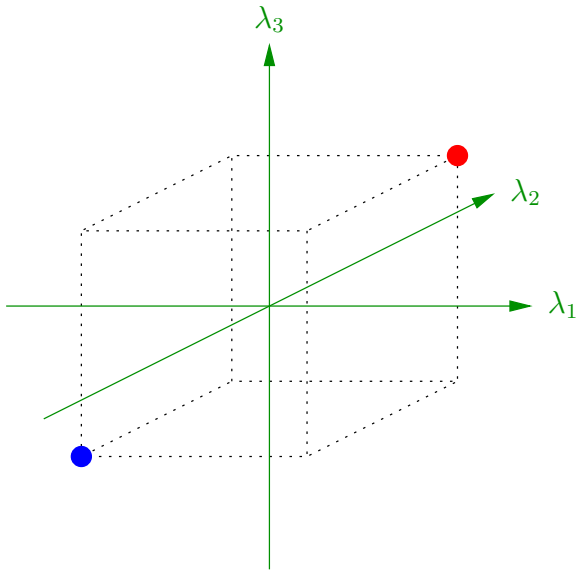


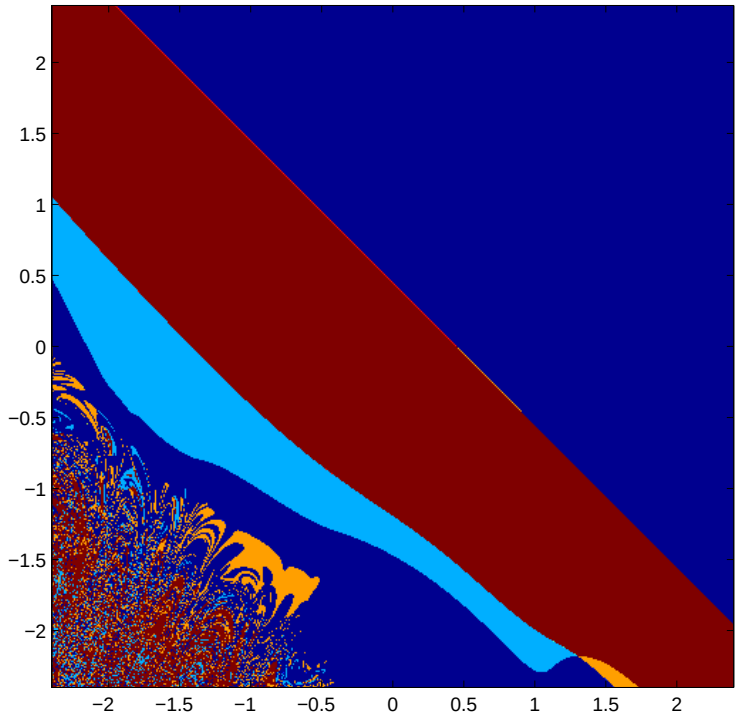
We consider the (trivial) binary linear $[3,0,\infty]$ code C with parity-check matrix

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

- Obviously, $\mathrm{C} = \{(0,0,0)\}$.
- Symbol-wise MAP decoding: yields always $\hat{x}_1 = 0$, $\hat{x}_2 = 0$, $\hat{x}_3 = 0$ (independent of $\mathbf{y}$).
- Block-wise MAP decoding: yields always $\hat{\mathbf{x}} = (0,0,0)$ (independent of $\mathbf{y}$).

A Simple Example (Part 2)

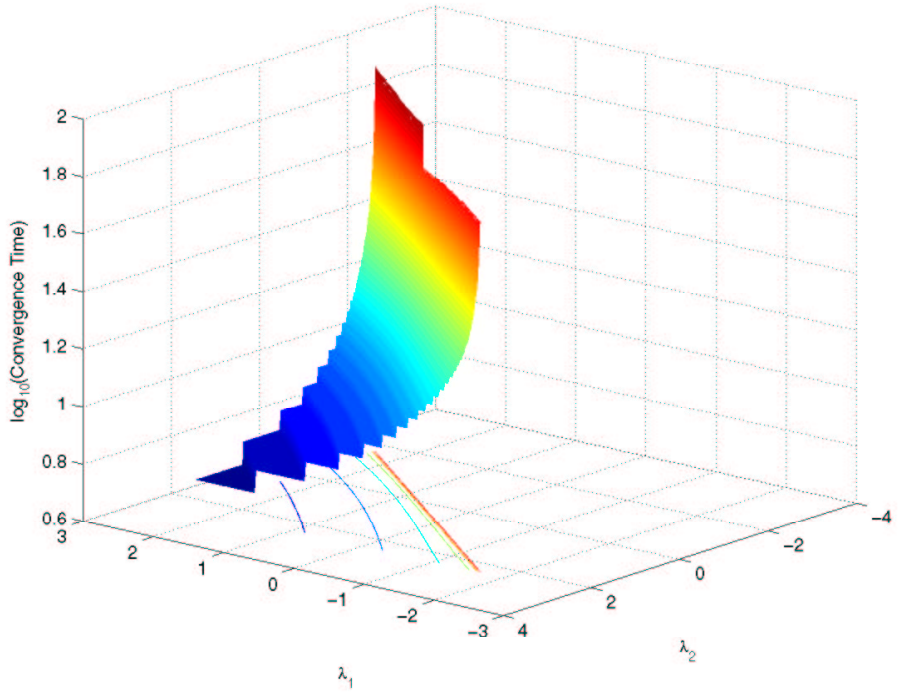




The plot shows the decision regions when using the sum-product algorithm for the trivial code (here, $\lambda_3 = -0.45$). As can be seen, the decision region for $\hat{x}_1 = 0$, $\hat{x}_2 = 0$, $\hat{x}_3 = 0$ seems to be described by

$$\lambda_1 + \lambda_2 + \lambda_3 > 0.$$

A Simple Example (Part 3)

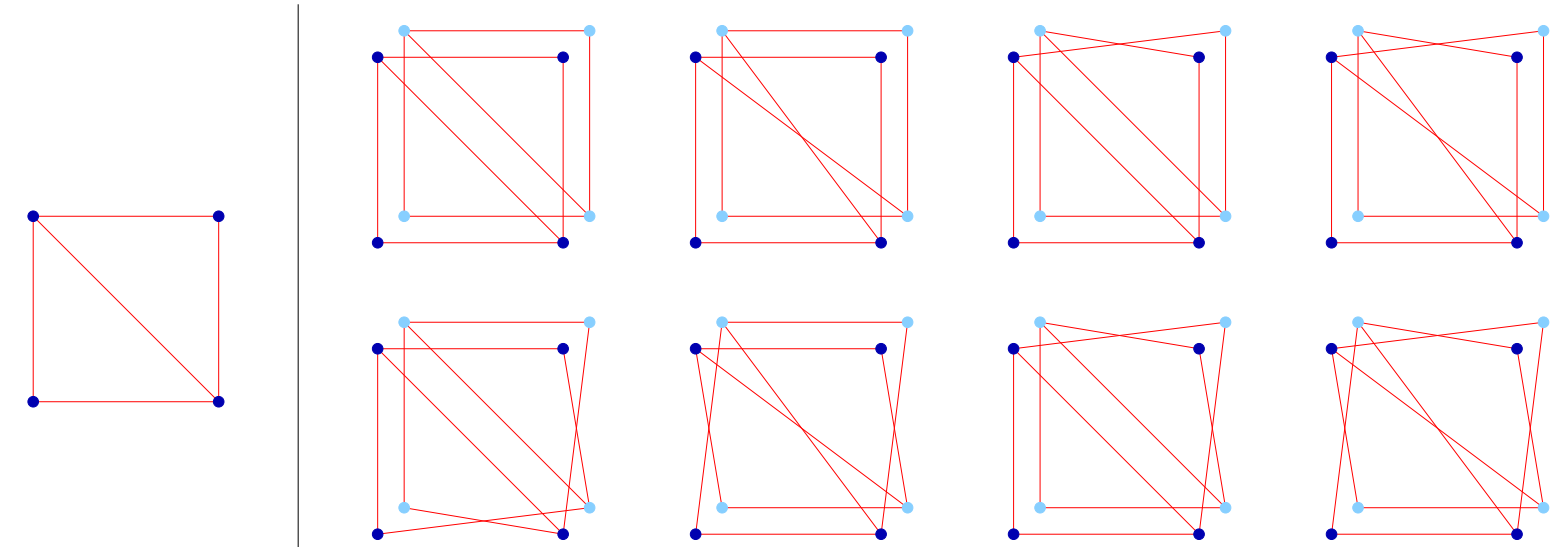


The plot shows the convergence time when using the sum-product algorithm for the trivial code (here, $\lambda_3 = -0.45$). As can be seen, the convergence time increases towards the plane

$$\lambda_1 + \lambda_2 + \lambda_3 = 0.$$

The message-passing decoding algorithm behaves as if code C were a repetition code. But where is the all-ones word in the decoding? Before we continue to give an interpretation of these results, we have to introduce graph covers ...

Graph Covers (Part 1)



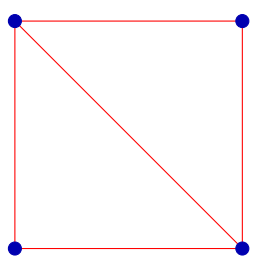
original graph

sample of possible double covers of the original graph

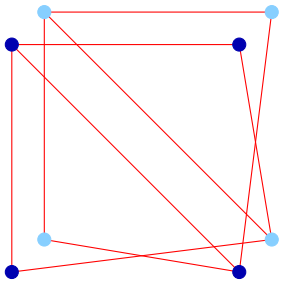
Definition: A double cover of a graph is ...

Note: the above graph has $2! \cdot 2! \cdot 2! \cdot 2! \cdot 2! = 32$ double covers.

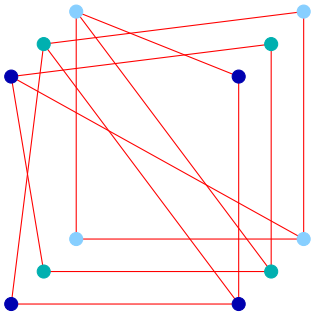
Graph Covers (Part 2)



original graph



(a possible)
double cover of
the original graph

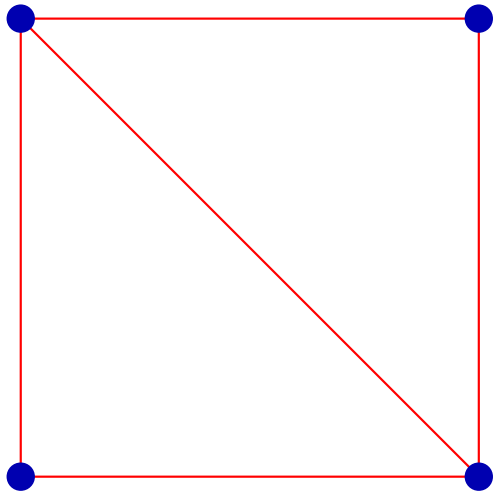


(a possible)
triple cover of
the original graph

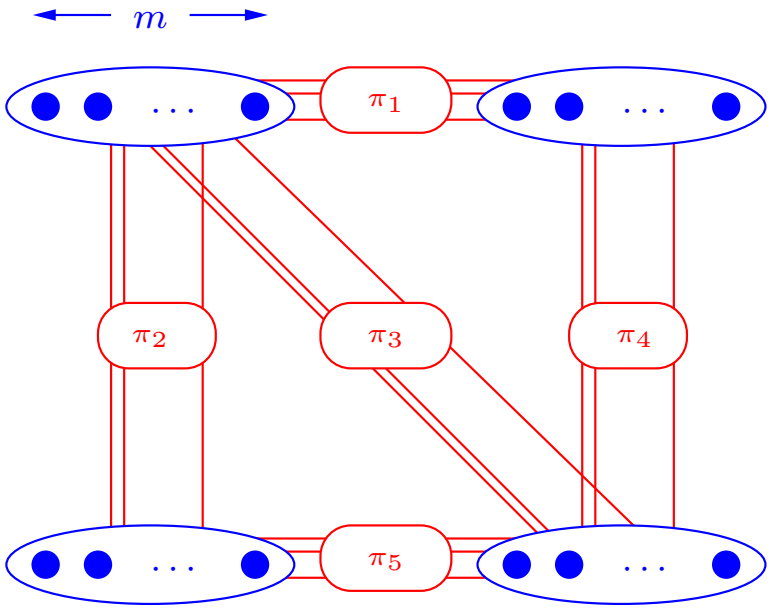
. . .

Besides double covers, a graph also has many triple covers, quadruple covers, quintuple covers, etc.

Graph Covers (Part 3)



original graph

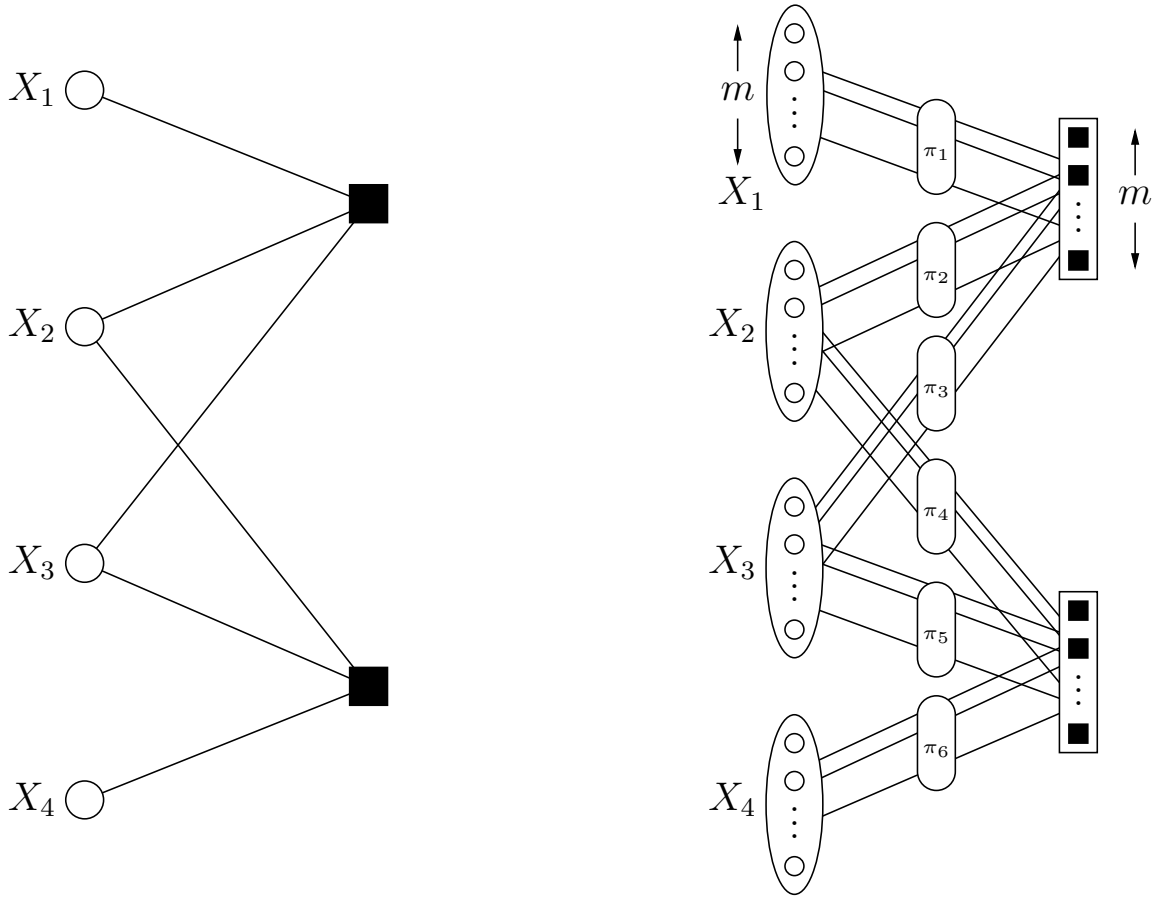


(possible)
m-fold cover of
original graph

An m-fold cover is also called a cover of degree m. Do not confuse this degree with the degree of a vertex!

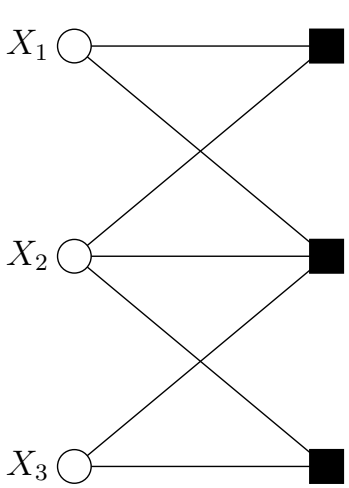
Note: there are many possible m-fold covers of a graph.

Factor-Graph Covers (Part 1)

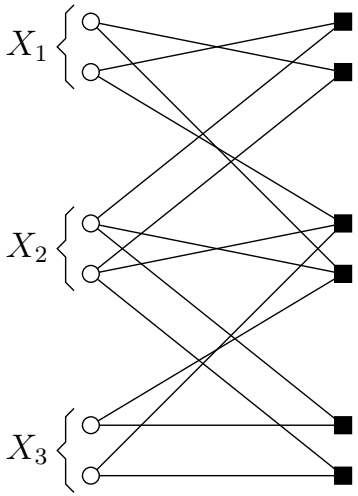


Similarly to graph covers, we can also define factor graph covers.

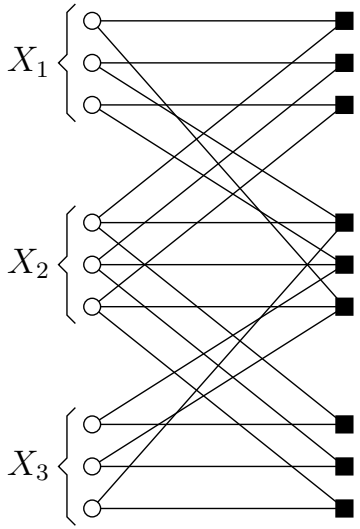
A Simple Example (Part 4)



original
factor graph



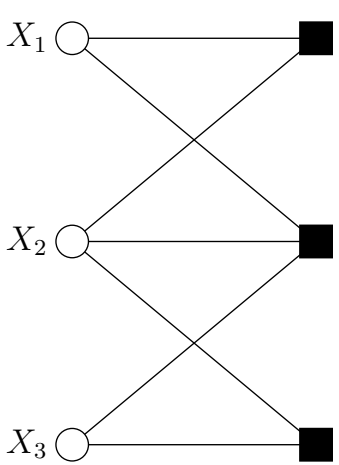
(a possible)
double cover of the
original factor graph



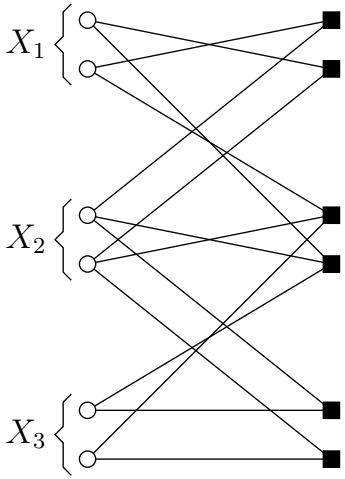
(a possible)
triple cover of the
original factor graph

The figure shows a (possible) double cover and a (possible) triple cover of the original factor graph.

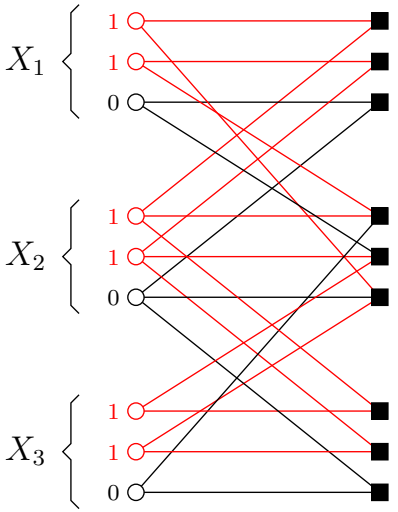
A Simple Example (Part 5)



original
factor graph



(a possible)
double cover of the
original factor graph



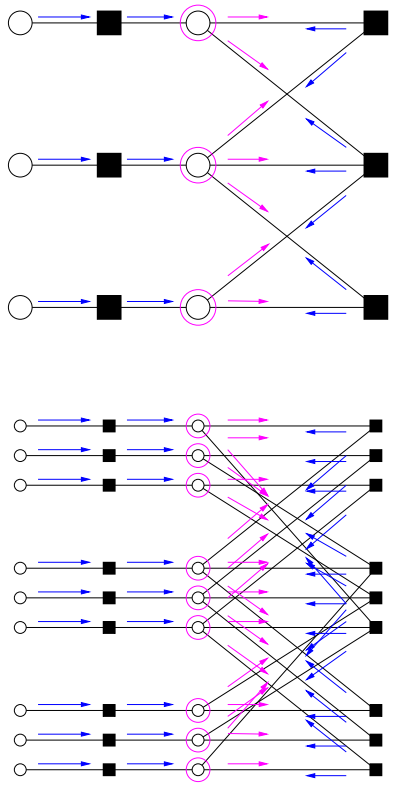
(a possible)
triple cover of the
original factor graph

The figure shows a (possible) double and a (possible) triple cover of the original factor graph.

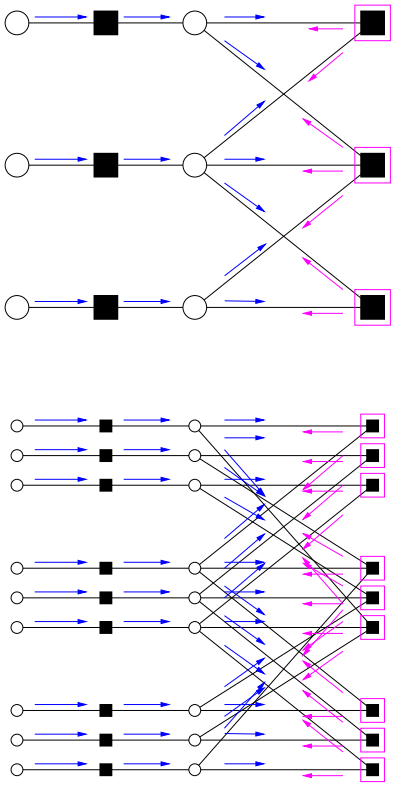
Assume that $\lambda_1 + \lambda_2 + \lambda_3 < 0$. Then the indicated (valid) configuration in the triple cover has a larger likelihood than the the all-zeros configuration.

A Simple Example (Part 6)

i-th iteration



$i.5$-th iteration



Why do factor graph covers matter? Well, a locally operating decoding algorithm cannot distinguish if it is decoding on the original factor graph or on any of its covers.

The messages in the triple cover factor graph correspond to three identical copies of the messages in the original factor graph.

Factor-Graph Covers (Part 2)

Two questions:

- What is the influence of a valid configuration of a finite cover upon the decoding behavior?

$$\Longrightarrow \text{Pseudo-weight}$$

- How do we characterize all the valid configurations from all the finite covers?

$$\Longrightarrow \text{Pseudo-codewords}$$
$$\Longrightarrow \text{Fundamental polytope / fundamental cone}$$

Valid Configurations in Factor Graph Covers (Part 1)

- We are looking at the factor graph of a code C of length n. We assume that all codewords are equally likely.
- We assume to have an m-fold cover of the factor graph. The valid configurations of this factor graph cover form a code $\tilde{\mathrm{C}}$ with codewords of length $m \cdot n$.
- Let $\tilde{\mathbf{0}}$ be the lifting of $\mathbf{0}$ to the cover.
- Let $\tilde{\mathbf{x}}$ be a (valid) configuration in the cover.
- Let $\tilde{\mathbf{y}}$ be the lifting of $\mathbf{y}$ to the cover, i.e. $\tilde{y}_{i,\ell} \stackrel{\triangle}{=} y_i$.
- Let $\lambda_i \stackrel{\triangle}{=} \log \frac{P_{Y_i|X_i}(y_i|0)}{P_{Y_i|X_i}(y_i|1)}$ be the i-th log-likelihood ratio.

We calculate

$$\log \frac{P_{\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{0}})}{P_{\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{x}})} = \sum_{i=1}^{n}\sum_{\ell=1}^{m} \log \frac{P_{Y_i|X_i}(\tilde{y}_{i,\ell}|0)}{P_{Y_i|X_i}(\tilde{y}_{i,\ell}|\tilde{x}_{i,\ell})} = \sum_{i=1}^{n} \left|\left\{\ell \mid \tilde{x}_{i,\ell} = 1\right\}\right| \cdot \lambda_i .$$

Valid Configurations in Factor Graph Covers (Part 2)

We see that all we need to know is how often the variables in a cover assume the value 1 or 0. Therefore, we define

$$\omega_i(\tilde{\mathbf{x}}) \stackrel{\triangle}{=} \frac{\left|\left\{\ell \mid \tilde{x}_{i,\ell} = 1\right\}\right|}{m}, \qquad \boldsymbol{\omega}(\tilde{\mathbf{x}}) \stackrel{\triangle}{=} \big(\omega_1(\tilde{\mathbf{x}}), \omega_2(\tilde{\mathbf{x}}), \ldots, \omega_n(\tilde{\mathbf{x}})\big),$$

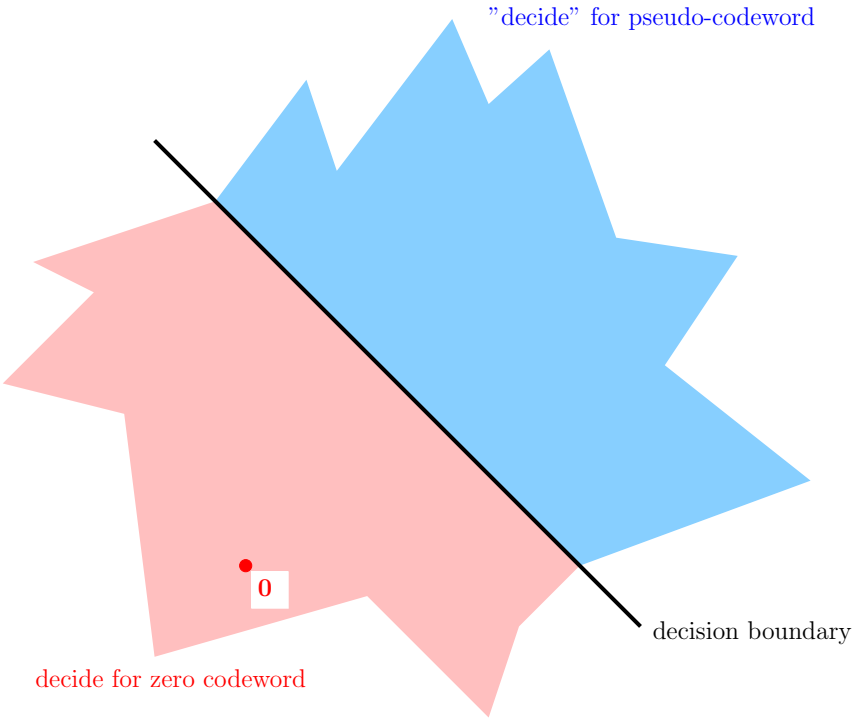
and obtain

$$\begin{aligned}
\log \frac{P_{\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{0}})}{P_{\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{x}})} &= \sum_{i=1}^{n} \left|\left\{\ell \mid \tilde{x}_{i,\ell} = 1\right\}\right| \cdot \lambda_i = m \cdot \sum_{i=1}^{n} \frac{\left|\left\{\ell \mid \tilde{x}_{i,\ell} = 1\right\}\right|}{m} \cdot \lambda_i \\
&= m \cdot \sum_{i=1}^{n} \omega_i(\tilde{\mathbf{x}}) \cdot \lambda_i \\
&\propto \left\langle \boldsymbol{\omega}(\tilde{\mathbf{x}}), \boldsymbol{\lambda} \right\rangle.
\end{aligned}$$

The vector $\boldsymbol{\omega}(\tilde{\mathbf{x}})$ gives the information what influence the configuration $\tilde{\mathbf{x}}$ (that lives in the cover factor graph) has when competing against the all-zeros codeword. We call $\boldsymbol{\omega}(\tilde{\mathbf{x}})$ a pseudo-codeword.

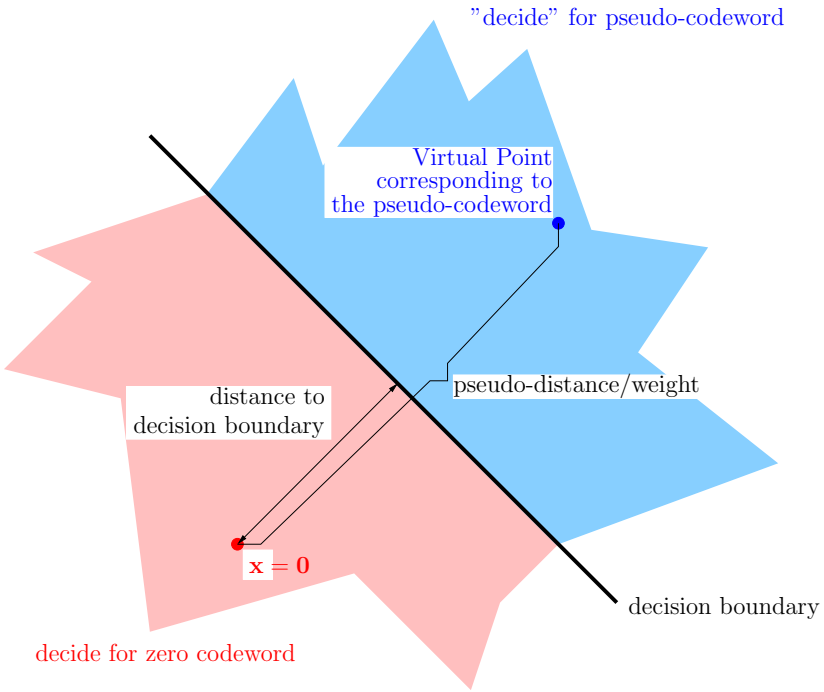
Pseudo-Weight / Pseudo-Distance (Part 1)

Assume, that only the zero codeword $\mathbf{0}$ and the pseudo-codeword $\boldsymbol{\omega}(\tilde{\mathbf{x}})$ are competing against each other.



$$\log \frac{P_{\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{0}})}{P_{\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{x}})} > 0 \quad \Longleftrightarrow \quad \langle \boldsymbol{\omega}(\tilde{\mathbf{x}}), \boldsymbol{\lambda} \rangle > 0 \quad \overset{\text{AWGNC}}{\Longleftrightarrow} \quad \langle \boldsymbol{\omega}(\tilde{\mathbf{x}}), \mathbf{y} \rangle > 0$$

Pseudo-Weight / Pseudo-Distance (Part 2)



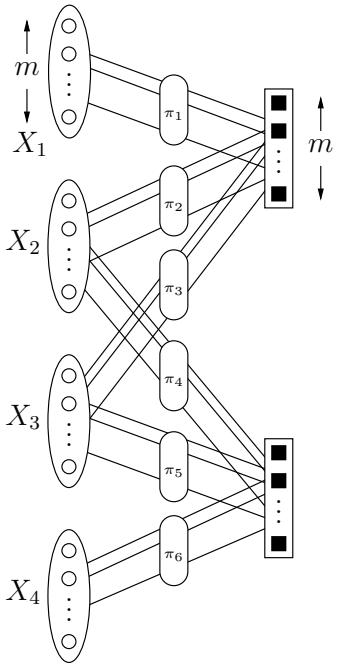
Based on the distance to the decision boundary we introduce a virtual point corresponding to the pseudo-codeword. The "distance"/weight of the virtual point is measured by the pseudo-"distance"/weight

$$w_{\mathrm{p}}^{\mathrm{AWGNC}}(\boldsymbol{\omega}) \triangleq \frac{||\boldsymbol{\omega}||_1^2}{||\boldsymbol{\omega}||_2^2} = \frac{(|\omega_1| + \cdots + |\omega_n|)^2}{|\omega_1|^2 + \cdots + |\omega_n|^2} = \frac{(\omega_1 + \cdots + \omega_n)^2}{\omega_1^2 + \cdots + \omega_n^2}.$$

Pseudo-Codewords (Part 1)

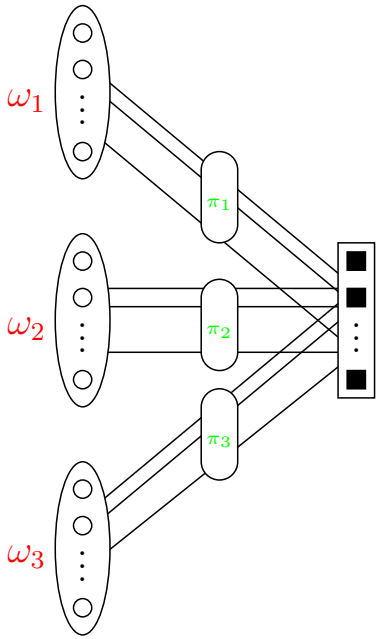
In any *locally operating* message passing algorithm, the set of pseudo-codewords competes with the transmitted codeword for being the "best" solution!

How to characterize the set of pseudo-codewords ω from the union of all degree-m covers for $m = 1, 2, 3, \ldots$?



Pseudo-Codewords (Part 2)

For a typical check we have:



We can find permutations π_1, π_2, π_3 for the tuple ω_1, ω_2, ω_3 if and only if

$$0 \le \omega_1 \le 1$$
$$0 \le \omega_2 \le 1$$
$$0 \le \omega_3 \le 1$$

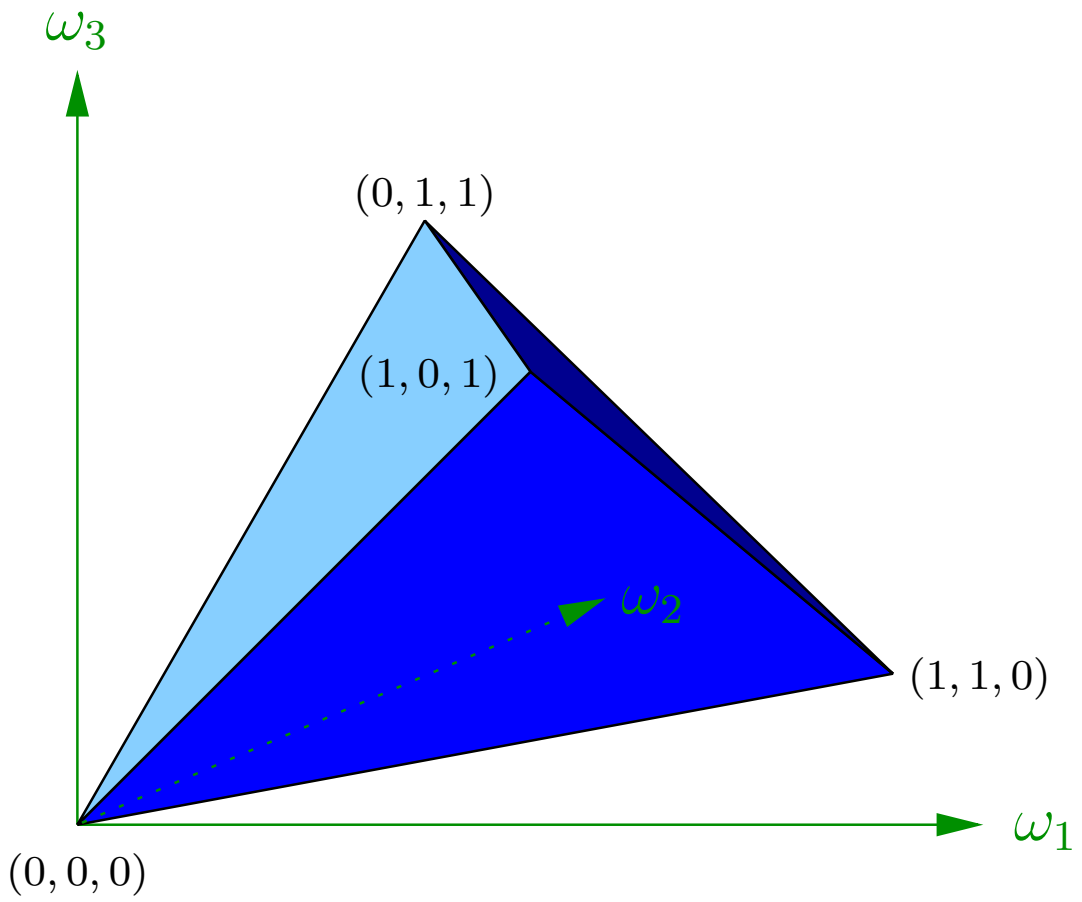
and

$$-\omega_1 + \omega_2 + \omega_3 \ge 0$$
$$+\omega_1 - \omega_2 + \omega_3 \ge 0$$
$$+\omega_1 + \omega_2 - \omega_3 \ge 0$$
$$+\omega_1 + \omega_2 + \omega_3 \le 2$$

or, equivalently,

$$0 \le \omega_i \le 1 \quad \text{and} \quad \max\{\omega_1, \omega_2, \omega_3\} \le \tfrac{1}{2}(\omega_1 + \omega_2 + \omega_3)$$
$$\omega_1 + \omega_2 + \omega_3 \le 2$$

Pseudo-Codewords (Part 3)



The set of all allowed configurations $(\omega_1, \omega_2, \omega_3)$ is called the fundamental polytope.

Pseudo-Codewords (Part 4)

In general, we have that a check of degree δ constrains the set of allowable ω_1, ω_2, ..., ω_δ to values such that

$$\max\{\omega_1,\omega_2,\ldots,\omega_\delta\} \le \frac{1}{2}\sum_{i=1}^{\delta}\omega_i, \qquad \text{(additional affine inequalities)}, \qquad 0 \le \omega_i \le 1.$$

We define an indicator function

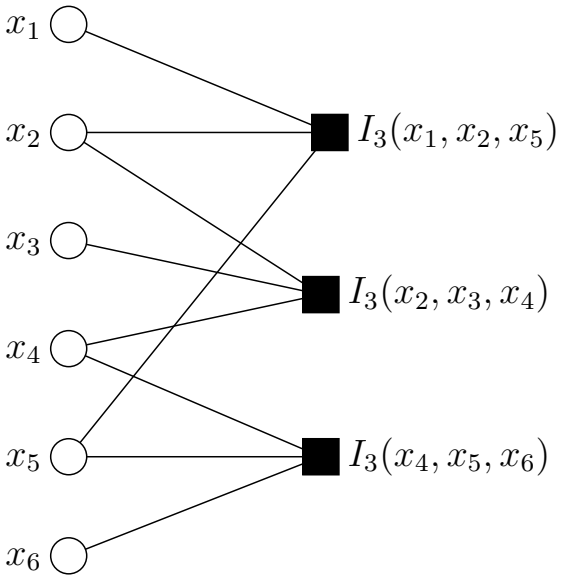
$$\hat{I}_\delta(\omega_1,\omega_2,\ldots,\omega_\delta) = \begin{cases} 1 & \max\{\omega_1,\omega_2,\ldots,\omega_\delta\} \le \frac{1}{2}\sum_{i=1}^{\delta}\omega_i, \text{ (additional affine inequalities),} \\ 0 & \text{otherwise.} \end{cases}$$

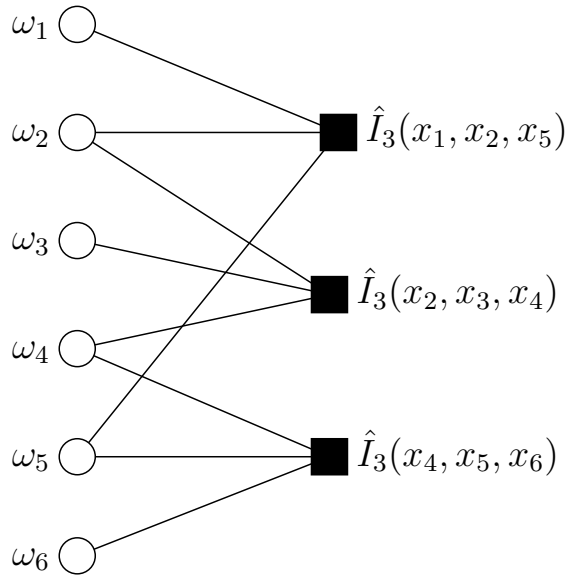
The indicator funtions $\hat{I}_\delta(\omega_1,\omega_2,\ldots,\omega_\delta)$ will allow us to write a facor graph for the pseudo-codeword indicator function.

In order to describe (traditional) codewords, we will use

$$I_\delta(x_1,x_2,\ldots,x_\delta) = \begin{cases} 1 & x_1+x_2+\cdots+x_\delta = 0 \pmod 2, \\ 0 & \text{otherwise.} \end{cases}$$

Pseudo-Codewords (Part 5)





Codeword indicator function:

$$I_3(x_1,x_2,x_5)\cdot I_3(x_2,x_3,x_4)\cdot I_3(x_4,x_5,x_6)$$

Set of codewords:

discrete set of size $2^{\dim(C)}$ in $\mathbb{R}^n$

Remember:

$$x_i \in \{0,1\}$$

Pseudo-codeword indicator function:

$$\hat{I}_3(\omega_1,\omega_2,\omega_5)\cdot \hat{I}_3(\omega_2,\omega_3,\omega_4)\cdot \hat{I}_3(\omega_4,\omega_5,\omega_6)$$

Set of all pseudo-codewords:

dense in the fund. polytope in $\mathbb{R}^n$ that is cut out by the individual indicator functions

Remember:

$$\omega_i \in [0,1]$$

Pseudo-Codewords (Part 6)

For ML/MAP decoding:

minimum Hamming weight / Hamming weight spectrum
is relevant!

For message-passing decoding:

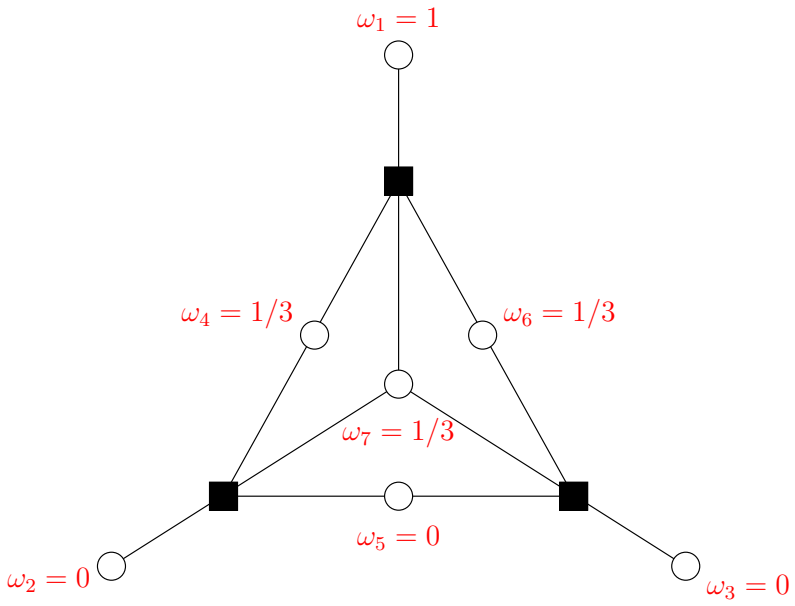
minimum pseudo-weight / pseudo-weight spectrum
is relevant!

Given an LDPC code graph, we therefore want to find the minimum pseudo-weight rather than the minimum Hamming weight!

Note: whereas the minimum Hamming weight is a function of the code, the minimum pseudo-weight is a function of a factor graph that is a realization of the code.

$[7,4,3]$ Hamming Code

We consider a possible factor/Tanner graph realization of the $[7,4,3]$ Hamming code.



The (scaled) pseudo-codeword shown in the above figure is

$$\boldsymbol{\omega} = \left(1 \quad 0 \quad 0 \quad \tfrac{1}{3} \quad 0 \quad \tfrac{1}{3} \quad \tfrac{1}{3}\right).$$

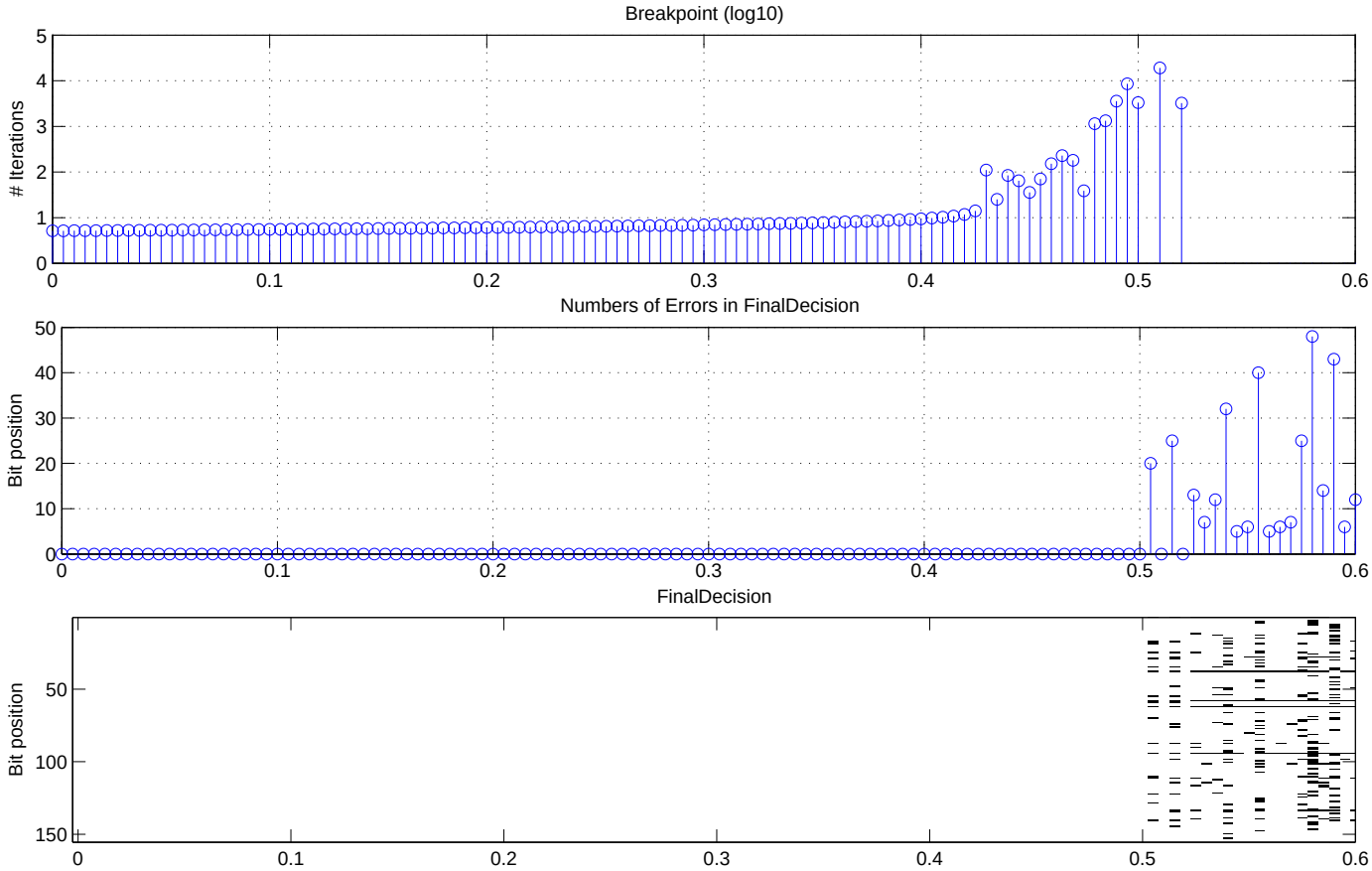
It has pseudo-weight $w_{\mathrm{p}}^{\mathrm{AWGNC}}(\boldsymbol{\omega}) = 3$.

A $[155, 64, 20]$ Code by Tanner (Part 1)

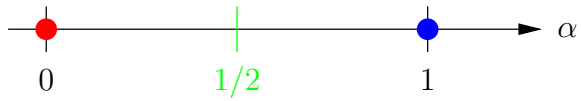
A $(3, 5)$-regular LDPC code constructed by Tanner.

Codelength	155
Rate	$64/155 = 0.4129$
Girth of the factor graph	8 (optimal)
Diameter of the factor graph	6 (optimal)
Minimum Hamming weight	20
Minimum pseudo-weight	$10.8 < w_{\mathrm{p,min}}^{\mathrm{AWGNC}} < 16.4$

A $[155, 64, 20]$ Code by Tanner (Part 2)



The horizontal axis shows the parameter α; $\alpha = 0.5$ corresponds to the hypothetical decision boundary.



Bounds on the Minimum Pseudo-Weight

For a given factor graph of a given code, we would like to find the minimum pseudo-weight, or at least lower and upper bounds for it.

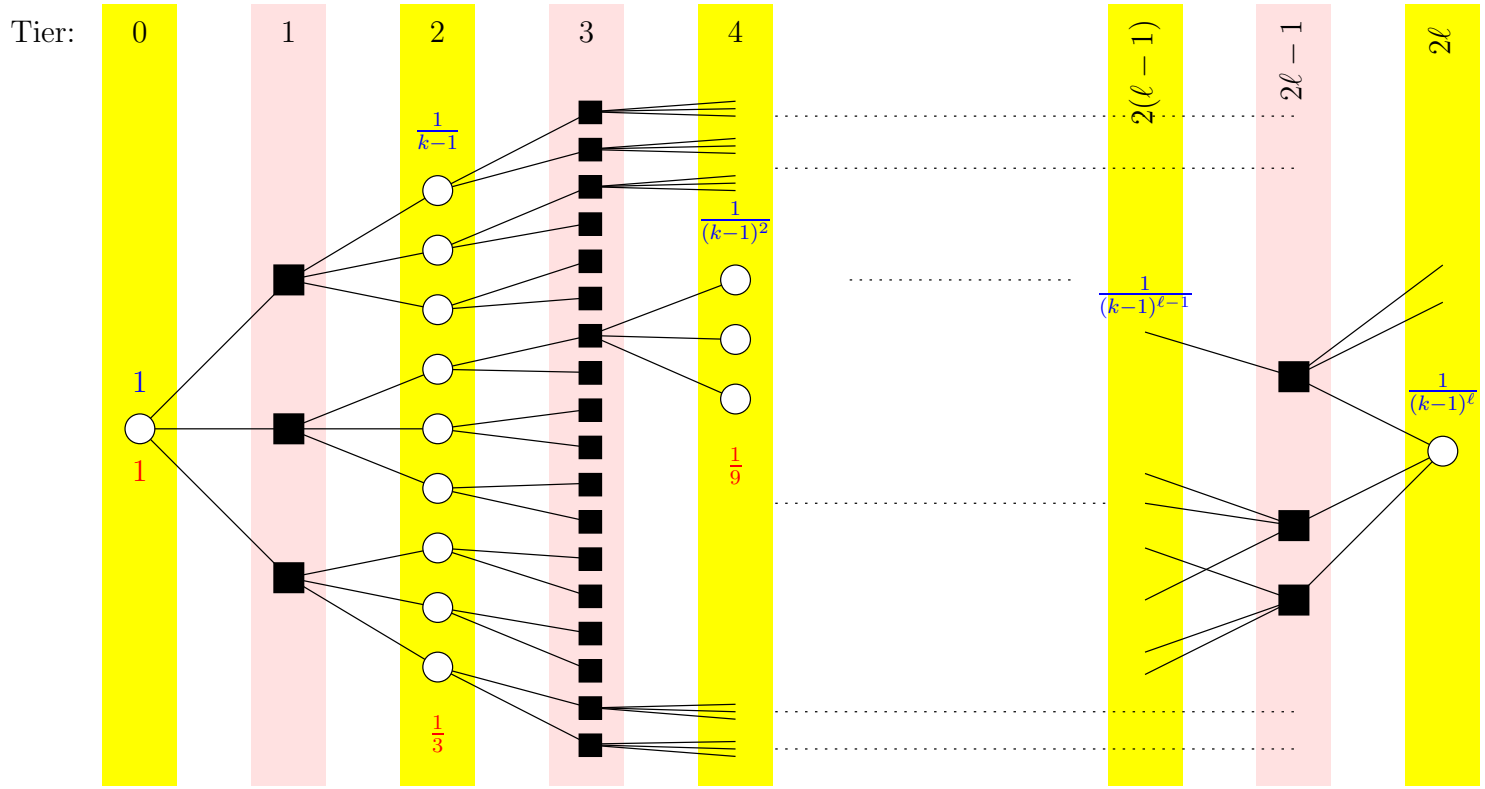
Techniques for obtaining upper bounds on the min. pseudo-weight:

- The pseudo-weight of any valid pseudo-codeword gives an upper bound.
- Canonical completion.

Techniques for obtaining lower bounds on the min. pseudo-weight:

- Bounds based on largest and second largest eigenvalue of $\mathbf{H}^{\mathsf{T}} \cdot \mathbf{H}$.
- Linear programming bounds.

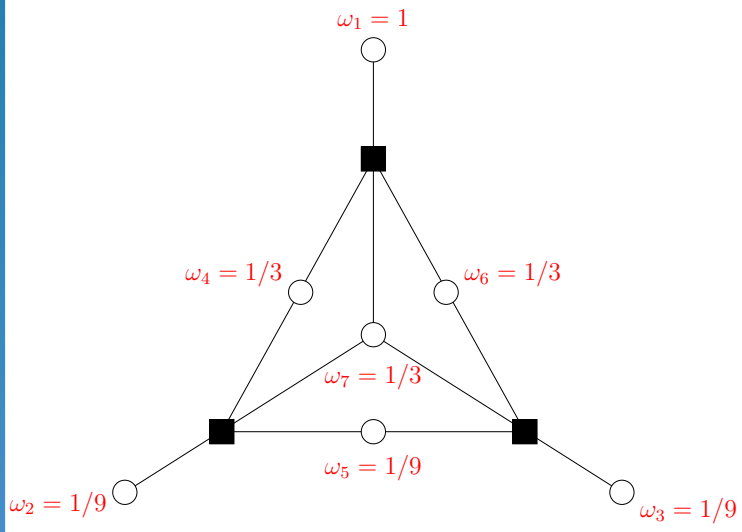
An Upper Bound on the Minimum Pseudo-Weight based on the Canonical Completion (Part 1)



The canonical completion for a $(3,4)$-regular LDPC code. On check-regular graphs the canonical completion always gives a (valid) pseudo-codeword.

An Upper Bound on the Minimum Pseudo-Weight based on the Canonical Completion (Part 2)

Example: $[7,4,3]$ binary Hamming code.



The (scaled) pseudo-codeword of the canonical completion starting at ω_1 is

$$\boldsymbol{\omega} = \begin{pmatrix} 1 & \frac{1}{9} & \frac{1}{9} & \frac{1}{3} & \frac{1}{9} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}.$$

The pseudo-weight of $\boldsymbol{\omega}$ is

$$w_{\mathrm{p}}^{\mathrm{AWGNC}}(\boldsymbol{\omega}) = \frac{||\boldsymbol{\omega}||_1^2}{||\boldsymbol{\omega}||_2^2}$$

$$= \frac{\left(1+\frac{1}{9}+\frac{1}{9}+\frac{1}{3}+\frac{1}{9}+\frac{1}{3}+\frac{1}{3}\right)^2}{1+\frac{1}{81}+\frac{1}{81}+\frac{1}{9}+\frac{1}{81}+\frac{1}{9}+\frac{1}{9}}$$

$$= 3.973.$$

An Upper Bound on the Minimum Pseudo-Weight based on the Canonical Completion (Part 3)

Theorem: Let C be a (j,k)-regular LDPC code with $3 \leq j < k$. Then the minimum pseudo-weight is upper bounded by

$$w_{\text{p,min}}^{\text{AWGNC}}(\mathsf{C}) \leq \beta'_{j,k} \cdot n^{\beta_{j,k}},$$

where

$$\beta'_{j,k} = \left(\frac{j(j-1)}{j-2} \right)^2, \quad \beta_{j,k} = \frac{\log\left((j-1)^2\right)}{\log\left((j-1)(k-1)\right)} < 1.$$

Corollary: The minimum relative pseudo-weight for any sequence $\{\mathsf{C}_i\}$ of (j,k)-regular LDPC codes of increasing length satisfies

$$\lim_{n \to \infty} \left(\frac{w_{\text{p,min}}^{\text{AWGNC}}(\mathsf{C}_i)}{n} \right) = 0.$$

A Lower Bound on the Minimum Pseudo-Weight based on Eigenvalues

Let C be a (j,k)-regular code of length n.

- Let $\mathbf{H}$ be the parity-check matrix.
- We assume that the corresponding factor/Tanner graph has one component.
- Let $\mathbf{L} \stackrel{\triangle}{=} \mathbf{H}^{\mathsf{T}}\mathbf{H}$.
- Let μ_1 and μ_2 be the largest and second largest eigenvalue, respectively, of $\mathbf{L}$.

Then the minimum Hamming weight and the minimum AWGNC pseudo-weight of C are lower bounded by

$$w_{\mathrm{H}}^{\min}(\mathsf{C}) \geq w_{\mathrm{p}}^{\min}(\mathsf{C}) \geq n \cdot \frac{2j - \mu_2}{\mu_1 - \mu_2}.$$

A Lower Bound on The Minimum Pseudo-Weight based on Linear Programming

Let $\boldsymbol{\omega}$ be any pseudo-codeword with $||\boldsymbol{\omega}||_1 = 1$. Then the (rank-1) matrix

$$\mathbf{M} \stackrel{\triangle}{=} \boldsymbol{\omega}^{\mathsf{T}} \cdot \boldsymbol{\omega} = \begin{pmatrix} \omega_1^2 & \omega_1\omega_2 & \omega_1\omega_3 & \cdots & \omega_1\omega_n \\ \omega_2\omega_1 & \omega_2^2 & \omega_2\omega_3 & \cdots & \omega_2\omega_n \\ \omega_3\omega_1 & \omega_3\omega_2 & \omega_3^2 & \cdots & \omega_3\omega_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_n\omega_1 & \omega_n\omega_2 & \omega_n\omega_3 & \cdots & \omega_n^2 \end{pmatrix}$$

has the following properties:

- entries are non-negative,
- $\sum_{i,j} [\mathbf{M}]_{i,j} = 1$,
- $\mathrm{Trace}(\mathbf{M}) = ||\boldsymbol{\omega}||_2^2$,
- row i of $\mathbf{M}$ equals $\omega_i \cdot \boldsymbol{\omega}$,
- column j of $\mathbf{M}$ equals $\omega_j \cdot \boldsymbol{\omega}^{\mathsf{T}}$.

Maximizing

$$\mathrm{Trace}(N)$$

over all $n \times n$-matrices $\mathbf{N}$ (not necessarily of rank 1) that fulfill

- entries are non-negative,
- $\sum_{i,j} [\mathbf{N}]_{i,j} = 1$,
- the rows are in the fundamental cone,
- the columns are in the fundamental cone,

we obtain $1/\mathrm{Trace}(N)$ as a lower bound on the minimum pseudo-weight. (Note: the above optimization problem is a linear program.)

Pseudo-Weights for other Channels

Let $\boldsymbol{\omega}$ be a (valid) pseudo-codeword. For each channel we can define a pseudo-weight, see [Wiberg:96], [FKKR:01].

- The AWGN channel pseudo-weight $w_{\mathrm{p}}(\boldsymbol{\omega})$ of $\boldsymbol{\omega}$ is given by

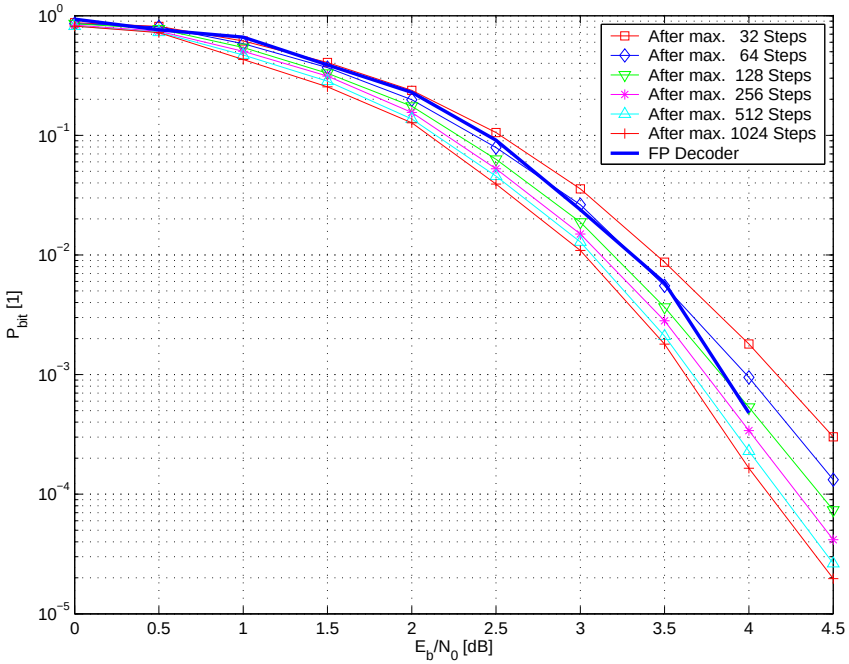
$$w_{\mathrm{p}}^{\mathrm{AWGNC}}(\boldsymbol{\omega}) = \frac{||\boldsymbol{\omega}||_1^2}{||\boldsymbol{\omega}||_2^2} = \frac{(|\omega_1| + \cdots + |\omega_n|)^2}{|\omega_1|^2 + \cdots + |\omega_n|^2} = \frac{(\omega_1 + \cdots + \omega_n)^2}{\omega_1^2 + \cdots + \omega_n^2}.$$

- The BSC channel pseudo-weight $w_{\mathrm{p}}^{\mathrm{BSC}}(\boldsymbol{\omega})$ is twice the median of the descendingly sorted vector $\boldsymbol{\omega}$.
- The BEC channel pseudo-weight $w_{\mathrm{p}}^{\mathrm{BEC}}(\boldsymbol{\omega})$ is the support of $\boldsymbol{\omega}$, i.e.

$$w_{\mathrm{p}}^{\mathrm{BEC}}(\boldsymbol{\omega}) = \mathrm{supp}(\boldsymbol{\omega}) = \big|\big\{i \in \{1, \ldots, n\} \mid \omega_i \neq 0\big\}\big|.$$

Note: the pseudo-weight definition depends on the channel but the fundamental polytope/cone is independent of the channel!

Connections to the Linear Programming Decoder



Max-Product decoder vs. linear program decoder

- The linear programming decoder was recently introduced by Feldman, Karger, and Wainwright.
- They formulate the decoding of a code as a relaxed integer programming problem in order to obtain a linear program.
- The most canonical relaxation yields exactly the polytope that we called the fundamental polytope.

Conclusions and Outlook (Part 1)

- MAP/ML vs. message-passing decoding:
  - When using a MAP/ML decoder, the transmitted codeword competes against all other codewords in the code.
  - When using a locally operating message-passing algorithms, the transmitted codeword competes against all pseudo-codewords.
- Codewords in graph covers are the systematic price one has to pay for using any locally operating message passing algorithm.
  - Pseudo-codewords are characterized by fundamental polytope/cone.
  - The pseudo-weight indicates badness of pseudo-codeword.

Conclusions and Outlook (Part 2)

- We have shown several techniques to lower/upper bound the minimum pseudo-weight of a factor graph realization of a code.
- Future work: LDPC code construction based on the avoidance of “bad patterns”.
- An intriguing question: how to design codes with good minimum pseudo-distance!