Pseudo-Codewords: Fractional Vectors in Coding Theory

Pascal O. Vontobel
Information Theory Research Group
Hewlett-Packard Laboratories Palo Alto

BIRS Workshop, Banff, Alberta, Canada, August 3, 2009

Overview of Talk

- Communications setup
- Linear programming (LP) decoding
- Pseudo-codeword spectra
- Graph-cover interpretation of pseudo-codewords
- Influence of redundant rows in the parity-check matrix and of cycles in the Tanner graph
- Pseudo-codwords and the edge zeta function
- Canonical completion construction
- LP decoding thresholds for the binary symmetric channel (BSC)

Note: see appendices for more details.

Communication systems and Shannon's channel coding theorem

Communication System (Part 1)

Source

Sink

Communication System (Part 1)

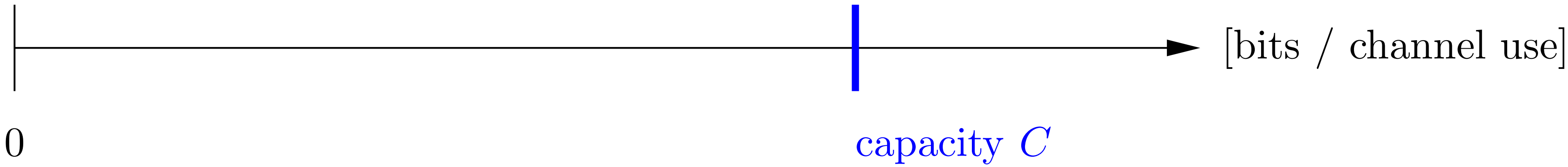
Communication System (Part 1)

Shannon (1948): it is a good idea to use channel codes!

Communication System (Part 1)

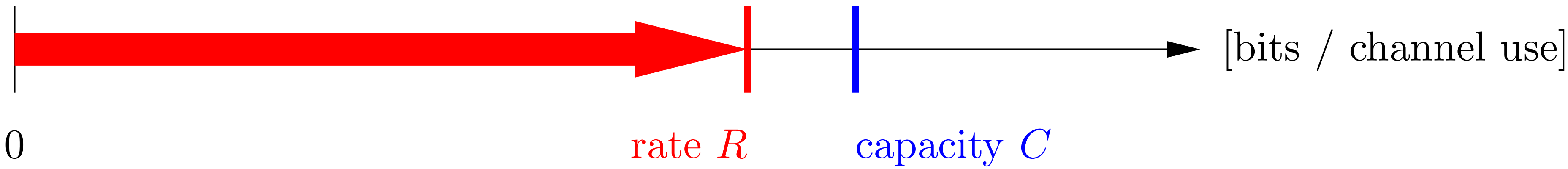
Shannon (1948): it is a good idea to use channel codes!

Communication System (Part 2)



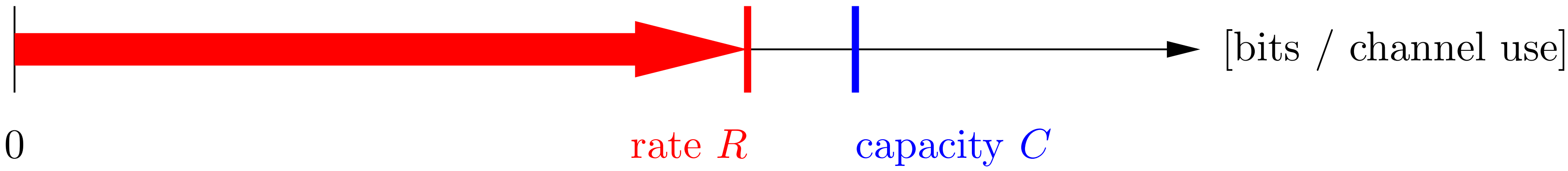
- A channel is characterized by a number C called the capacity.

Communication System (Part 2)



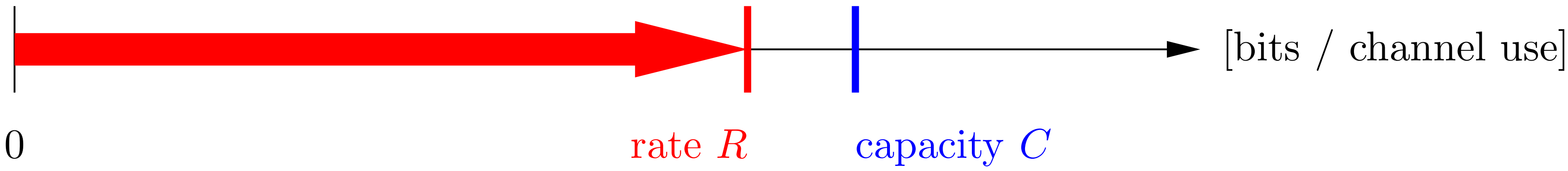
- A channel is characterized by a number C called the capacity.
- A code is characterized by a number R called the rate.

Communication System (Part 2)



- A channel is characterized by a number C called the capacity.
- A code is characterized by a number R called the rate.
- If $R < C$: there are codes, encoders, and decoders such that arbitrarily low error probabilities can be guaranteed (as long as one allows arbitrarily long codes).

Communication System (Part 2)



- A channel is characterized by a number C called the capacity.
- A code is characterized by a number R called the rate.
- If $R < C$: there are codes, encoders, and decoders such that arbitrarily low error probabilities can be guaranteed (as long as one allows arbitrarily long codes).
- Shannon's proof was though non-constructive, i.e. it was not clear at all how to obtain specific well-performing finite-length codes that possess efficient encoders and decoders.

"Traditional" vs. "Modern" Coding and Decoding

	Code design		**Decoding**
"Traditional"	Reed-Solomon codes etc.	→	?
"Modern"	?	→	Message-passing iterative decoding LP decoding

"Traditional" vs. "Modern" Coding and Decoding

	Code design		**Decoding**
"Traditional"	Reed-Solomon codes etc.	$\longrightarrow$	Berlekamp-Massey decoder etc.
"Modern"	?	$\longrightarrow$	Message-passing iterative decoding LP decoding

"Traditional" vs. "Modern" Coding and Decoding

	Code design		**Decoding**
"Traditional"	Reed-Solomon codes etc.	$\longrightarrow$	Berlekamp-Massey decoder etc.
"Modern"	Codes on Graphs (LDPC/Turbo codes, etc.)	$\longrightarrow$	Message-passing Iterative decoding LP decoding

Communication Model (Part 1)

Information word: $\mathbf{u} = (u_1, \ldots, u_k) \in \mathcal{U}^k$

Sent codeword: $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C} \subseteq \mathcal{X}^n$

Received word: $\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$

Communication Model (Part 1)

Information word:	$\mathbf{u} = (u_1, \ldots, u_k) \in \mathcal{U}^k$
Sent codeword:	$\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C} \subseteq \mathcal{X}^n$
Received word:	$\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$

Decoding: Based on $\mathbf{y}$ we would like to estimate the transmitted codeword $\hat{\mathbf{x}}$ or the information word $\hat{\mathbf{u}}$.

Communication Model (Part 1)

Information word: $\mathbf{u} = (u_1, \ldots, u_k) \in \mathcal{U}^k$

Sent codeword: $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C} \subseteq \mathcal{X}^n$

Received word: $\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$

Decoding: Based on $\mathbf{y}$ we would like to estimate the transmitted codeword $\hat{\mathbf{x}}$ or the information word $\hat{\mathbf{u}}$.

Depending on what criterion we optimize, we obtain different decoding algorithms.

Communication Model (Part 2)

- Min. the block error prob. results in block-wise MAP decoding

$$\hat{\mathbf{u}}_{\mathrm{MAP}}^{\mathrm{block}}(\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\; P_{\mathbf{U}|\mathbf{Y}}(\mathbf{u}|\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\; P_{\mathbf{U},\mathbf{Y}}(\mathbf{u},\mathbf{y}).$$

Communication Model (Part 2)

$$\text{BSS} \xrightarrow{\mathbf{U}} \text{Channel Coding} \xrightarrow{\mathbf{X}} \text{Channel} \xrightarrow{\mathbf{Y}} \text{Channel Decoding} \xrightarrow{\hat{\mathbf{X}}\ \ \hat{\mathbf{U}}} \text{Sink}$$

- Min. the block error prob. results in block-wise MAP decoding

$$\hat{\mathbf{u}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\ P_{\mathbf{U}|\mathbf{Y}}(\mathbf{u}|\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\ P_{\mathbf{U},\mathbf{Y}}(\mathbf{u},\mathbf{y}).$$

- This can also be written as

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\ P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\ P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}).$$

Communication Model (Part 2)

$$\boxed{\text{BSS}} \xrightarrow{\mathbf{U}} \boxed{\text{Channel Coding}} \xrightarrow{\mathbf{X}} \boxed{\text{Channel}} \xrightarrow{\mathbf{Y}} \boxed{\text{Channel Decoding}} \xrightarrow{\hat{\mathbf{X}}\ \hat{\mathbf{U}}} \boxed{\text{Sink}}$$

- Min. the block error prob. results in block-wise MAP decoding

$$\hat{\mathbf{u}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\ P_{\mathbf{U}|\mathbf{Y}}(\mathbf{u}|\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\ P_{\mathbf{U},\mathbf{Y}}(\mathbf{u},\mathbf{y}).$$

- This can also be written as

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\ P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\ P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}).$$

- If all codewords are equally likely then

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\ P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \underset{\mathbf{x}\in\mathcal{C}}{\operatorname{argmax}}\ P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$$

Communication Model (Part 2)

- Min. the block error prob. results in block-wise MAP decoding

$$\hat{\mathbf{u}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\; P_{\mathbf{U}|\mathbf{Y}}(\mathbf{u}|\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\; P_{\mathbf{U},\mathbf{Y}}(\mathbf{u},\mathbf{y}).$$

- This can also be written as

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\; P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\; P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}).$$

- If all codewords are equally likely then

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\; P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \underset{\mathbf{x}\in\mathcal{C}}{\operatorname{argmax}}\; P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \triangleq \hat{\mathbf{x}}^{\text{block}}_{\text{ML}}(\mathbf{y}).$$

Linear Code Representations

Image representation:

Kernel representation:

Linear Code Representations

Image representation (based on generator matrix $\mathbf{G}$):

$$\mathcal{C} = \left\{ \mathbf{x} \in \mathbb{F}^n \;\middle|\; \text{there exists } \mathbf{u} \in \mathbb{F}^k \text{ such that } \mathbf{x} = \mathbf{u} \cdot \mathbf{G} \right\}.$$

Kernel representation:

Linear Code Representations

Image representation (based on generator matrix $\mathbf{G}$):

$$\mathcal{C} = \left\{ \mathbf{x} \in \mathbb{F}^n \;\middle|\; \text{there exists } \mathbf{u} \in \mathbb{F}^k \text{ such that } \mathbf{x} = \mathbf{u} \cdot \mathbf{G} \right\} .$$

Kernel representation (based on parity-check matrix $\mathbf{H}$):

$$\mathcal{C} = \left\{ \mathbf{x} \in \mathbb{F}^n \;\middle|\; \mathbf{x} \cdot \mathbf{H}^{\mathsf{T}} = \mathbf{0} \right\} .$$

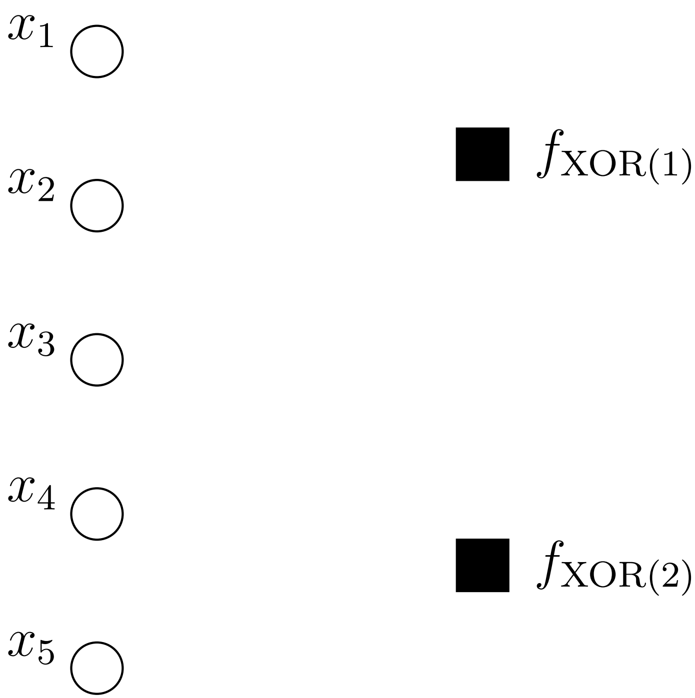
Linear Code Representations (Example 1)

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

Linear Code Representations (Example 1)

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

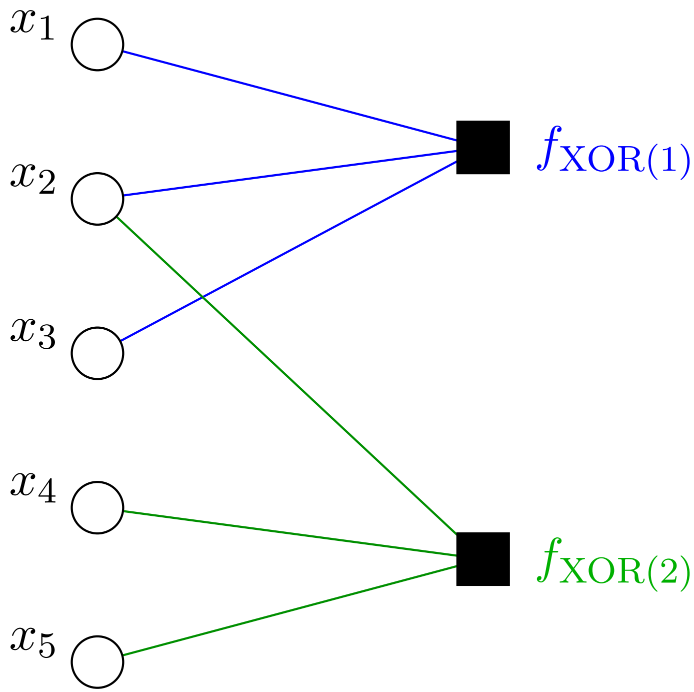
Tanner / factor graph representation:



Linear Code Representations (Example 1)

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

Tanner / factor graph representation:



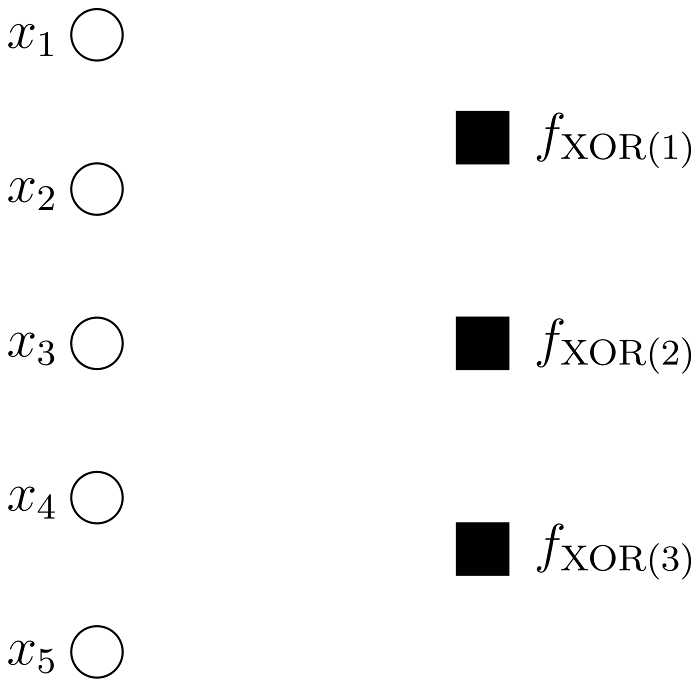
Linear Code Representations (Example 2)

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

Linear Code Representations (Example 2)

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

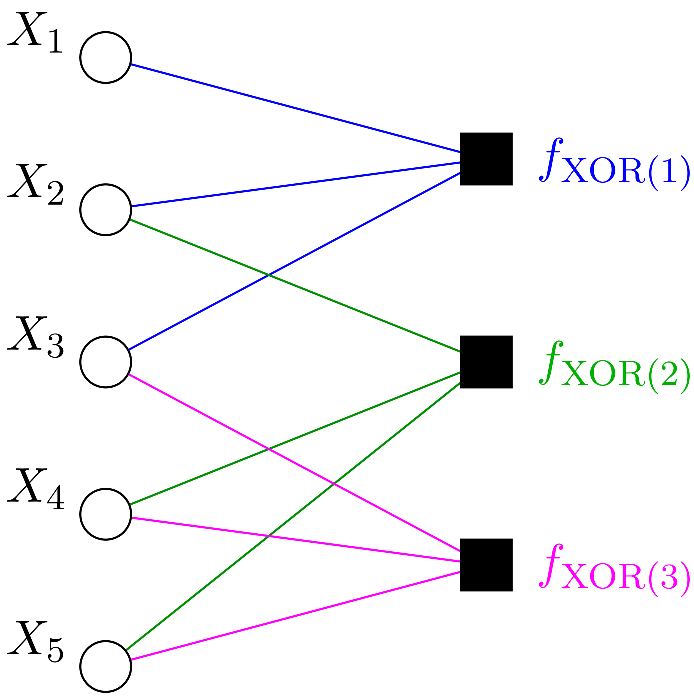
Tanner / factor graph representation:



Linear Code Representations (Example 2)

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

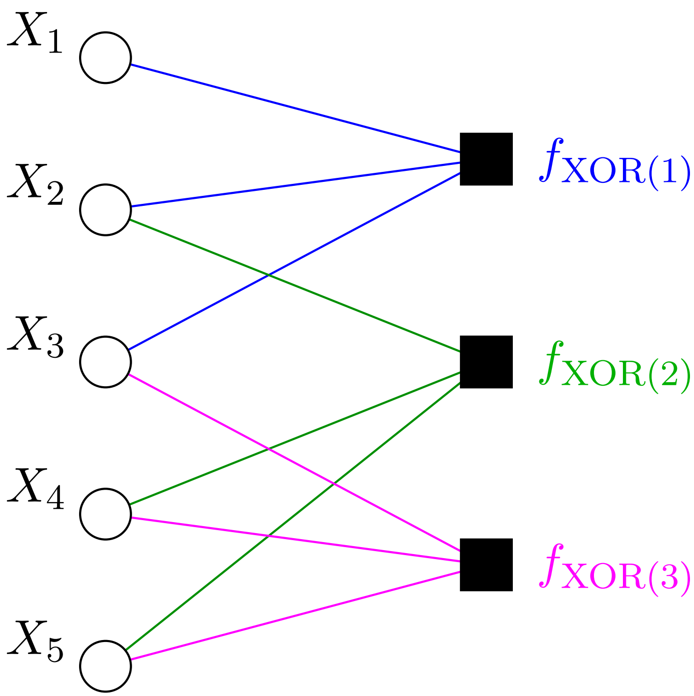
Tanner / factor graph representation:



Linear Code Representations (Example 2)

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

Tanner / factor graph representation:



Note: in contrast to Example 1, this Tanner graph has cycles.

Expressing a decoder as the solution of a linear program

ML Decoding as an *Integer* LP

For memoryless channels, block-wise ML decoding of a binary code can be written as a linear program.

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg\max_{\mathbf{x} \in \mathcal{C}} \; P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$$

ML Decoding as an *Integer* LP

For memoryless channels, block-wise ML decoding of a binary code can be written as a linear program.

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg\max_{\mathbf{x}\in\mathcal{C}} \; P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \arg\min_{\mathbf{x}\in\mathcal{C}} \; \sum_{i=1}^{n} x_i \lambda_i,$$

ML Decoding as an *Integer* LP

For memoryless channels, block-wise ML decoding of a binary code can be written as a linear program.

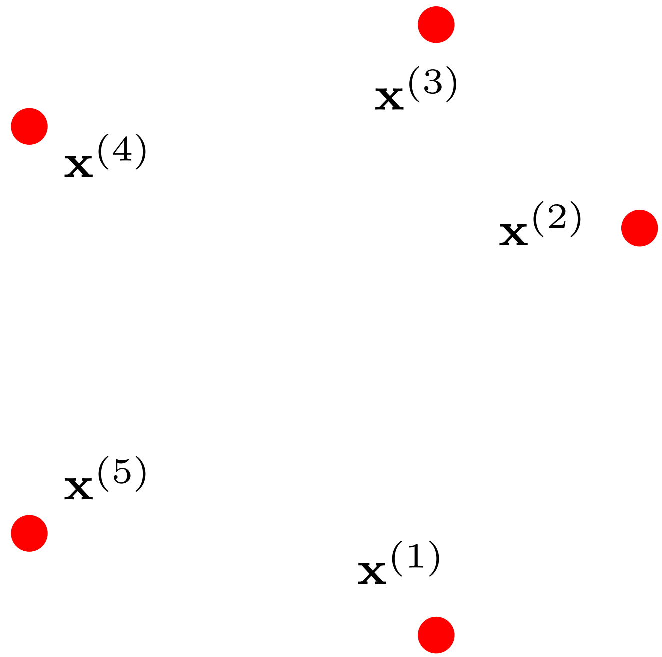
$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg\max_{\mathbf{x}\in\mathcal{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} x_i \lambda_i,$$

where

$$\lambda_i \triangleq \lambda_i(y_i) \triangleq \log \frac{P_{Y|X}(y_i|0)}{P_{Y|X}(y_i|1)}.$$

ML Decoding as an LP

$$\arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \lambda_i x_i$$



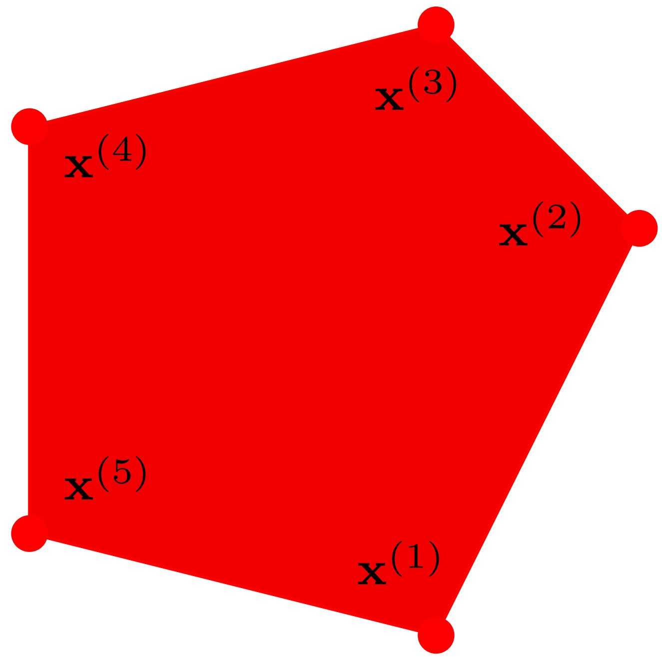
e.g.

$$\mathcal{C} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)}\right\}$$

ML Decoding as an LP

$$\arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \lambda_i x_i$$

$$\arg\min_{\mathbf{x}\in\mathrm{conv}(C)} \sum_{i=1}^{n} \lambda_i x_i$$



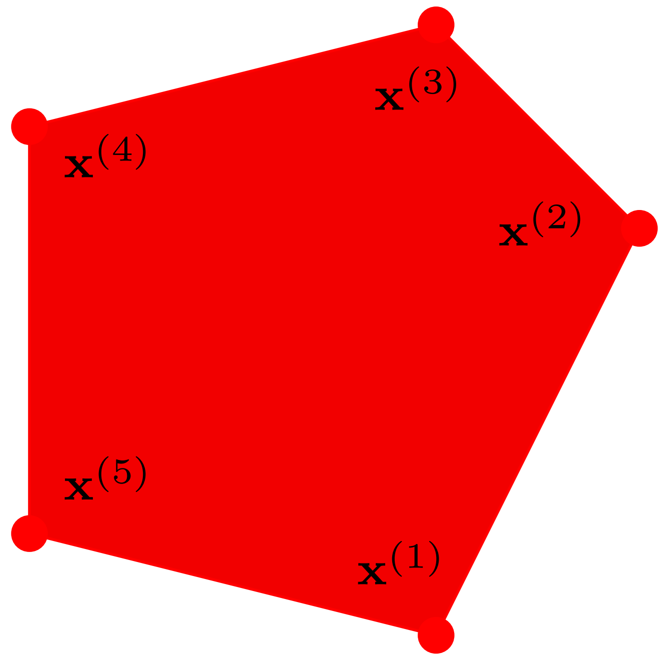
e.g.

$$\mathcal{C} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)}\right\}$$

ML Decoding as an LP

$$\arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \lambda_i x_i$$

$$\stackrel{*}{=} \arg\min_{\mathbf{x}\in\operatorname{conv}(C)} \sum_{i=1}^{n} \lambda_i x_i$$



e.g.

$$\mathcal{C} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)}\right\}$$

ML Decoding as an LP

$$\hat{\mathbf{x}}_{\text{ML}}^{\text{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \text{conv}(\mathcal{C})} \sum_{i=1}^{n} x_i \lambda_i,$$

This is a linear program.

ML Decoding as an LP

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathrm{conv}(\mathcal{C})} \sum_{i=1}^{n} x_i \lambda_i,$$

This is a linear program.

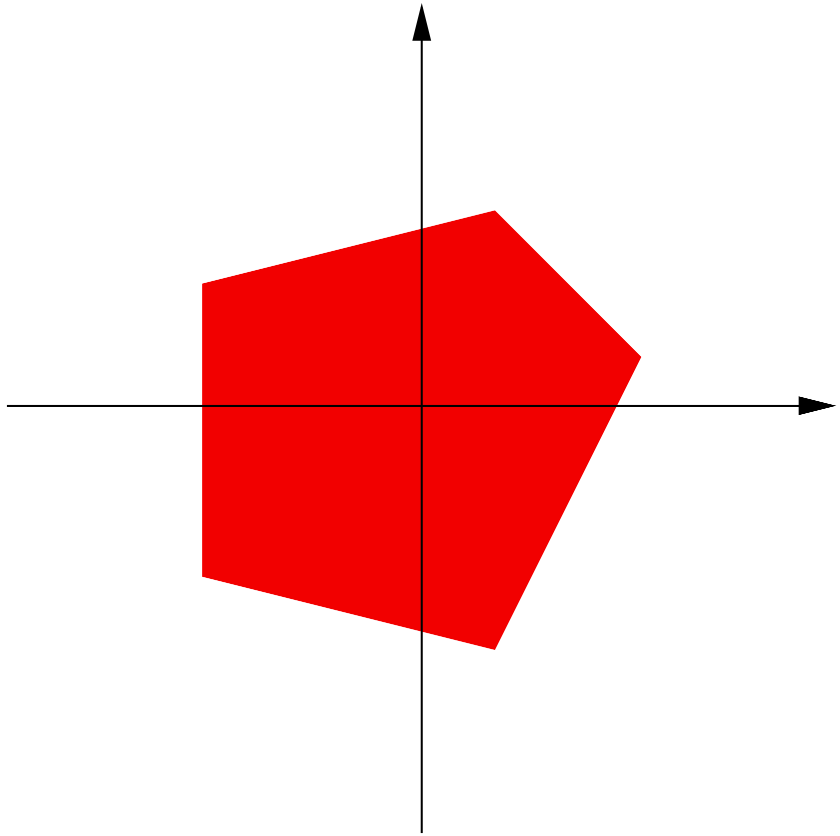
However, the

number of variables / equalities / inequalities

needed to describe the polytope $\mathrm{conv}(\mathcal{C})$ is (usually) exponential in n.

Relaxing the Previous LP

$$\arg \min_{\mathbf{x}\in\text{conv}(C)} \sum_{i=1}^{n} \lambda_i x_i$$

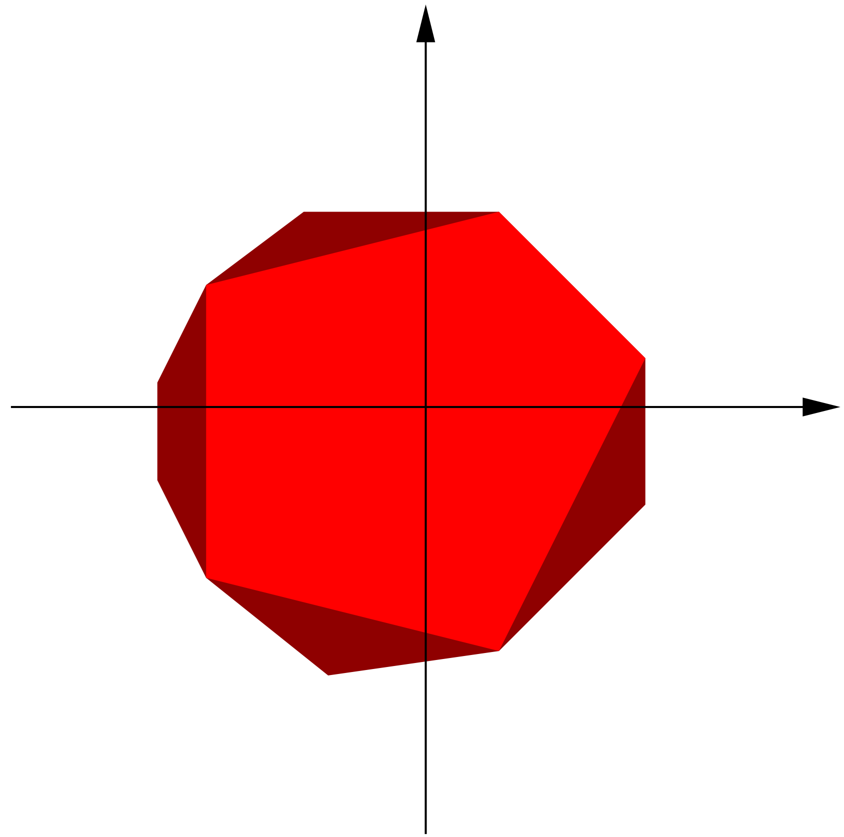


Relaxing the Previous LP

$$\arg \min_{\mathbf{x} \in \mathrm{conv}\,(C)} \sum_{i=1}^{n} \lambda_i x_i$$

is replaced by

$$\arg \min_{\mathbf{x} \in \mathrm{relax}(\mathrm{conv}\,(C))} \sum_{i=1}^{n} \lambda_i x_i$$

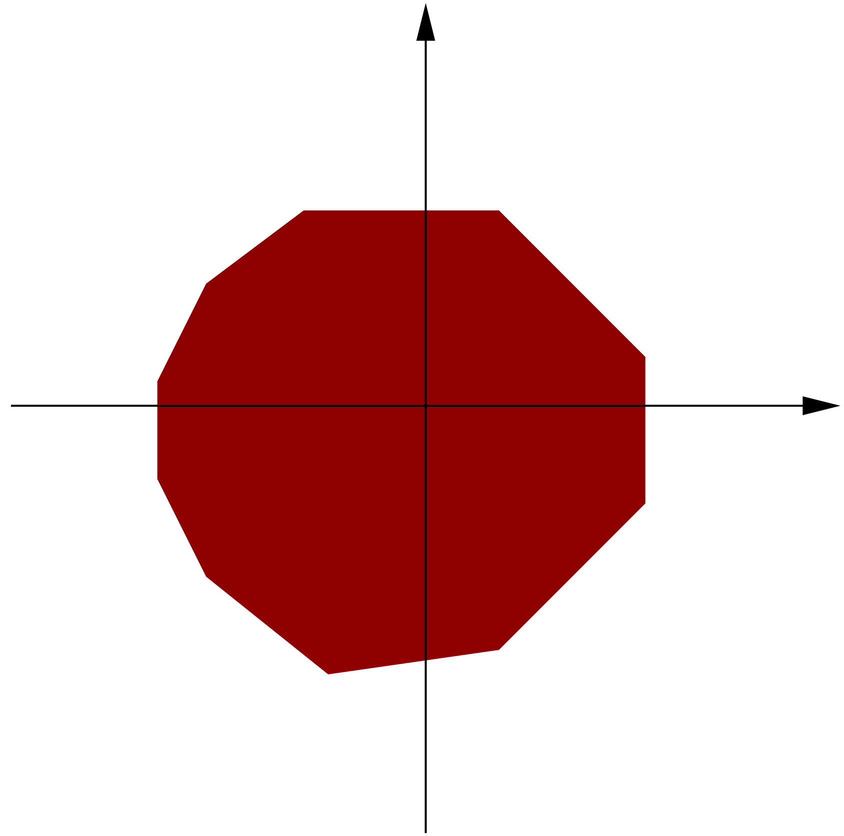


Relaxing the Previous LP

$$\arg \min_{\mathbf{x} \in \operatorname{conv}(C)} \sum_{i=1}^{n} \lambda_i x_i$$

is replaced by

$$\arg \min_{\mathbf{x} \in \operatorname{relax}(\operatorname{conv}(C))} \sum_{i=1}^{n} \lambda_i x_i$$

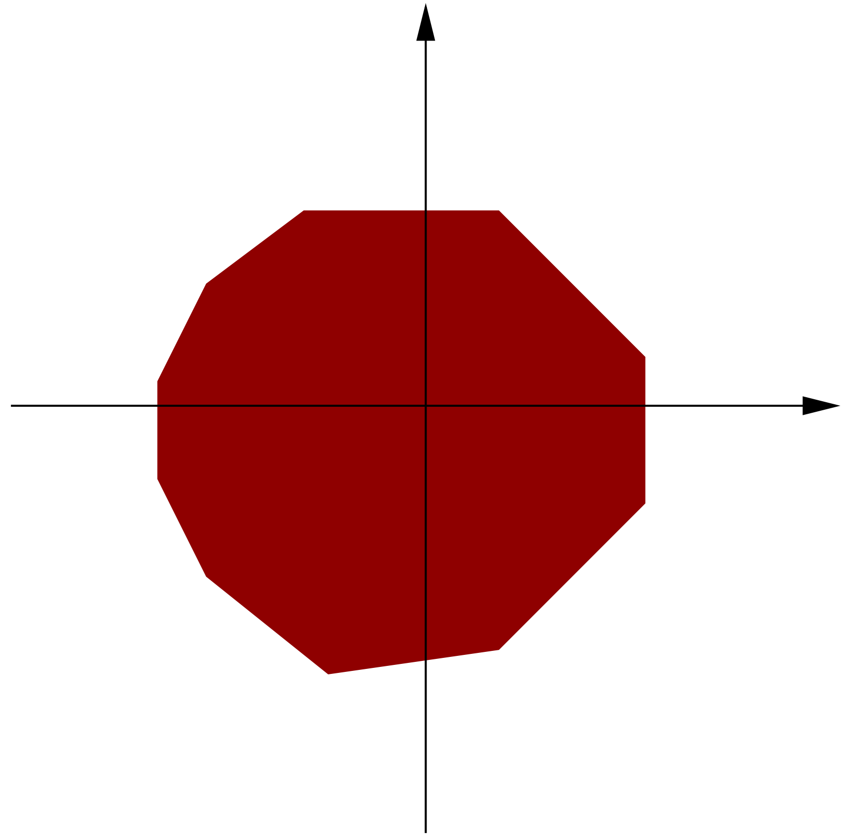


Relaxing the Previous LP

$$\arg \min_{\mathbf{x} \in \operatorname{conv}(C)} \sum_{i=1}^{n} \lambda_i x_i$$

is replaced by

$$\arg \min_{\mathbf{x} \in \operatorname{relax}(\operatorname{conv}(C))} \sum_{i=1}^{n} \lambda_i x_i$$



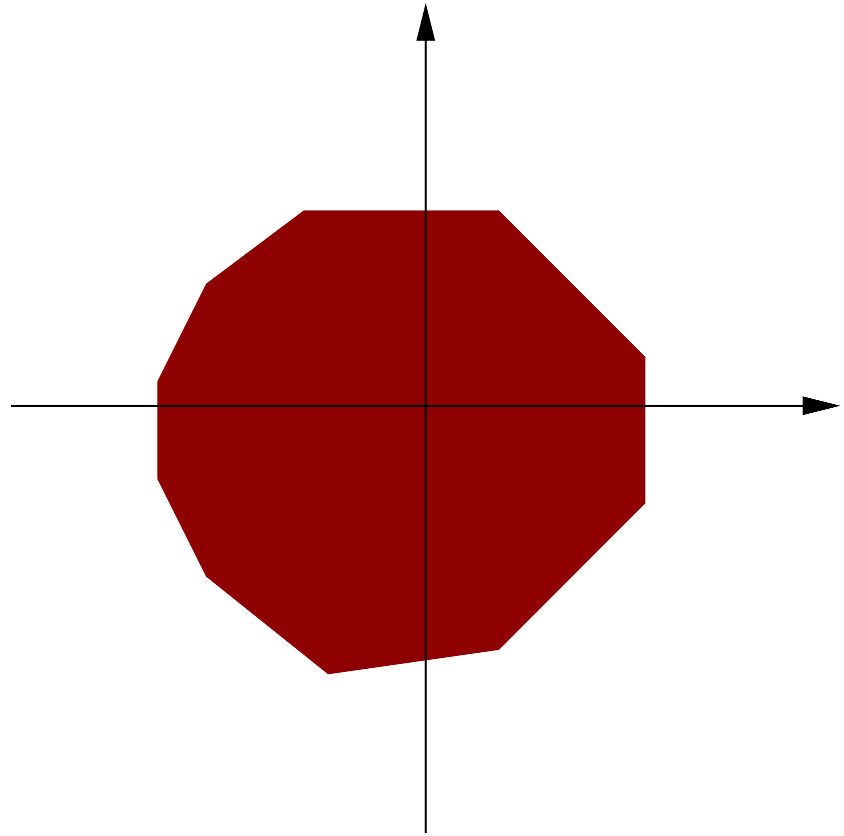
Desirable features:

Relaxing the Previous LP

$$\arg \min_{\mathbf{x} \in \operatorname{conv}(C)} \sum_{i=1}^{n} \lambda_i x_i$$

is replaced by

$$\arg \min_{\mathbf{x} \in \operatorname{relax}(\operatorname{conv}(C))} \sum_{i=1}^{n} \lambda_i x_i$$



Desirable features:

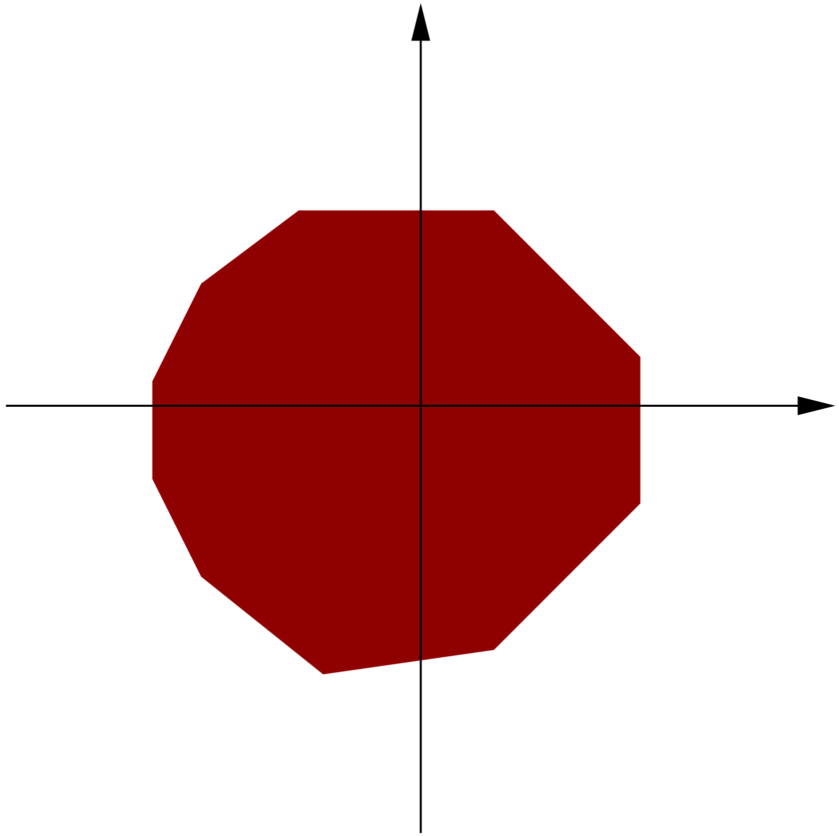
- old vertices are also vertices in relaxation;

Relaxing the Previous LP

$$\arg \min_{\mathbf{x} \in \operatorname{conv}(C)} \sum_{i=1}^{n} \lambda_i x_i$$

is replaced by

$$\arg \min_{\mathbf{x} \in \operatorname{relax}(\operatorname{conv}(C))} \sum_{i=1}^{n} \lambda_i x_i$$



Desirable features:

- old vertices are also vertices in relaxation;
- relaxation has simple description.

A Interesting Relaxation

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{array}{l} \Rightarrow \mathcal{C}_1 \\ \Rightarrow \mathcal{C}_2 \\ \Rightarrow \mathcal{C}_3 \end{array}$$

$$\Rightarrow \mathcal{C} = \bigcap_{j=1}^{m} \mathcal{C}_j$$

A Interesting Relaxation

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{array}{l} \Rightarrow \mathcal{C}_1 \\ \Rightarrow \mathcal{C}_2 \\ \Rightarrow \mathcal{C}_3 \end{array} \qquad \begin{array}{l} \Rightarrow \mathrm{conv}(\mathcal{C}_1) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_2) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_3) \end{array}$$

$$\Rightarrow \mathcal{C} = \bigcap_{j=1}^{m} \mathcal{C}_j \qquad \Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)}_{\text{relaxation}}$$

Block-wise ML Decoding vs. LP Decoding

Block-wise ML decoding:

LP decoding:

Block-wise ML Decoding vs. LP Decoding

Block-wise ML decoding:

$$\hat{\mathbf{x}}^{\text{block}}_{\text{ML}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \text{conv}(\mathcal{C})} \sum_{i=1}^{n} x_i \lambda_i .$$

LP decoding:

Block-wise ML Decoding vs. LP Decoding

Block-wise ML decoding:

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathrm{conv}(\mathcal{C})} \sum_{i=1}^{n} x_i \lambda_i .$$

LP decoding:

$$\hat{\boldsymbol{\omega}}_{\mathrm{LP}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \mathrm{relax}(\mathrm{conv}\,(C))} \sum_{i=1}^{n} \omega_i \lambda_i .$$

Block-wise ML Decoding vs. LP Decoding

Block-wise ML decoding:

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathrm{conv}\left(\cap_{j=1}^{m} \mathcal{C}_j\right)} \sum_{i=1}^{n} x_i \lambda_i .$$

LP decoding:

$$\hat{\boldsymbol{\omega}}_{\mathrm{LP}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \cap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)} \sum_{i=1}^{n} \omega_i \lambda_i .$$

Block-wise ML Decoding vs. LP Decoding

Block-wise ML decoding:

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathrm{conv}\left(\cap_{j=1}^{m} \mathcal{C}_j\right)} \sum_{i=1}^{n} x_i \lambda_i .$$

LP decoding:

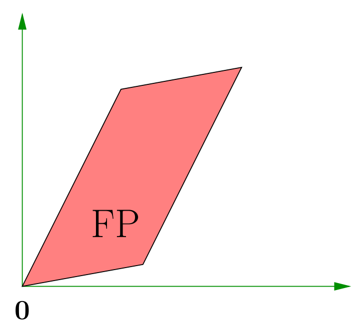
$$\hat{\boldsymbol{\omega}}_{\mathrm{LP}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \cap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)} \sum_{i=1}^{n} \omega_i \lambda_i .$$

The above choice of $\mathrm{relax}(\mathrm{conv}\,(C))$ was suggested by [Feldman/Wainwright/Karger:03/05]. (Here, $\mathcal{C}_j$ is the set of vectors that satisfy only the parity-check given by the j-th row of $\mathbf{H}$.)

Fundamental Polytope / Cone

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{array}{l} \Rightarrow \mathrm{conv}(\mathcal{C}_1) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_2) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_3) \end{array}$$

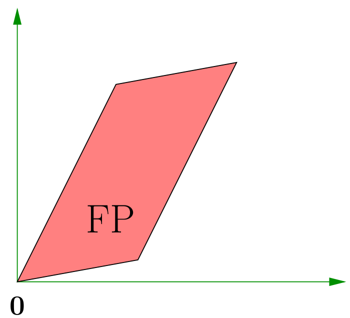
$$\Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)}_{\text{Fundamental polytope}}$$

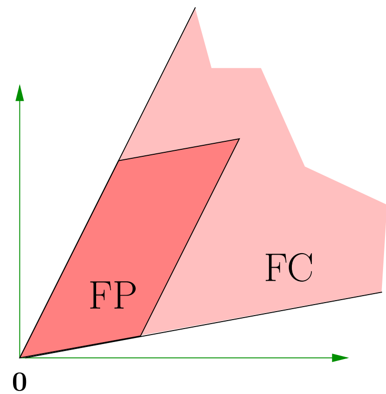


Fundamental Polytope / Cone

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{array}{l} \Rightarrow \mathrm{conv}(\mathcal{C}_1) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_2) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_3) \end{array}$$

$$\Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)}_{\text{Fundamental polytope}}$$



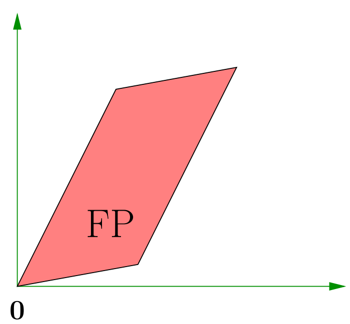


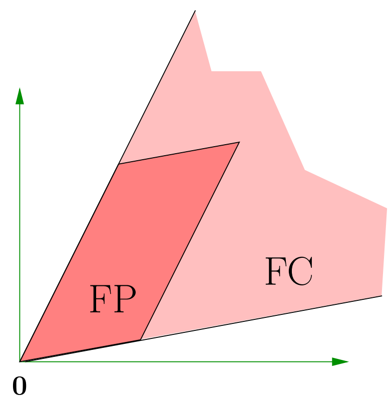
LABS hp

Fundamental Polytope / Cone

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{matrix} \Rightarrow \mathrm{conv}(\mathcal{C}_1) & \quad \Rightarrow \mathrm{conic}(\mathcal{C}_1) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_2) & \quad \Rightarrow \mathrm{conic}(\mathcal{C}_2) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_3) & \quad \Rightarrow \mathrm{conic}(\mathcal{C}_3) \end{matrix}$$

$$\Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)}_{\text{Fundamental polytope}} \quad \Rightarrow \underbrace{\mathcal{K}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathrm{conic}(\mathcal{C}_j)}_{\text{Fundamental cone}}$$





Fundamental Polytope / Cone

Definitions:

Fundamental Polytope / Cone

Definitions:

- Vectors in the fundamental polytope are called pseudo-codewords.

Fundamental Polytope / Cone

Definitions:

- Vectors in the fundamental polytope are called pseudo-codewords. (Sometimes only the vertices of the fundamental polytope are called pseudo-codewords.)

Fundamental Polytope / Cone

Definitions:

- Vectors in the fundamental polytope are called pseudo-codewords. (Sometimes only the vertices of the fundamental polytope are called pseudo-codewords.)
- Vectors in the fundamental cone are also called pseudo-codewords.

Fundamental Polytope / Cone

Definitions:

- Vectors in the fundamental polytope are called pseudo-codewords. (Sometimes only the vertices of the fundamental polytope are called pseudo-codewords.)
- Vectors in the fundamental cone are also called pseudo-codewords.
- Edges of the fundamental polytope/cone through origin are called minimal pseudo-codewords.

Fundamental Polytope / Cone

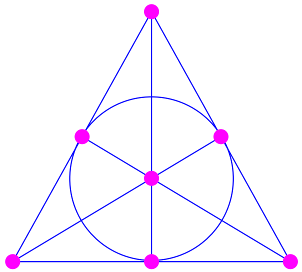
Definitions:

- Vectors in the fundamental polytope are called pseudo-codewords. (Sometimes only the vertices of the fundamental polytope are called pseudo-codewords.)
- Vectors in the fundamental cone are also called pseudo-codewords.
- Edges of the fundamental polytope/cone through origin are called minimal pseudo-codewords.

Very important: the fundamental polytope is a function of the parity-check matrix representing a code — differrent parity-check matrices for the same code can yield different fundamental polytopes.

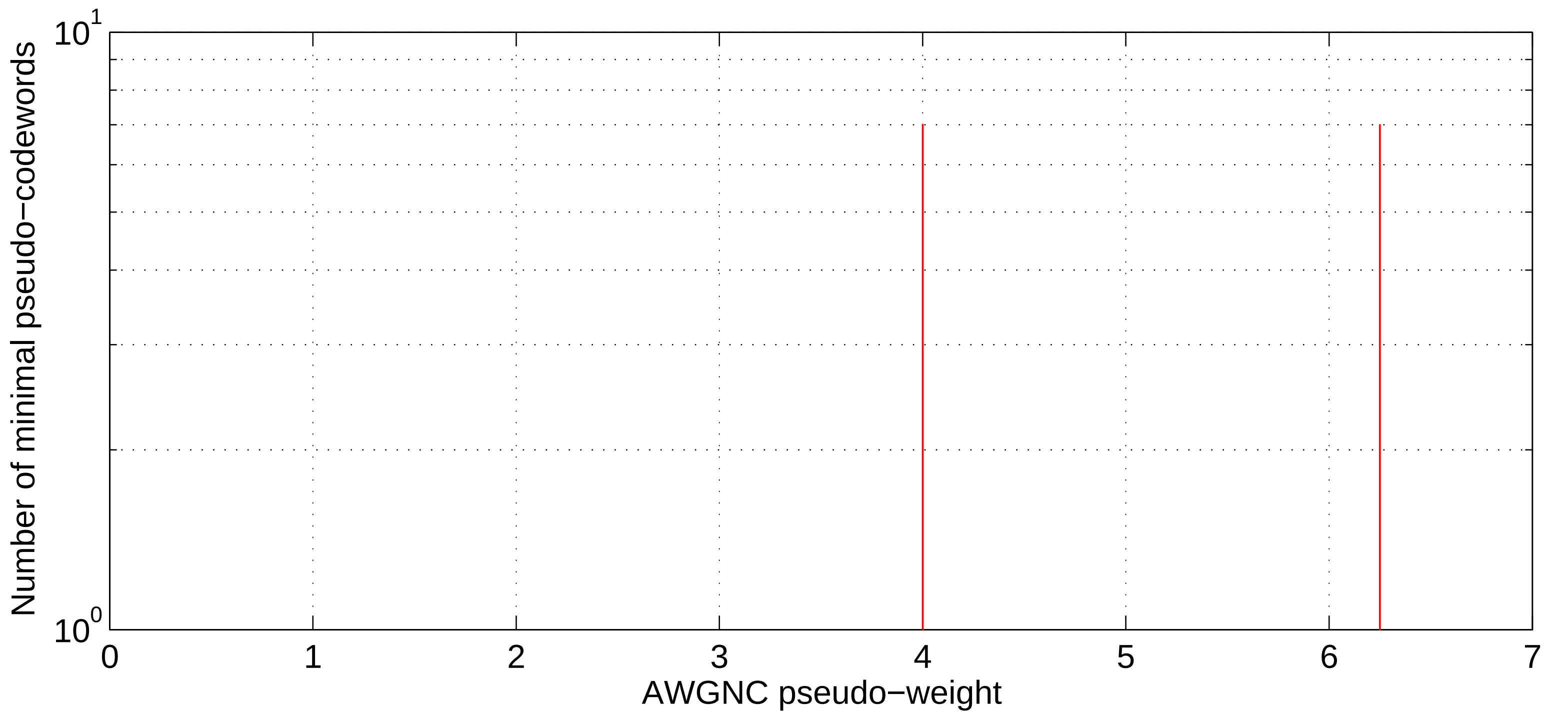
Pseudo-codeword spectra

Pseudo-Codeword Spectra



Consider the PG(2,2)-based $[7, 3, 4]$ binary linear code.

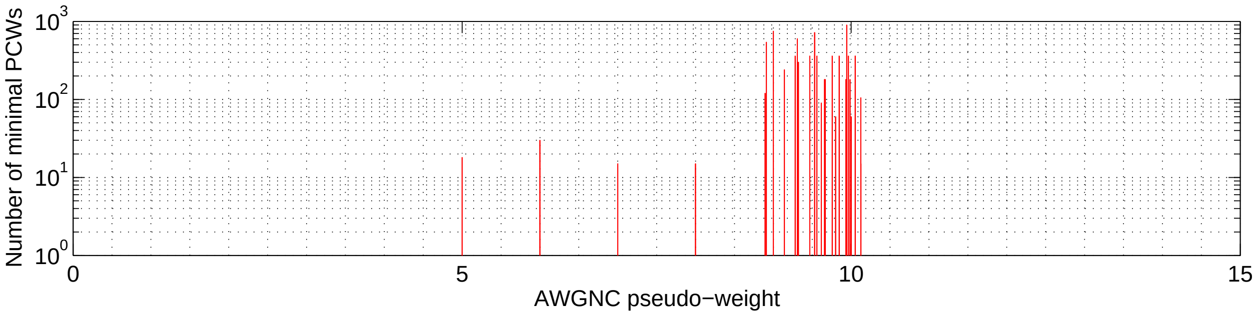
Here is its minimal pseudo-codeword spectrum:



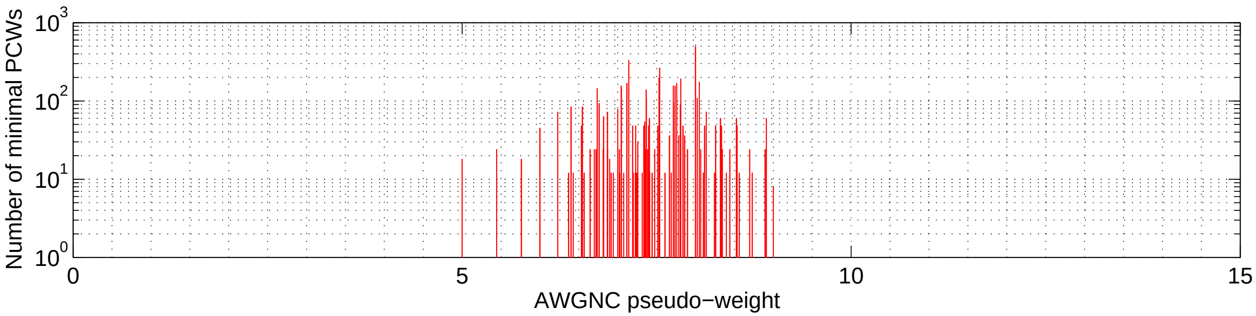
Pseudo-Codeword Spectra

Consider the EG(2,4)-based $[15, 7, 5]$ binary linear code.

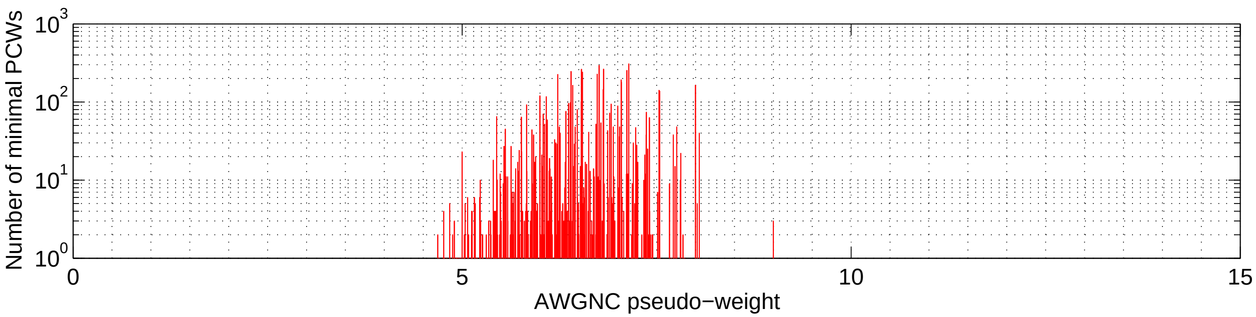
Here are some minimal pseudo-codeword spectra for different parity-check matrices of this code:



PCM of size 15×15



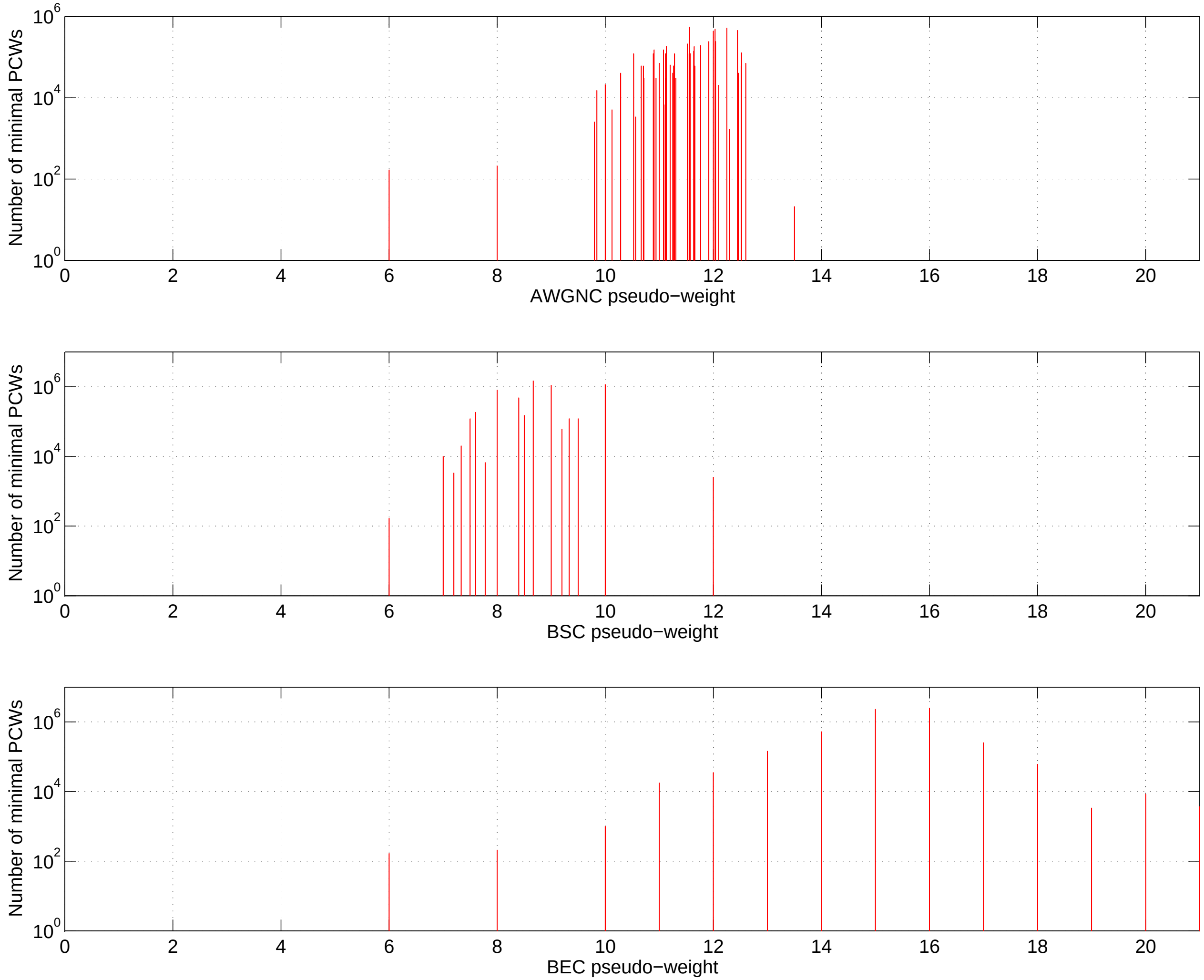
PCM of size 9×15



PCM of size 8×15

Pseudo-Codeword Spectra

Consider the PG(2,4)-based $[21, 11, 6]$ binary linear code.



Pseudo-Codeword Spectra

Some remarks:

- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes

Pseudo-Codeword Spectra

Some remarks:

- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes
  - where the minimal BEC pseudo-weight grows with growing block length,

Pseudo-Codeword Spectra

Some remarks:

- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes
  - where the minimal BEC pseudo-weight grows with growing block length,
  - but where the minimual AWGNC pseudo-weight is bounded from above.

Pseudo-Codeword Spectra

Some remarks:

- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes
  - where the minimal BEC pseudo-weight grows with growing block length,
  - but where the minimual AWGNC pseudo-weight is bounded from above.

$\Rightarrow$ It is important which channel is used!

Pseudo-Codeword Spectra

Some remarks:

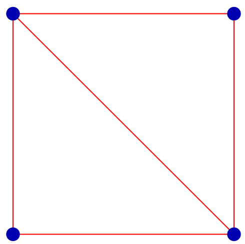
- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes
  - where the minimal BEC pseudo-weight grows with growing block length,
  - but where the minimual AWGNC pseudo-weight is bounded from above.

  ⇒ It is important which channel is used!

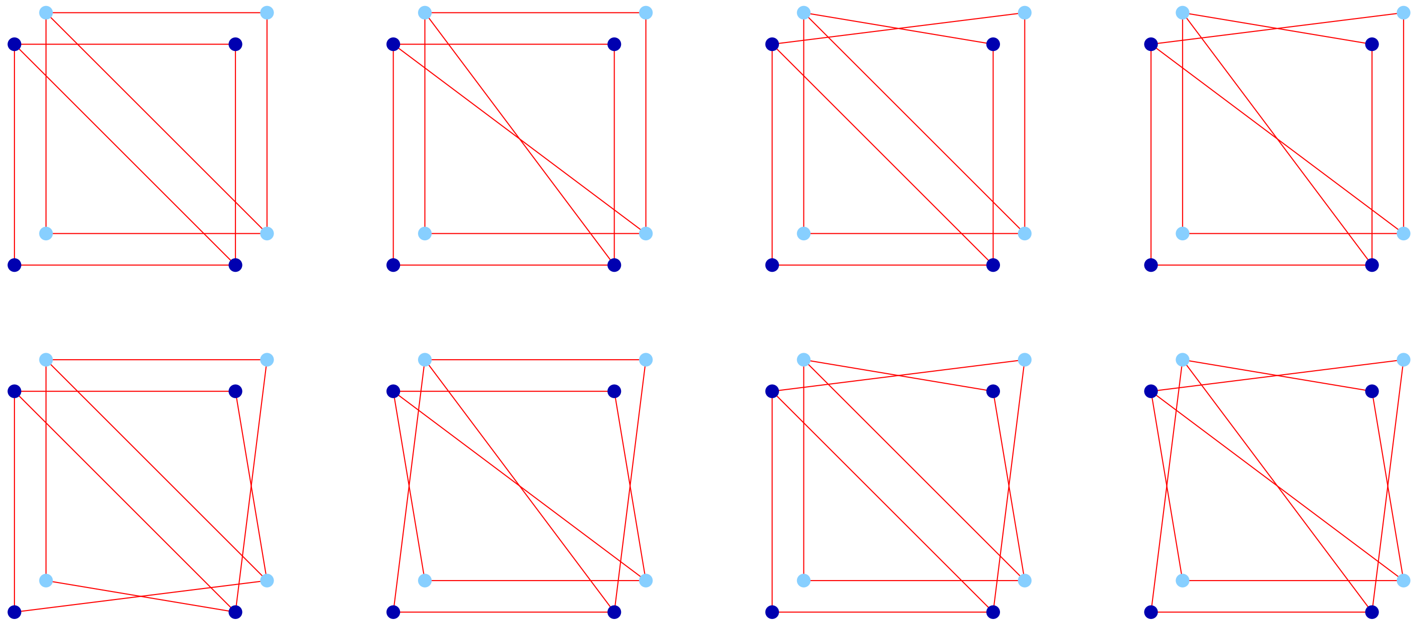
- Chertkov / Stepanov paper (ISIT 2007) presented an intesting heuristic for approximating the pseudo-weight spectra of minimal codewords for a given code.

Graph-cover interpretation of pseudo-codewords

Graph Covers (Part 1)



original graph

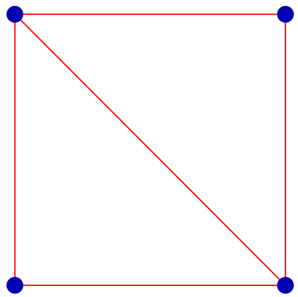


sample of possible double covers of the original graph

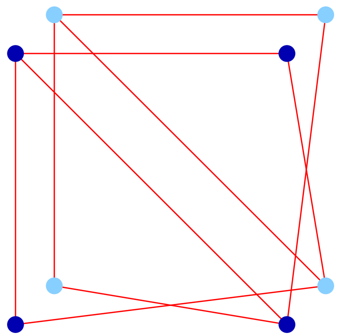
Definition: A double cover of a graph is . . .

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

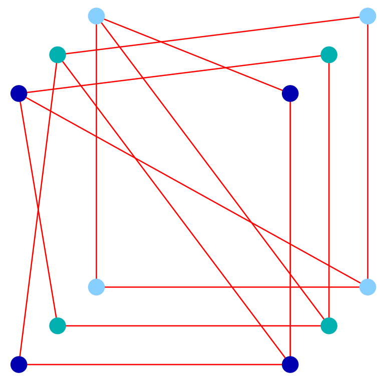
Graph Covers (Part 2)



original graph



(a possible)
double cover of
the original graph

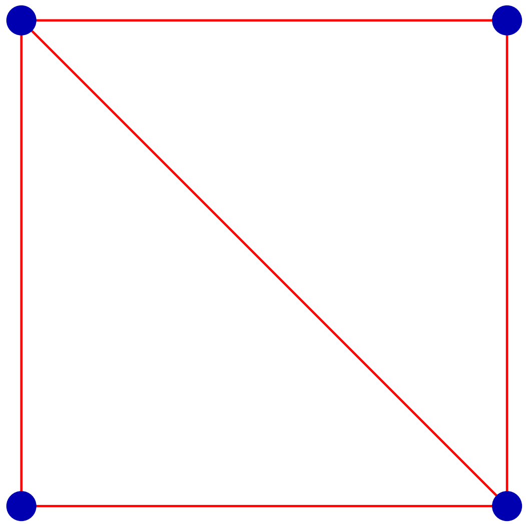


(a possible)
triple cover of
the original graph

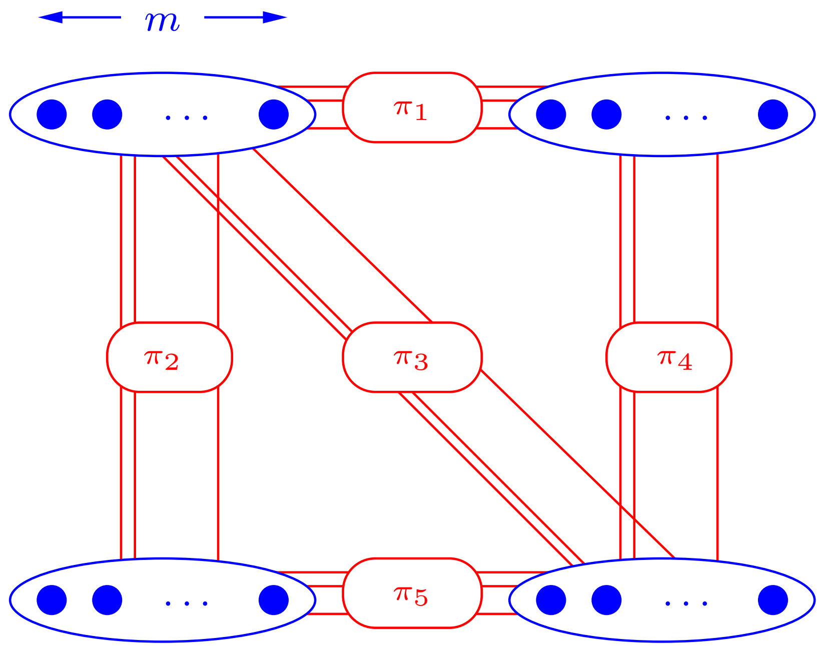
. . .

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

Graph Covers (Part 3)



original graph



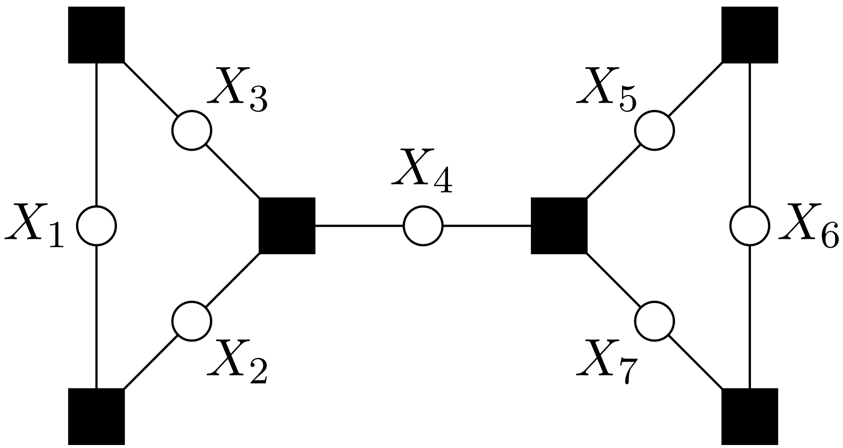
(possible)
m-fold cover of
original graph

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.

Codewords in Graph Covers (Part 1)

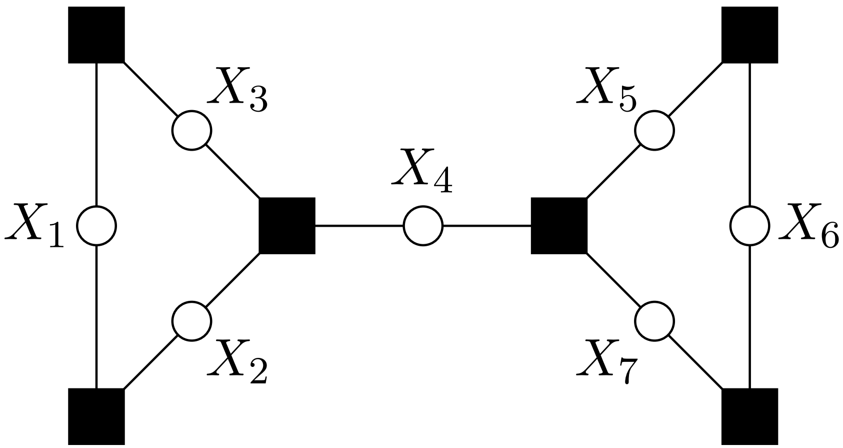
We can also consider covers of Tanner/factor graphs. Here is e.g. a possible double cover of some Tanner/factor graph.



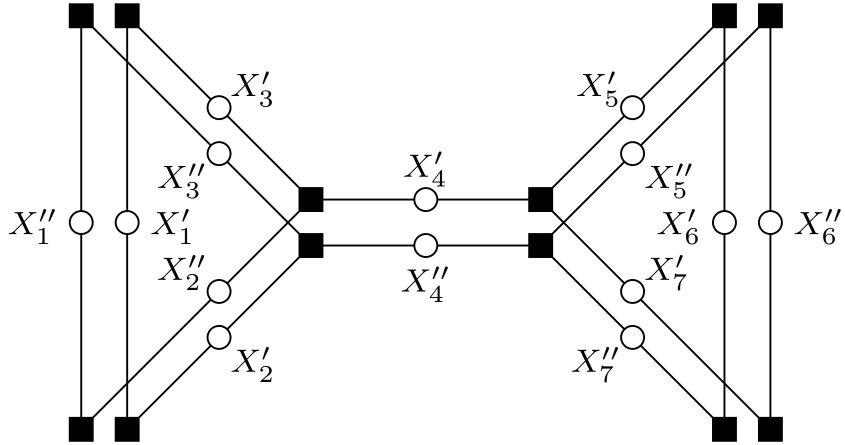
Base factor/Tanner graph
of a length-7 code

Codewords in Graph Covers (Part 1)

We can also consider covers of Tanner/factor graphs. Here is e.g. a possible double cover of some Tanner/factor graph.



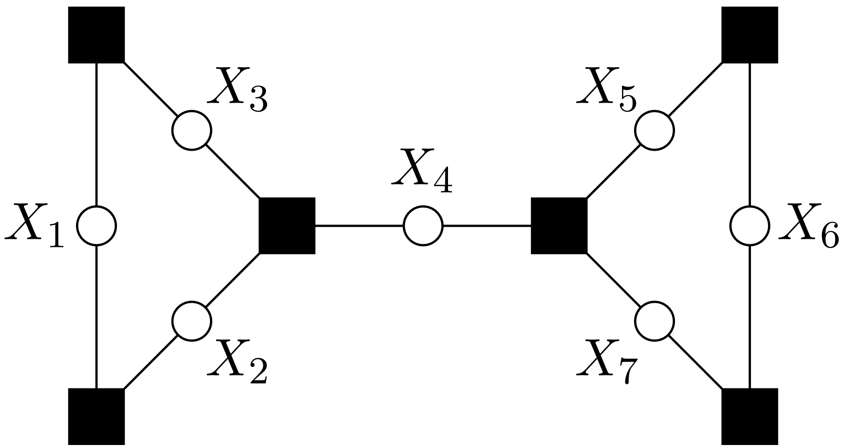
Base factor/Tanner graph
of a length-7 code



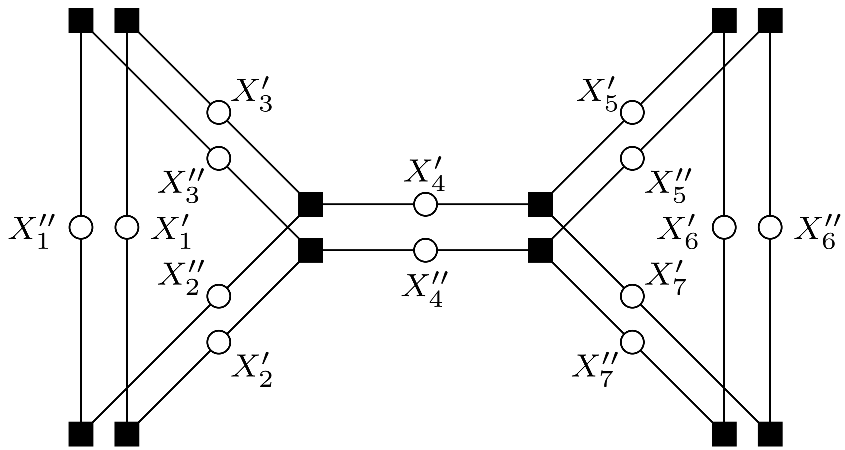
Possible double cover of
the base Tanner/factor graph

Codewords in Graph Covers (Part 1)

We can also consider covers of Tanner/factor graphs. Here is e.g. a possible double cover of some Tanner/factor graph.



Base factor/Tanner graph of a length-7 code

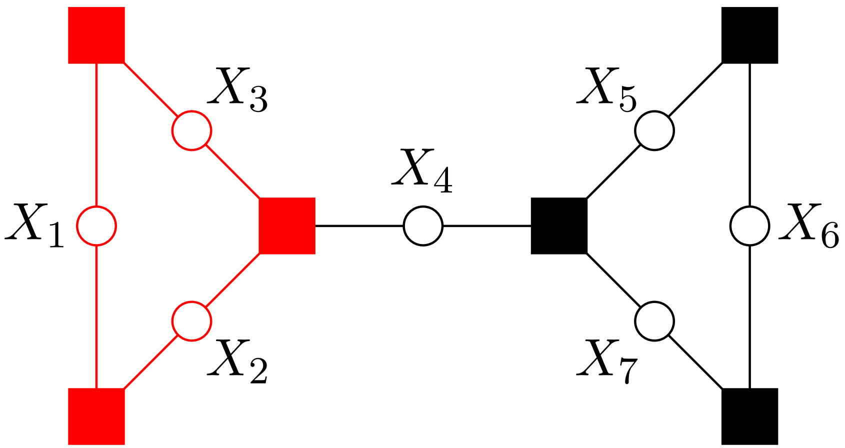


Possible double cover of the base Tanner/factor graph

Let us study the codes defined by the graph covers of the base Tanner/factor graph.

Codewords in Graph Covers (Part 2)

Obviously, any codeword in the base Tanner/factor graph can be lifted to a codeword in the double cover of the base Tanner/factor graph.

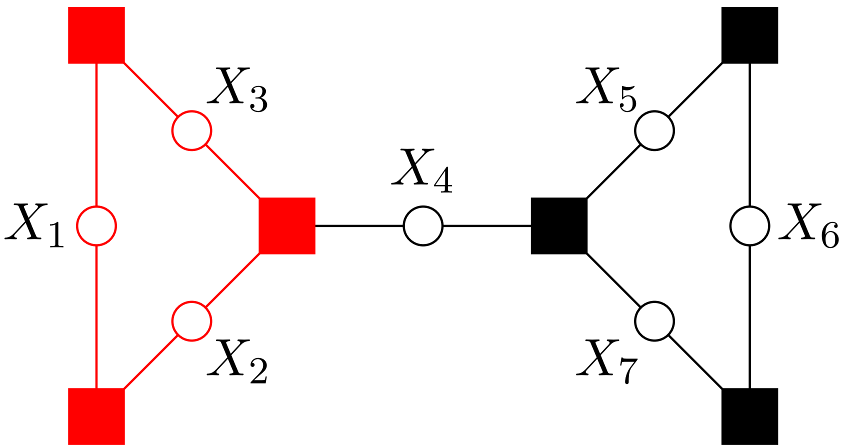


$\Rightarrow$

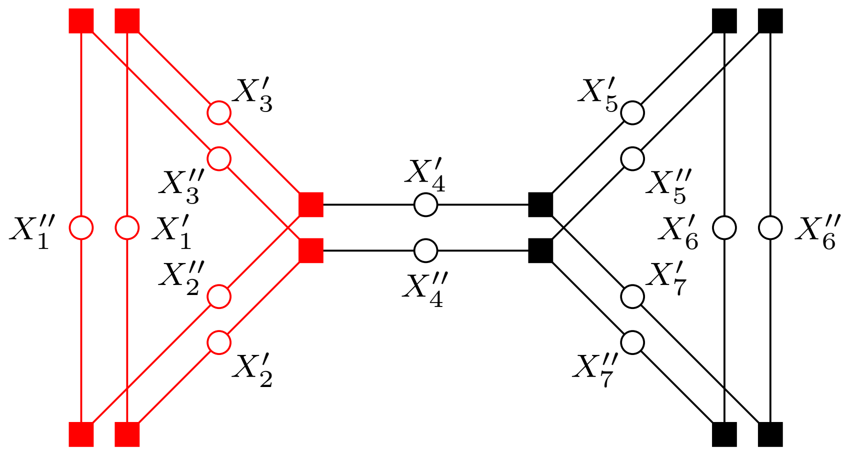
$$(1, 1, 1, 0, 0, 0, 0)$$

Codewords in Graph Covers (Part 2)

Obviously, any codeword in the base Tanner/factor graph can be lifted to a codeword in the double cover of the base Tanner/factor graph.



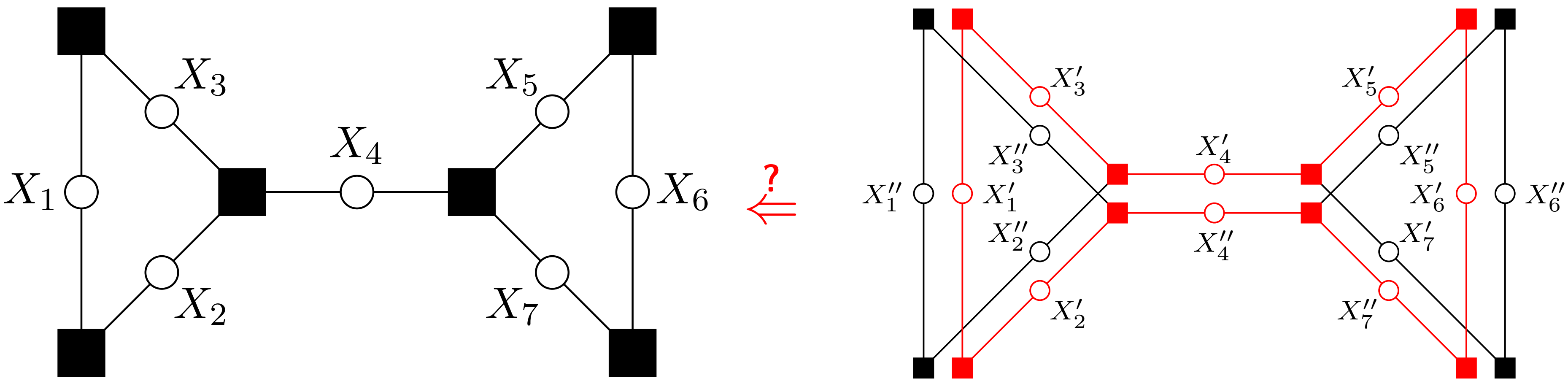
$\Rightarrow$



$(1, 1, 1, 0, 0, 0, 0)$ $\qquad$ $(1{:}1,\ 1{:}1,\ 1{:}1,\ 0{:}0,\ 0{:}0,\ 0{:}0,\ 0{:}0)$

Codewords in Graph Covers (Part 3)

But in the double cover of the base Tanner/factor graph there are also codewords that are not liftings of codewords in the base Tanner/factor graph!

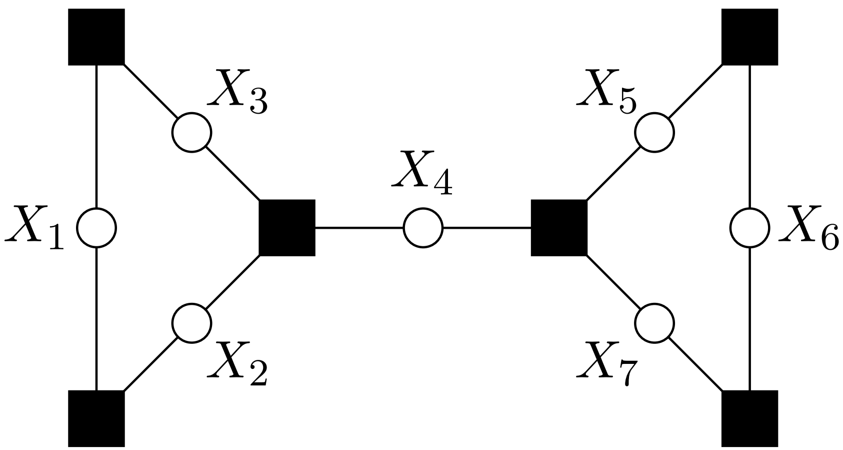


?

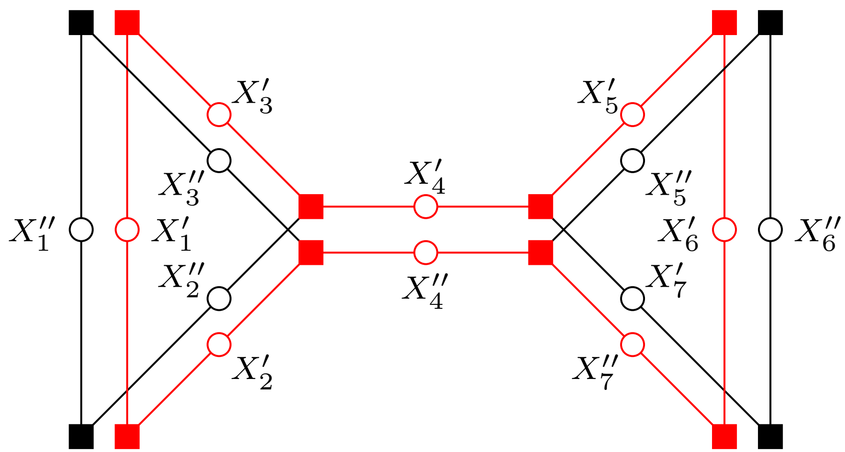
$(1{:}0,\ 1{:}0,\ 1{:}0,\ 1{:}1,\ 1{:}0,\ 1{:}0,\ 0{:}1)$

Codewords in Graph Covers (Part 3)

But in the double cover of the base Tanner/factor graph there are also codewords that are not liftings of codewords in the base Tanner/factor graph!



$\overset{?}{\Leftarrow}$



What about

$$\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{2}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right) ?$$

$$(1{:}0,\ 1{:}0,\ 1{:}0,\ 1{:}1,\ 1{:}0,\ 1{:}0,\ 0{:}1)$$

Codewords in Graph Covers (Part 4)

Theorem:

Codewords in Graph Covers (Part 4)

Theorem:

- Let $\mathcal{P} \triangleq \mathcal{P}(\mathbf{H})$ be the fundamental polytope of a parity-check matrix $\mathbf{H}$.

Codewords in Graph Covers (Part 4)

Theorem:

- Let $\mathcal{P} \triangleq \mathcal{P}(\mathbf{H})$ be the fundamental polytope of a parity-check matrix $\mathbf{H}$.
- Let $\mathcal{P}'$ be the set of all vectors obtained through codewords in finite covers.

Codewords in Graph Covers (Part 4)

Theorem:

- Let $\mathcal{P} \triangleq \mathcal{P}(\mathbf{H})$ be the fundamental polytope of a parity-check matrix $\mathbf{H}$.
- Let $\mathcal{P}'$ be the set of all vectors obtained through codewords in finite covers.

Then, $\mathcal{P}'$ is dense in $\mathcal{P}$, i.e.

$$\mathcal{P}' = \mathcal{P} \cap \mathbb{Q}^n$$

$$\mathcal{P} = \mathrm{closure}(\mathcal{P}').$$

Codewords in Graph Covers (Part 4)

Theorem:

- Let $\mathcal{P} \triangleq \mathcal{P}(\mathbf{H})$ be the fundamental polytope of a parity-check matrix $\mathbf{H}$.
- Let $\mathcal{P}'$ be the set of all vectors obtained through codewords in finite covers.

Then, $\mathcal{P}'$ is dense in $\mathcal{P}$, i.e.

$$\mathcal{P}' = \mathcal{P} \cap \mathbb{Q}^n$$
$$\mathcal{P} = \mathrm{closure}(\mathcal{P}').$$

Moreover, note that all vertices of $\mathcal{P}$ are vectors with rational entries and are therefore also in $\mathcal{P}'$.

Influence

of redundant rows in the parity-check matrix

and of cycles in the Tanner graph

A Tanner Graph with Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_2) \\ \cdots \end{array}$$

A Tanner Graph with Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \cdots \end{array}$$

$$\tilde{\mathbf{H}} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{12}) \\ \cdots \end{array}$$

A Tanner Graph with Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \cdots \end{matrix}$$

$$\tilde{\mathbf{H}} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{12}) \\ \cdots \end{matrix}$$

If the support of the blue and the green line coincide in at least two position then we have

$$\mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \supseteq \mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_{12}).$$

A Tanner Graph without Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} \cdots \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_2) \\ \cdots \end{matrix}$$

A Tanner Graph without Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \Rightarrow \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_2) \\ \cdots \end{array}$$

$$\tilde{\mathbf{H}} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \Rightarrow \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_{12}) \\ \cdots \end{array}$$

A Tanner Graph without Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \cdots \end{matrix}$$

$$\tilde{\mathbf{H}} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{12}) \\ \cdots \end{matrix}$$

If the support of the blue and the green line coincide in at most one position then we have

$$\mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \;=\; \mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_{12}).$$

Tanner Graphs with/without Four-Cycles

Proposition: It seems to be favorable to have no four-cycles in the Tanner graph: "we get some inequalities for free!"

Tanner Graphs with/without Four-Cycles

Proposition: It seems to be favorable to have no four-cycles in the Tanner graph: "we get some inequalities for free!"

Note: this argument can be easily extended to Tanner graphs with no six-cycles, no eight-cycles, etc.

Obtaining tighter Relaxations

Let the relaxation $\mathrm{relax}(\mathcal{C})$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_3) \end{array}$$

Therefore,

$$\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_3).$$

How well can we do by adding more (redundand) lines to the parity-check matrix?

Obtaining tighter Relaxations (Part 2)

What about taking a parity-check matrix $\mathbf{H}'$ that contains all the non-zero codewords from the dual code?

$$\mathbf{H}' = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_3) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{12}) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{13}) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{23}) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{123}) \end{array}$$

$$\mathrm{relax}'(\mathcal{C}) \triangleq \mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_3) \cap \mathrm{conv}(\mathcal{C}_{12}) \cap \mathrm{conv}(\mathcal{C}_{13}) \cap \mathrm{conv}(\mathcal{C}_{23}) \cap \mathrm{conv}(\mathcal{C}_{123}).$$

Obtaining tighter Relaxations (Part 3)

Translating a theorem from matroid theory we get the following result:

Theorem (Seymour 1981) We have

$$\mathrm{relax}'(\mathcal{C}) = \mathrm{conv}(\mathcal{C})$$

if and only if there is no way to shorten and puncture $\mathcal{C}$ such that we get the codes F_7^*, $M(K_5)$, or R_{10}.

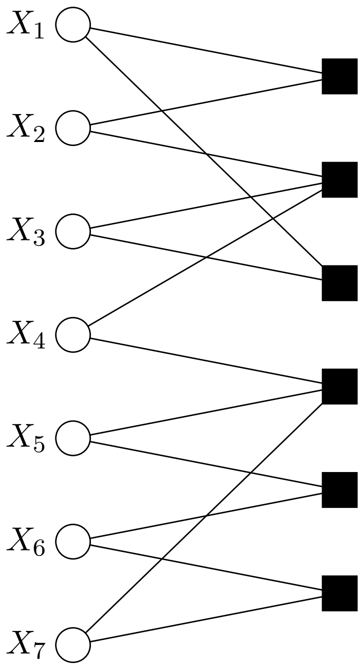
F_7^*:	$[7, 3, 4]$ code
$M(K_5)$:	$[10, 6, 3]$ code
R_{10}:	$[10, 5, 4]$ code

Pseudo-codwords and the edge zeta function

Tanner/Factor Graph of a Cycle Code

Cycle codes are codes which have a Tanner/factor graph where all bit nodes have degree two. (Equivalently, the parity-check matrix has two ones per column.)

Example:

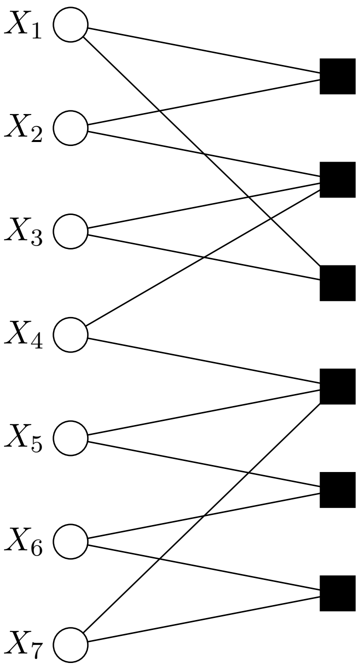


Tanner/factor graph

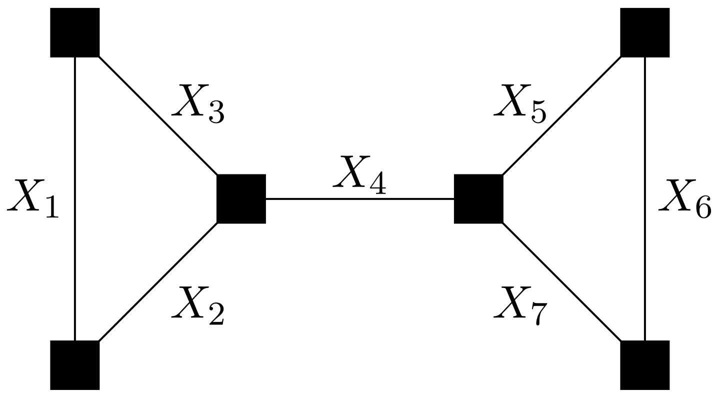
Tanner/Factor Graph of a Cycle Code

Cycle codes are codes which have a Tanner/factor graph where all bit nodes have degree two. (Equivalently, the parity-check matrix has two ones per column.)

Example:



Tanner/factor graph

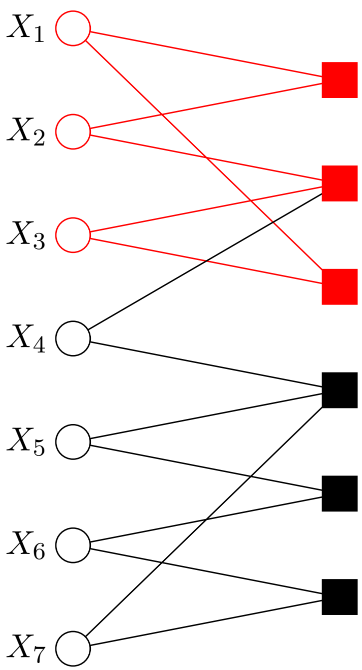


Corresponding normal factor graph

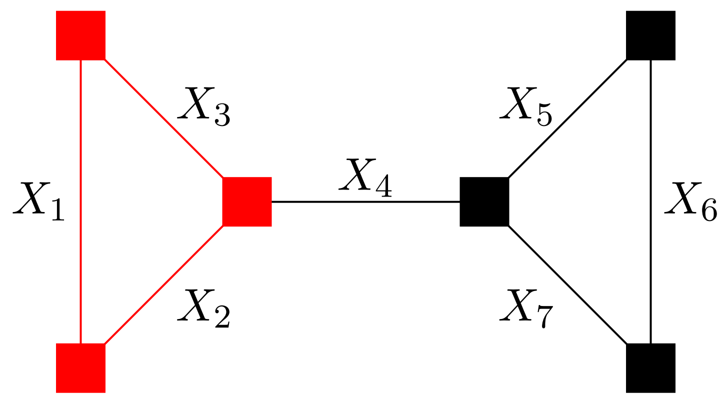
Tanner/Factor Graph of a Cycle Code

Cycle codes are called cycle codes because codewords correspond to simple cycles (or to the symmetric difference set of simple cycles) in the Tanner/factor graph.

Example:



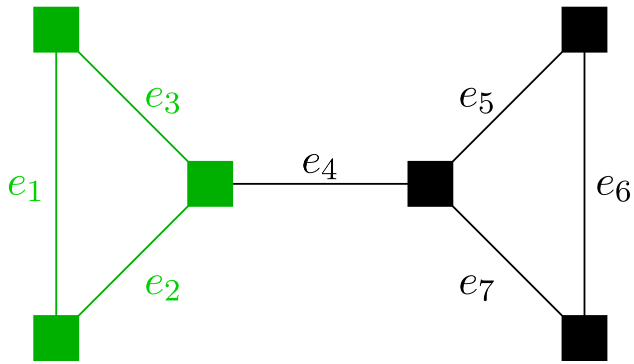
Tanner/factor graph



Corresponding normal factor graph

The Edge Zeta Function of a Graph

Definition (Hashimoto, see also Stark/Terras):



Here: $\Gamma = (e_1, e_2, e_3)$

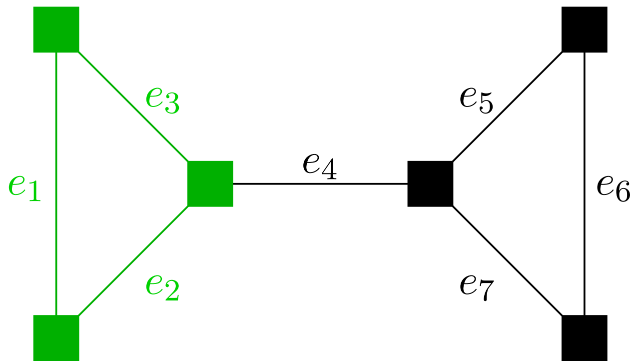
Let Γ be a path in a graph X with edge-set E; write

$$\Gamma = (e_{i_1}, \ldots, e_{i_k})$$

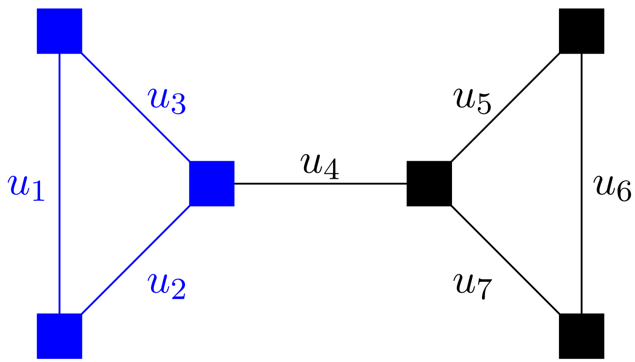
to indicate that Γ begins with the edge e_{i_1} and ends with the edge e_{i_k}.

The Edge Zeta Function of a Graph

Definition (Hashimoto, see also Stark/Terras):



Here: $\Gamma = (e_1, e_2, e_3)$



Here: $g(\Gamma) = u_1 u_2 u_3$

Let Γ be a path in a graph X with edge-set E; write

$$\Gamma = (e_{i_1}, \ldots, e_{i_k})$$

to indicate that Γ begins with the edge e_{i_1} and ends with the edge e_{i_k}.

The monomial of Γ is given by

$$g(\Gamma) \triangleq u_{i_1} \cdots u_{i_k},$$

where the u_i's are indeterminates.

The Edge Zeta Function of a Graph

Definition (Hashimoto, see also Stark/Terras):
The edge zeta function of X is defined to be the power series

$$\zeta_X(u_1, \ldots, u_n) \in \mathbb{Z}[[u_1, \ldots, u_n]]$$

given by

$$\zeta_X(u_1, \ldots, u_n) = \prod_{[\Gamma] \in A(X)} \frac{1}{1 - g(\Gamma)},$$

where $A(X)$ is the collection of equivalence classes of backtrackless, tailless, primitive cycles in X.
Note: unless X contains only one cycle, the set $A(X)$ will be countably infinite.

The Edge Zeta Function of a Graph

Theorem (Bass):

- The edge zeta function $\zeta_X(u_1, \ldots, u_n)$ is a rational function.
- More precisely, for any directed graph $\vec{X}$ of X, we have

$$\zeta_X(u_1, \ldots, u_n) = \frac{1}{\det\left(\mathbf{I} - \mathbf{U}\mathbf{M}(\vec{X})\right)} = \frac{1}{\det\left(\mathbf{I} - \mathbf{M}(\vec{X})\mathbf{U}\right)}$$

  where

  - $\mathbf{I}$ is the identity matrix of size $2n$,
  - $\mathbf{U} = \text{diag}(u_1, \ldots, u_n, u_1, \ldots, u_n)$ is a diagonal matrix of indeterminants.
  - $\mathbf{M}(\vec{X})$ is a $2n \times 2n$ matrix derived from some directed graph version $\vec{X}$ of X.

Relationship Pseudo-Codewords and Edge Zeta Function (Part 1: Theorem)

Theorem:

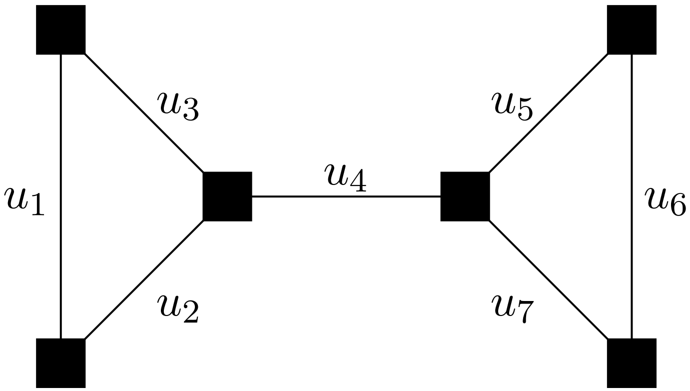
- Let C be a cycle code defined by a parity-check matrix $\mathbf{H}$ having normal graph $N \triangleq N(\mathbf{H})$.
- Let $n = n(N)$ be the number of edges of N.
- Let $\zeta_N(u_1, \ldots, u_n)$ be the edge zeta function of N.
- Then

the monomial $u_1^{p_1} \ldots u_n^{p_n}$ has a nonzero coefficient
in the Taylor series expansion of ζ_N

if and only if

the corresponding exponent vector $(p_1, \ldots, p_n)$
is an unscaled pseudo-codeword for C.

Relationship Pseudo-Codewords and Edge Zeta Function (Part 2: Example)

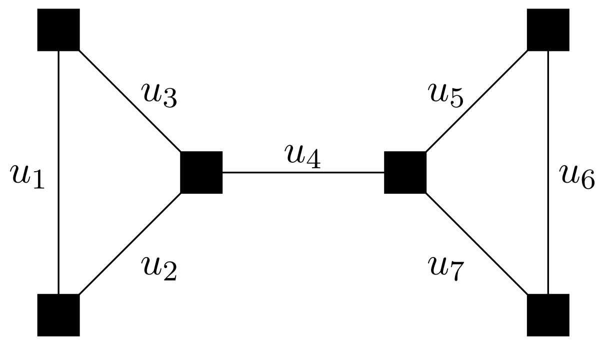


This normal graph N has the following inverse edge zeta function:

$$\zeta_N(u_1,\ldots,u_7) = \frac{1}{\det(\mathbf{I}_{14} - \mathbf{UM})}$$

$$= \frac{1}{\begin{array}{l} 1 - 2u_1u_2u_3 + u_1^2u_2^2u_3^2 - 2u_5u_6u_7 + 4u_1u_2u_3u_5u_6u_7 - 2u_1^2u_2^2u_3^2u_5u_6u_7 \\ -4u_1u_2u_3u_4^2u_5u_6u_7 + 4u_1^2u_2^2u_3^2u_4^2u_5u_6u_7 + u_5^2u_6^2u_7^2 - 2u_1u_2u_3u_5^2u_6^2u_7^2 \\ +u_1^2u_2^2u_3^2u_5^2u_6^2u_7^2 + 4u_1u_2u_3u_4^2u_5^2u_6^2u_7^2 - 4u_1^2u_2^2u_3^2u_4^2u_5^2u_6^2u_7^2 \end{array}}$$

Relationship Pseudo-Codewords and Edge Zeta Function (Part 3: Example)



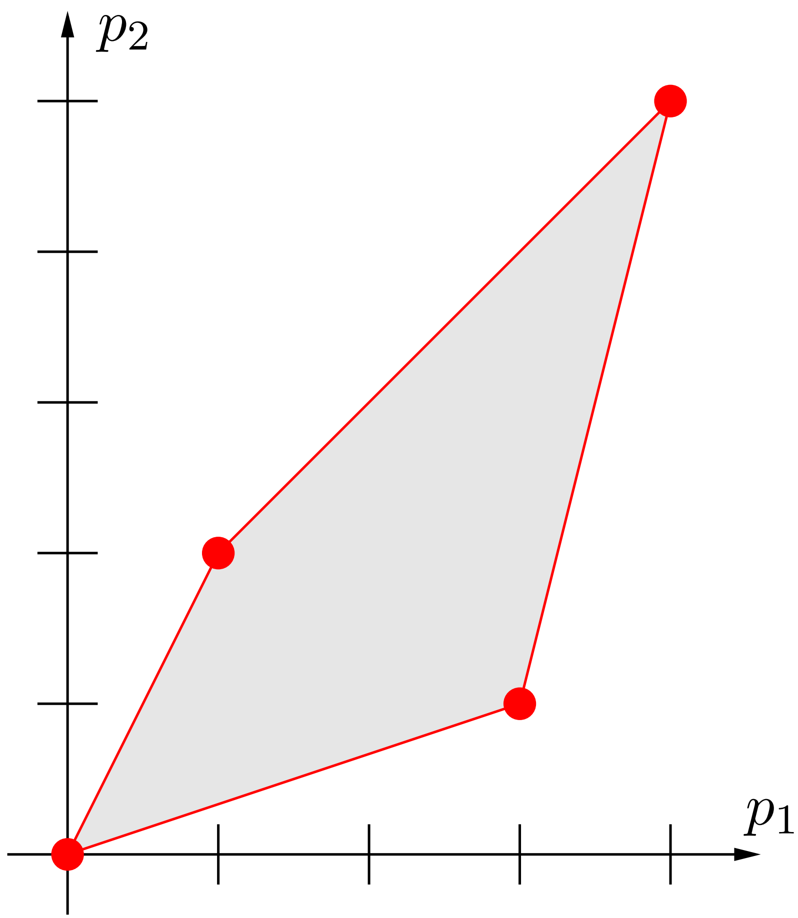
The Taylor series exansion is

$$\begin{aligned}\zeta_N(u_1,\ldots,u_7) &= 1 + 2u_1u_2u_3 + 3u_1^2u_2^2u_3^2 + 2u_5u_6u_7 \\ &\quad + 4u_1u_2u_3u_5u_6u_7 + 6u_1^2u_2^2u_3^2u_5u_6u_7 \\ &\quad + 4u_1u_2u_3u_4^2u_5u_6u_7 + 12u_1^2u_2^2u_3^2u_4^2u_5u_6u_7 \\ &\quad + \cdots\end{aligned}$$

We get the following exponent vectors:

$(0,0,0,0,0,0,0)$	codeword
$(1,1,1,0,0,0,0)$	codeword
$(2,2,2,0,0,0,0)$	pseudo-codeword (in $\mathbb{Z}$-span)
$(0,0,0,0,1,1,1)$	codeword
$(1,1,1,0,1,1,1)$	codeword
$(2,2,2,0,1,1,1)$	pseudo-codeword (in $\mathbb{Z}$-span)
$(1,1,1,2,1,1,1)$	pseudo-codeword (not in $\mathbb{Z}$-span)
$(2,2,2,2,1,1,1)$	pseudo-codeword (in $\mathbb{Z}$-span)

The Newton Polytope of a Polynomial



Here: $P(u_1, u_2)$
$= u_1^0 u_2^0 + 3u_1^1 u_2^2 + 4u_1^3 u_2^1 - 2u_1^4 u_2^5$

Definition:
The Newton polytope of a polynomial $P(u_1, \ldots, u_n)$ in n indeterminates is the convex hull of the points in n-dimensional space given by the exponent vectors of the nonzero monomials appearing in $P(u_1, \ldots, u_n)$.

Similarly, we can associate a polyhedron to a power series.

Characterizing the Fundamental Cone Through the Zeta Function

Collecting the results from the previous slides we get:

Proposition: Let $\mathcal{C}$ be some cycle code with parity-check matrix $\mathbf{H}$ and normal factor graph $N(\mathbf{H})$.

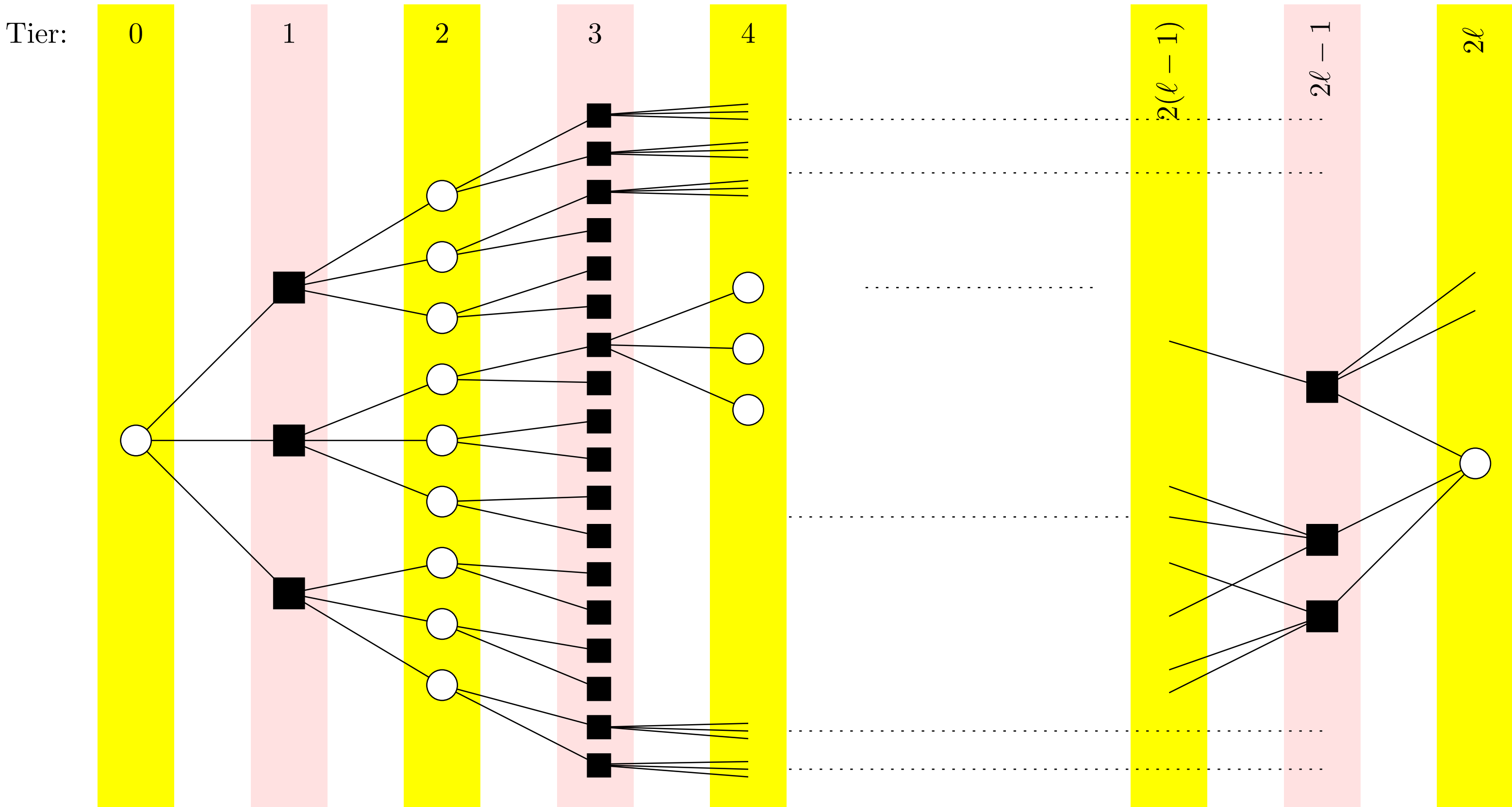
The Newton polyhedron of the zeta function of $N(\mathbf{H})$

equals

the fundamental cone $\mathcal{K}(\mathbf{H})$.

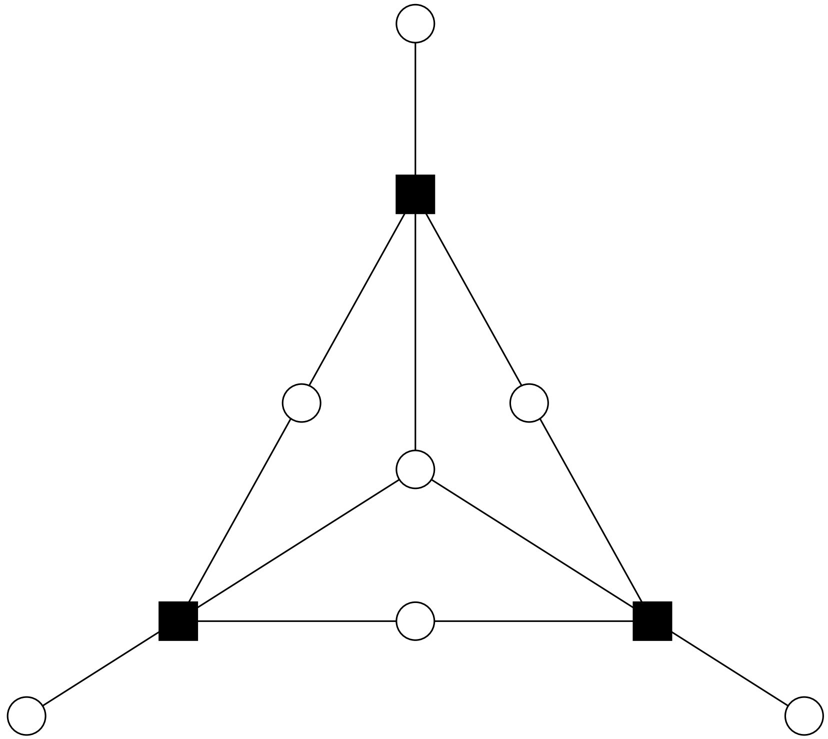
The canonical completion

Trying to Construct a Codeword



Pseudo-Codewords: the Canonical Completion

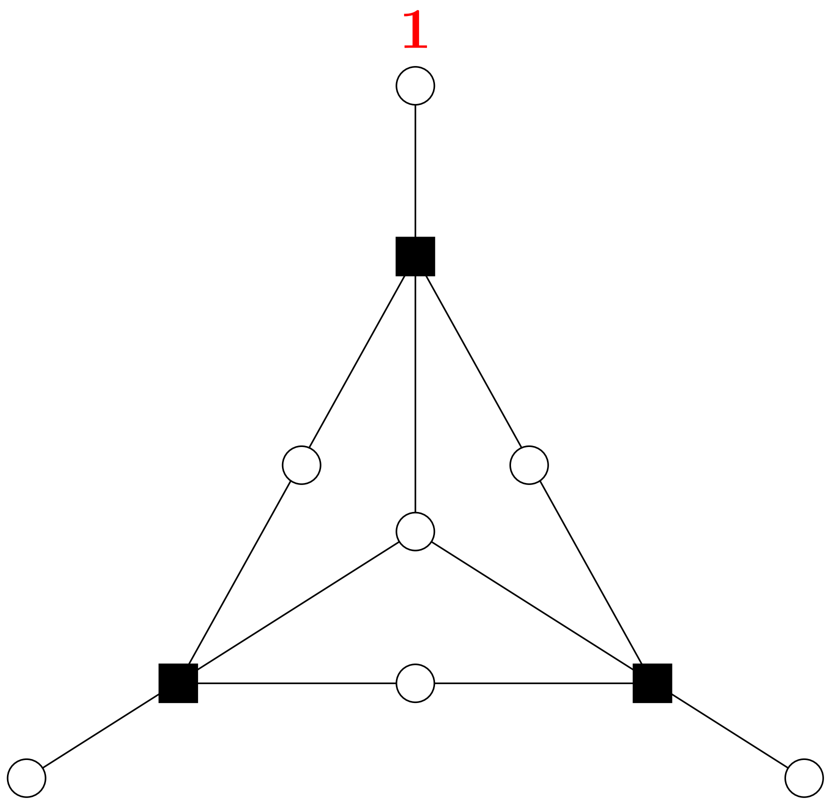
Example: $[7,4,3]$ binary Hamming code.



Note that all checks have degree $k=4$. $\Rightarrow$ completion factor $\frac{1}{k-1}=\frac{1}{3}$.

Pseudo-Codewords: the Canonical Completion

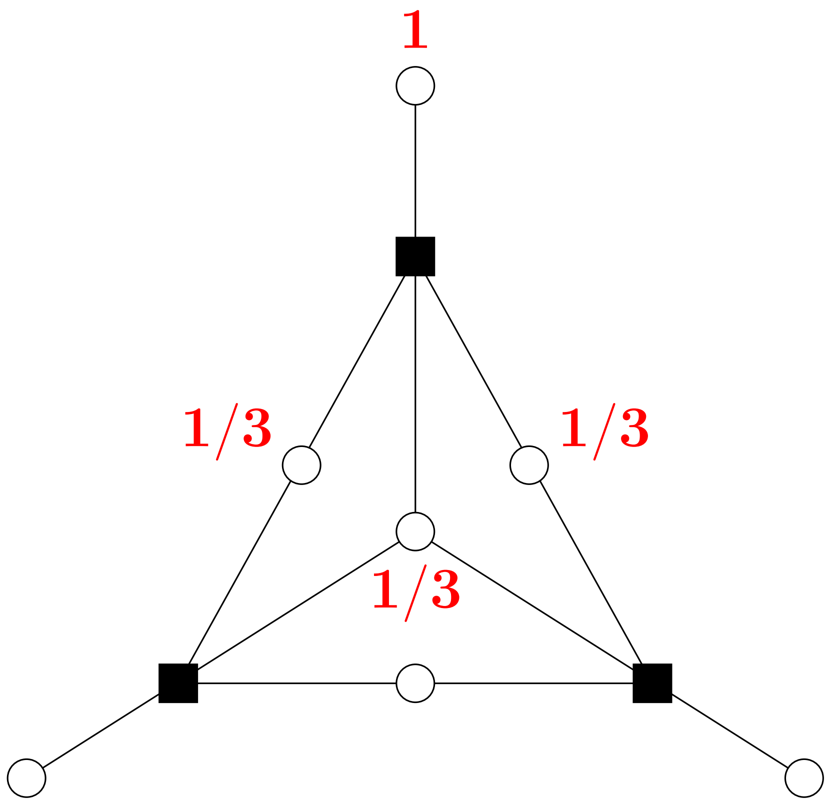
Example: $[7, 4, 3]$ binary Hamming code.



Note that all checks have degree $k = 4$. $\Rightarrow$ completion factor $\frac{1}{k-1} = \frac{1}{3}$.

Pseudo-Codewords: the Canonical Completion

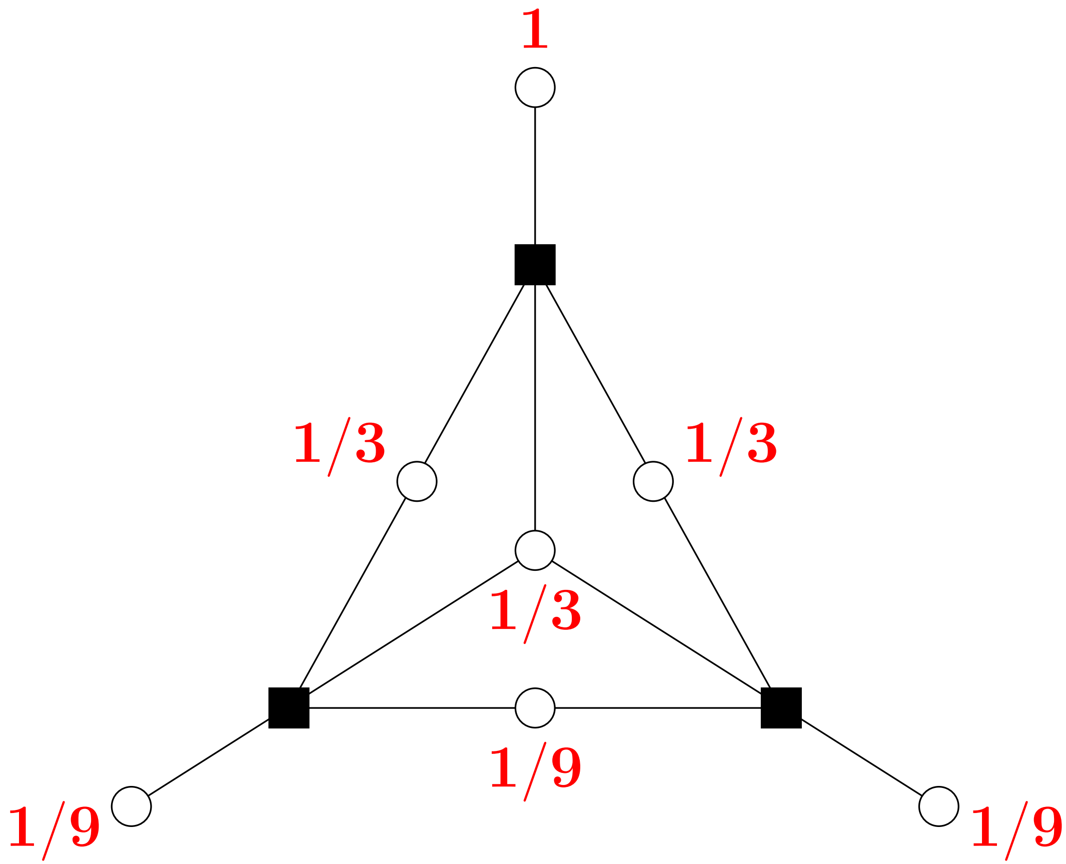
Example: $[7, 4, 3]$ binary Hamming code.



Note that all checks have degree $k = 4$. $\Rightarrow$ completion factor $\frac{1}{k-1} = \frac{1}{3}$.

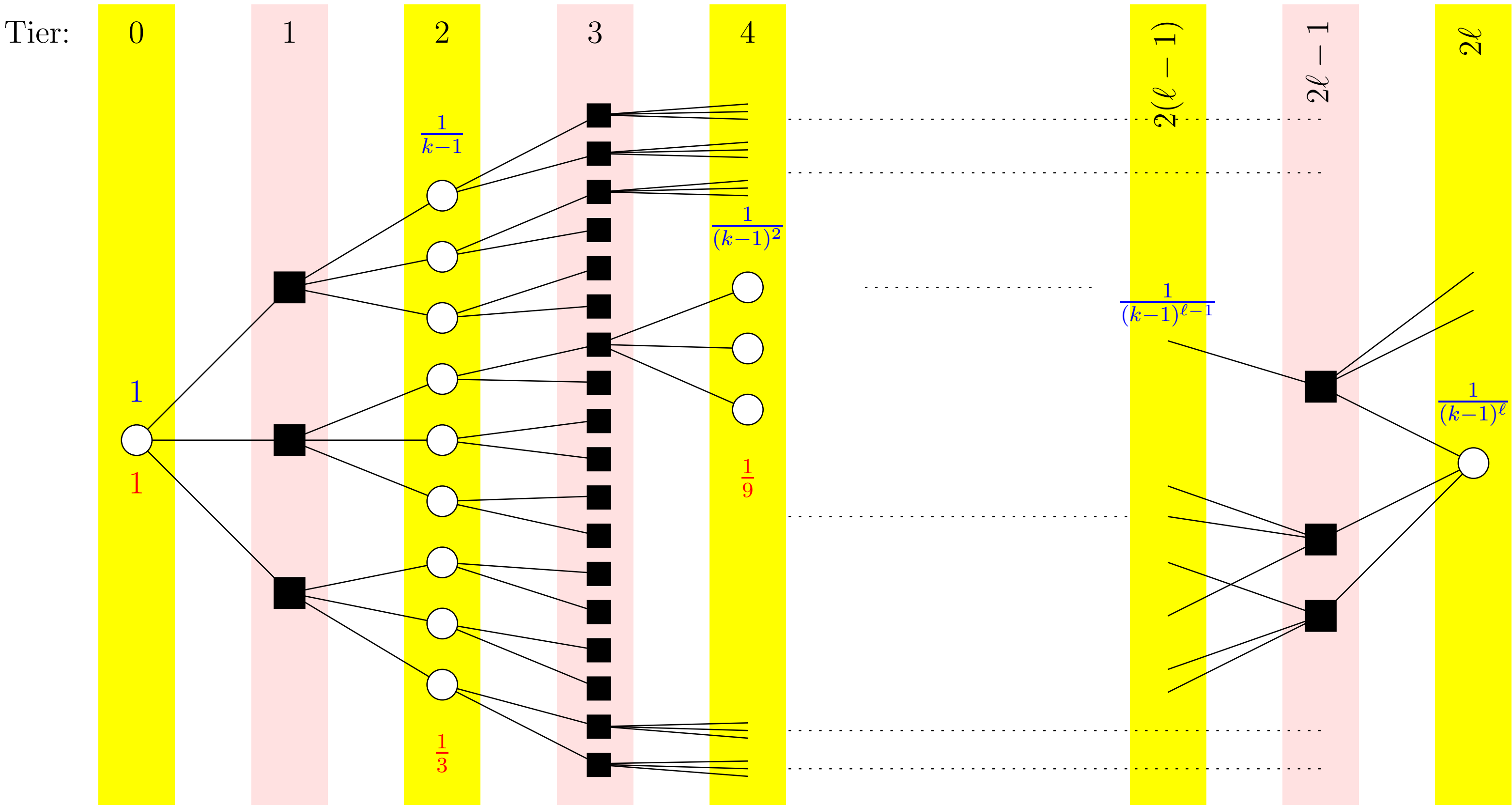
Pseudo-Codewords: the Canonical Completion

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

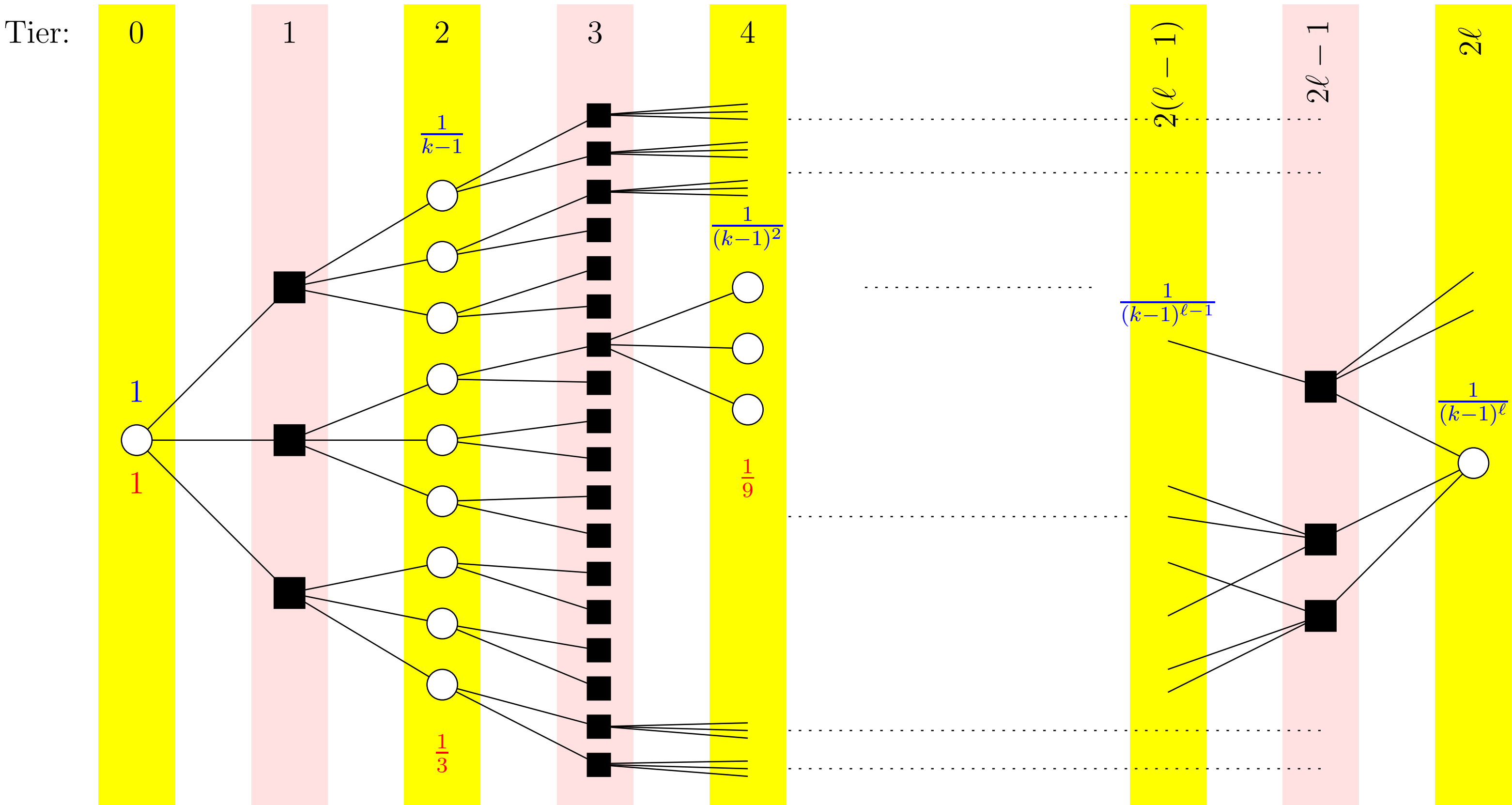


Note that all checks have degree $k = 4$. $\Rightarrow$ completion factor $\frac{1}{k-1} = \frac{1}{3}$.

Pseudo-Codewords: the Canonical Completion



Pseudo-Codewords: the Canonical Completion



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

An Upper Bound on the Minimum Pseudo-Weight based on Can. Compl.

An Upper Bound on the Minimum Pseudo-Weight based on Can. Compl.

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

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

where

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

An Upper Bound on the Minimum Pseudo-Weight based on Can. Compl.

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

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

where

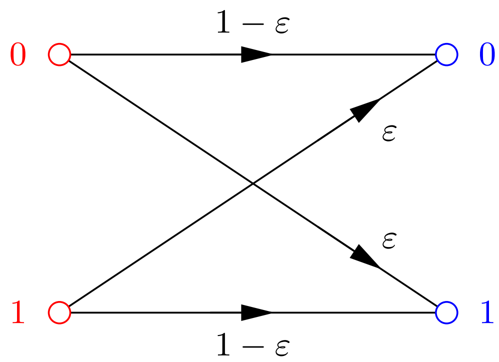
$$\beta'_{j,k} = \left(\frac{j(j-1)}{j-2} \right)^2, \quad \beta_{j,k} = \frac{\log\left((j-1)^2\right)}{\log\left((j-1)(k-1)\right)} < 1.$$

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

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

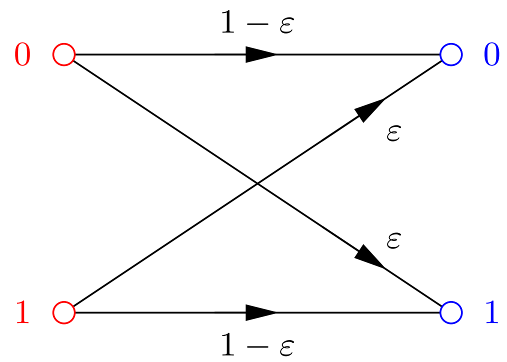
LP decoding thresholds for the BSC

The Binary Symmetric Channel (Part 1)



Let $\varepsilon \in [0,1]$. The binary symmetric channel (BSC) with cross-over probability ε is a discrete memoryless channel

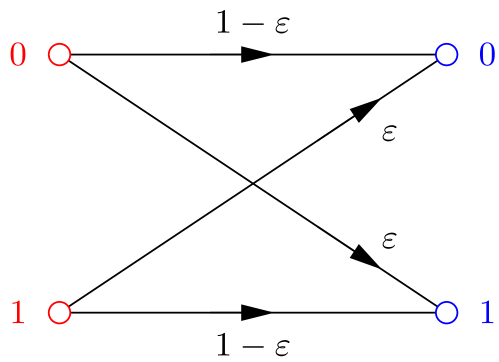
The Binary Symmetric Channel (Part 1)



Let $\varepsilon \in [0, 1]$. The binary symmetric channel (BSC) with cross-over probability ε is a discrete memoryless channel

- with input alphabet $\mathcal{X} = \{0,\ 1\}$,

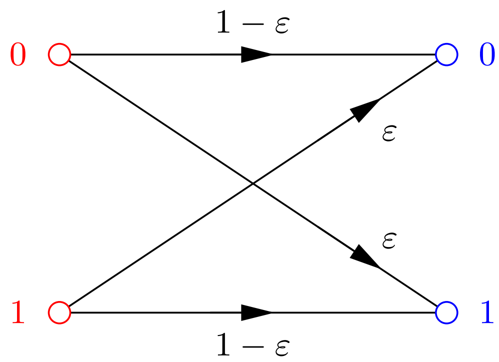
The Binary Symmetric Channel (Part 1)



Let $\varepsilon \in [0, 1]$. The binary symmetric channel (BSC) with cross-over probability ε is a discrete memoryless channel

- with input alphabet $\mathcal{X} = \{0,\ 1\}$,
- with output alphabet $\mathcal{Y} = \{0,\ 1\}$,

The Binary Symmetric Channel (Part 1)

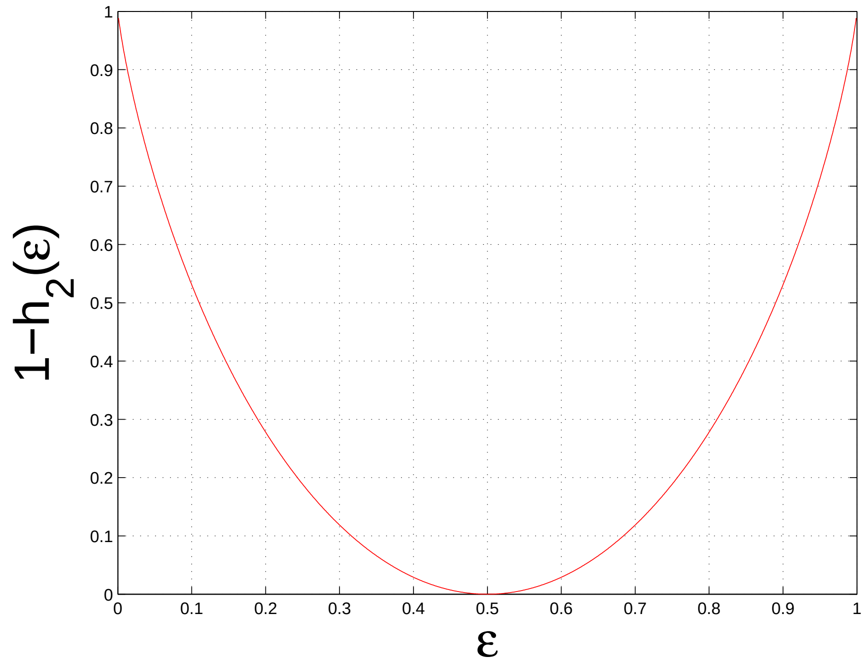


Let $\varepsilon \in [0,1]$. The binary symmetric channel (BSC) with cross-over probability ε is a discrete memoryless channel

- with input alphabet $\mathcal{X} = \{0,\ 1\}$,
- with output alphabet $\mathcal{Y} = \{0,\ 1\}$,
- and with conditional probability mass function

$$P_{Y_i|X_i}(y_i|x_i) = \begin{cases} 1-\varepsilon & (y_i = x_i) \\ \varepsilon & (y_i \neq x_i) \end{cases}.$$

The Binary Symmetric Channel (Part 2)

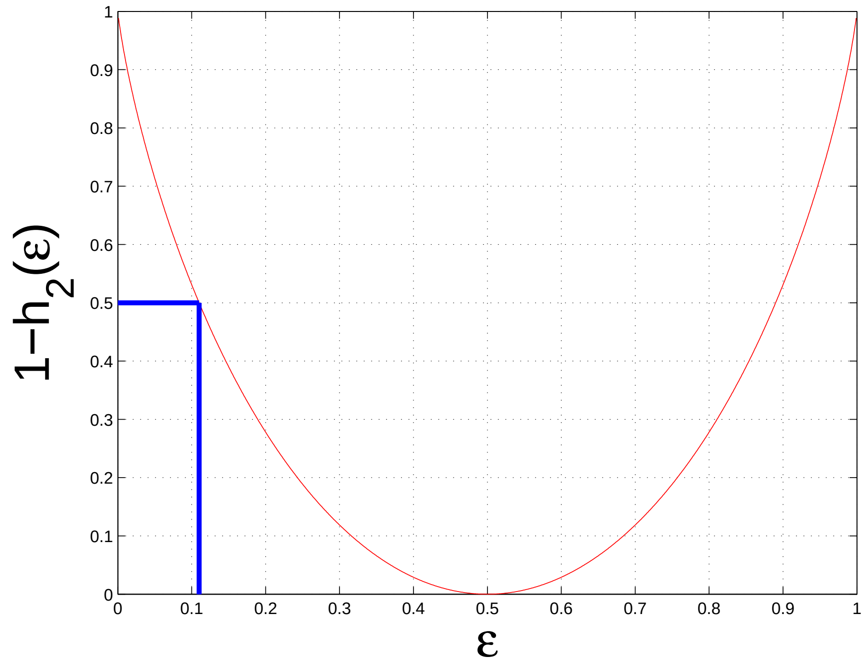


The capacity for the BSC as a function of the cross-over probability ε is

$$C_{\mathrm{BSC}} = 1 - h_2(\varepsilon),$$

where $h_2(\varepsilon) \triangleq -\varepsilon \log_2(\varepsilon) - (1-\varepsilon)\log_2(1-\varepsilon)$.

The Binary Symmetric Channel (Part 2)



The capacity for the BSC as a function of the cross-over probability ε is

$$C_{\text{BSC}} = 1 - h_2(\varepsilon),$$

where $h_2(\varepsilon) \triangleq -\varepsilon \log_2(\varepsilon) - (1-\varepsilon)\log_2(1-\varepsilon)$.

The Binary Symmetric Channel (Part 3)

Assume that the channel is a BSC with cross-over probability ε.

The Binary Symmetric Channel (Part 3)

Assume that the channel is a BSC with cross-over probability ε.

Channel capacity:

- Channel coding theorem

- Converse to the channel coding theorem

The Binary Symmetric Channel (Part 3)

Assume that the channel is a BSC with cross-over probability ε.

Channel capacity:

- Channel coding theorem
  (Gallager's random coding error exponent, etc.)
- Converse to the channel coding theorem
  (Fano's inequality, etc.)

The Binary Symmetric Channel (Part 3)

Assume that the channel is a BSC with cross-over probability ε.

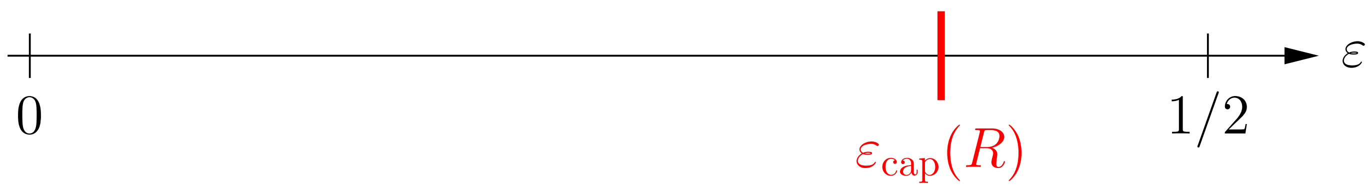
Channel capacity:

- Channel coding theorem

  (Gallager's random coding error exponent, etc.)

- Converse to the channel coding theorem

  (Fano's inequality, etc.)



Important: we are allowed to use the best available coding and decoding schemes for a given rate R.

The Binary Symmetric Channel (Part 4)

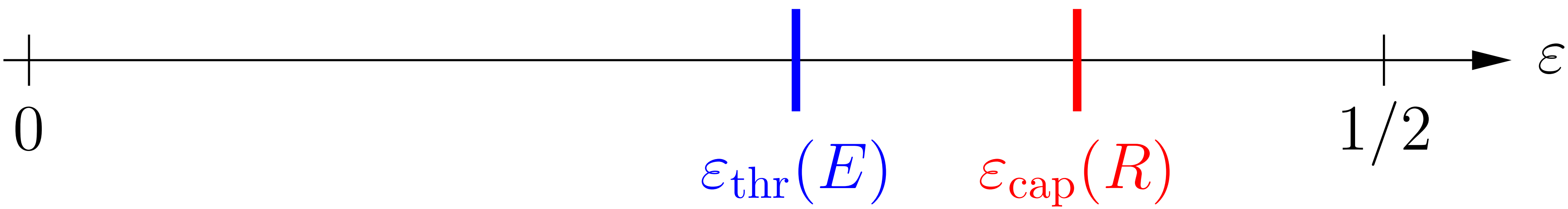
Assume that the channel is a BSC with cross-over probability ε.

Additionally, assume that we put restrictions on the coding schemes and/or on the decoding schemes.

The Binary Symmetric Channel (Part 4)

Assume that the channel is a BSC with cross-over probability ε.
Additionally, assume that we put restrictions on the coding schemes and/or on the decoding schemes.

$\Rightarrow$ Thresholds.

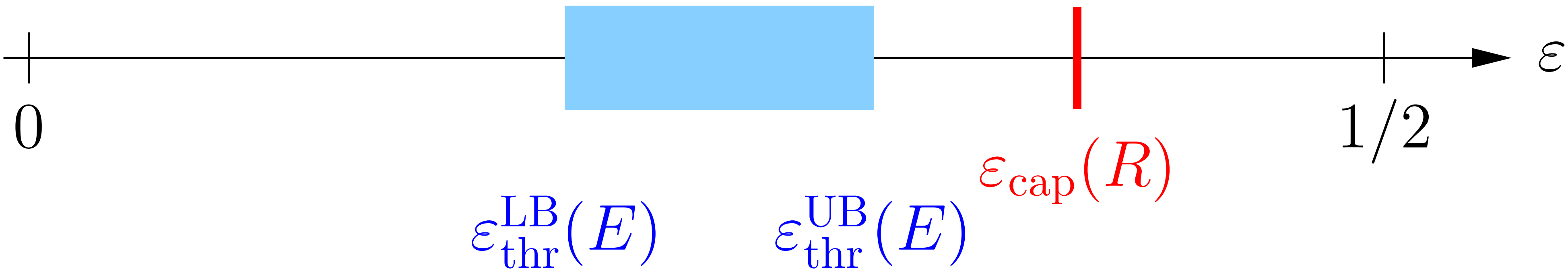


The Binary Symmetric Channel (Part 4)

Assume that the channel is a BSC with cross-over probability ε.

Additionally, assume that we put restrictions on the coding schemes and/or on the decoding schemes.

$\Rightarrow$ Thresholds.



Existence of LP Decoding Thresholds

- A priori it is not clear for what families/ensembles of codes there is an LP decoding threshold.

Existence of LP Decoding Thresholds

- A priori it is not clear for what families/ensembles of codes there is an LP decoding threshold.
- The tight connection between min-sum algorithm decoding and LP decoding suggests that families/ensembles that have a threshold under min-sum algorithm decoding also have a threshold under LP decoding.

Existence of LP Decoding Thresholds

- A priori it is not clear for what families/ensembles of codes there is an LP decoding threshold.
- The tight connection between min-sum algorithm decoding and LP decoding suggests that families/ensembles that have a threshold under min-sum algorithm decoding also have a threshold under LP decoding.
- [Koetter:Vontobel:06]: there is an LP decoding threshold for $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC codes where $2 < w_{\mathrm{col}} < w_{\mathrm{row}}$.

BSC: An Upper Bound on the Threshold (Part 1)

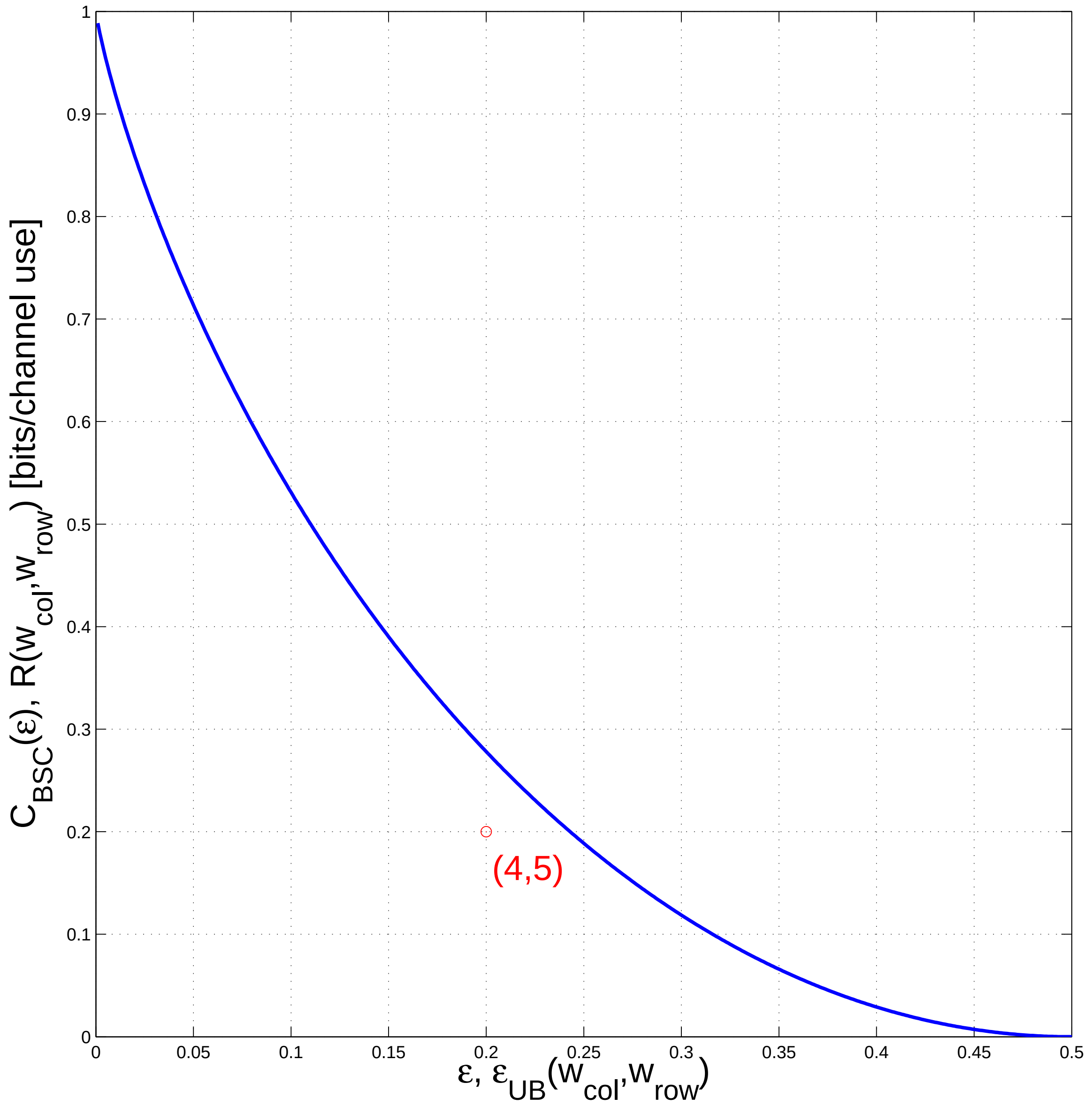
Theorem:

- Consider a family of $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular codes of increasing block length n.
- Consider a BSC with cross-over probability ε.
- In the limit $n \to \infty$, if

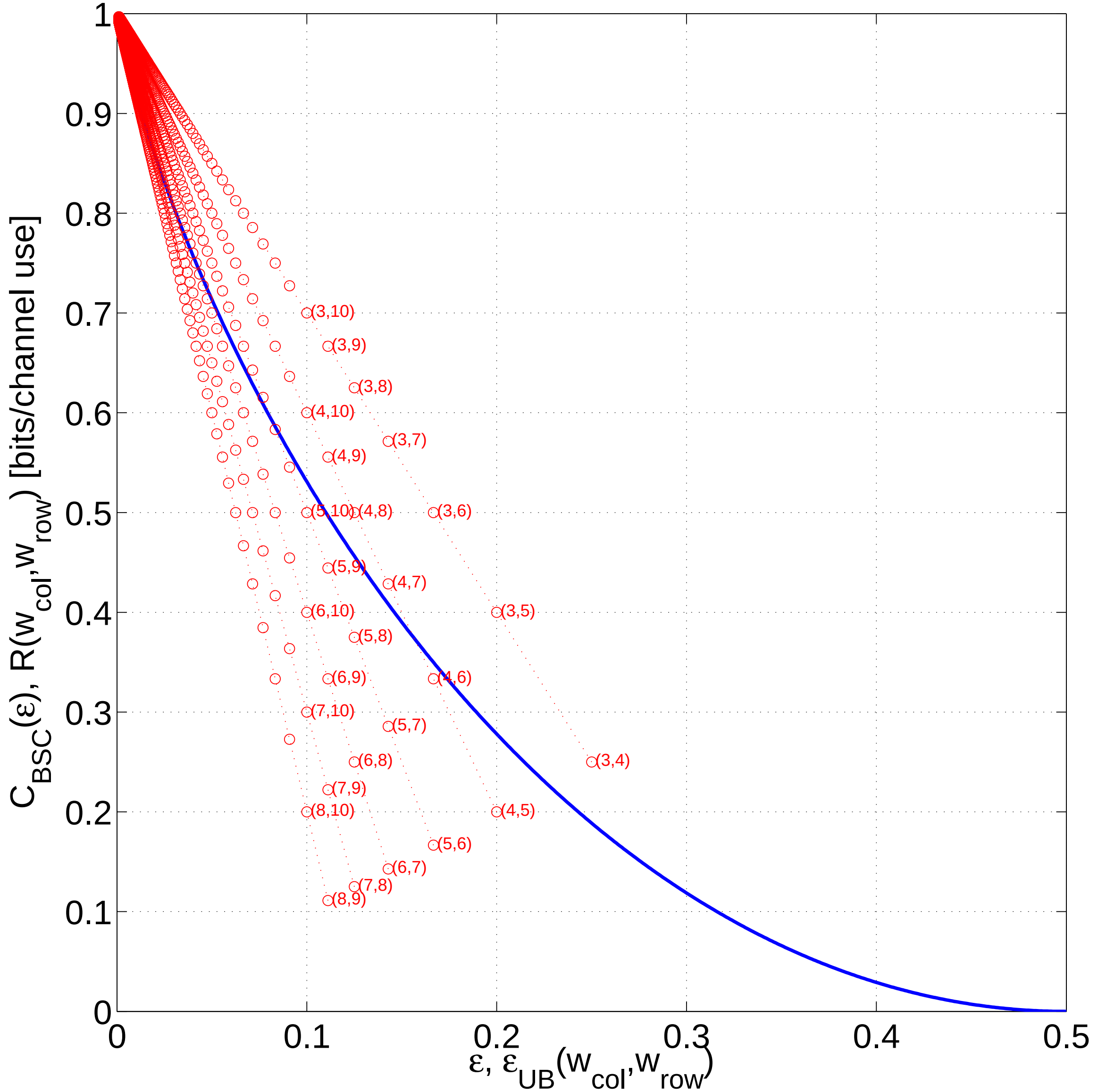
$$\varepsilon > \frac{1}{w_{\mathrm{row}}}$$

then with probability 1 the LP decoder will not decode to the transmitted codeword.

BSC: An Upper Bound on the Threshold (Part 2)



BSC: An Upper Bound on the Threshold (Part 2)



BSC: An Upper Bound on the Threshold (Part 3)

Theorem: Consider a family of codes where the minimal row-degree goes to $w_{\mathrm{row}}^{\mathrm{min}}(\infty)$ when $n \to \infty$ and a BSC with cross-over probability ε. In the limit $n \to \infty$, if

$$\varepsilon > \frac{1}{w_{\mathrm{row}}^{\mathrm{min}}(\infty)}$$

then with probability 1 the LP decoder will not decode to the transmitted codeword.

BSC: An Upper Bound on the Threshold (Part 3)

Theorem: Consider a family of codes where the minimal row-degree goes to $w_{\text{row}}^{\min}(\infty)$ when $n \to \infty$ and a BSC with cross-over probability ε. In the limit $n \to \infty$, if

$$\varepsilon > \frac{1}{w_{\text{row}}^{\min}(\infty)}$$

then with probability 1 the LP decoder will not decode to the transmitted codeword.

Corollary: For any family of codes where $w_{\text{row}}^{\min}(n)$ grows unboundedly, i.e. where

$$\lim_{n \to \infty} w_{\text{row}}^{\min}(n) = \infty,$$

the above right-hand side expression goes to 0.

BSC: An Upper Bound on the Threshold (Proof)

Linear programming (LP) decoding:

$$\hat{\boldsymbol{\omega}} = \arg \min_{\omega \in \mathcal{P}(\mathbf{H})} \sum_{i=1}^{n} \lambda_i \omega_i .$$

BSC: An Upper Bound on the Threshold (Proof)

Linear programming (LP) decoding:

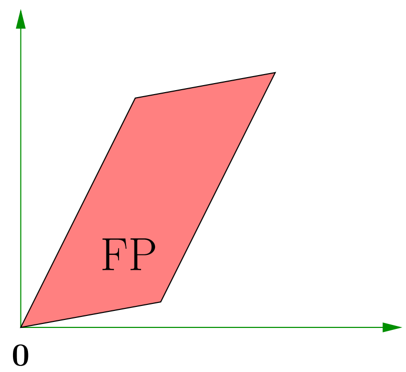
$$\hat{\boldsymbol{\omega}} = \arg \min_{\boldsymbol{\omega} \in \mathcal{P}(\mathbf{H})} \sum_{i=1}^{n} \lambda_i \omega_i .$$

Assume that the zero codeword has been sent. LP decoding does not decide for the all-zeros codeword if there is a vector

$$\boldsymbol{\omega} \in \mathcal{P}(\mathbf{H}) \setminus \{\mathbf{0}\}$$

such that

$$\sum_{i=1}^{n} \lambda_i \omega_i < 0.$$



BSC: An Upper Bound on the Threshold (Proof)

Linear programming (LP) decoding:

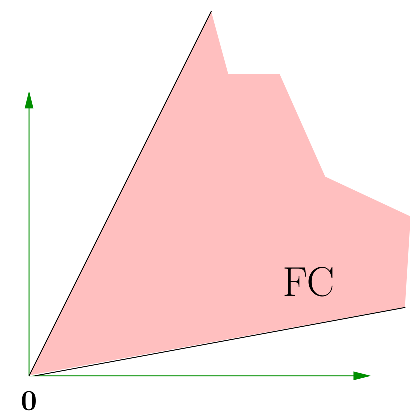
$$\hat{\boldsymbol{\omega}} = \arg \min_{\boldsymbol{\omega} \in \mathcal{P}(\mathbf{H})} \sum_{i=1}^{n} \lambda_i \omega_i .$$

Assume that the zero codeword has been sent. LP decoding does not decides for the all-zeros codeword if there is a vector

$$\boldsymbol{\omega} \in \mathcal{K}(\mathbf{H}) \setminus \{\mathbf{0}\}$$

such that

$$\sum_{i=1}^{n} \lambda_i \omega_i < 0.$$



BSC: An Upper Bound on the Threshold (Proof)

- Assume that we have a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC code.

BSC: An Upper Bound on the Threshold (Proof)

- Assume that we have a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC code.
- Moreover, let $\boldsymbol{\omega} \in \mathbb{R}^n$ be a vector with the following entries:

$$\omega_i \triangleq \begin{cases} \frac{1}{w_{\mathrm{row}}-1} & \text{if } \lambda_i \geq 0 \\ 1 & \text{if } \lambda_i < 0 \end{cases}.$$

BSC: An Upper Bound on the Threshold (Proof)

- Assume that we have a $(w_{\text{col}}, w_{\text{row}})$-regular LDPC code.
- Moreover, let $\boldsymbol{\omega} \in \mathbb{R}^n$ be a vector with the following entries:

$$\omega_i \triangleq \begin{cases} \frac{1}{w_{\text{row}}-1} & \text{if } \lambda_i \geq 0 \\ 1 & \text{if } \lambda_i < 0 \end{cases}.$$

One can easily verify that $\boldsymbol{\omega} \in \mathcal{K}(\mathbf{H})$.

BSC: An Upper Bound on the Threshold (Proof)

- Assume that we have a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC code.
- Moreover, let $\boldsymbol{\omega} \in \mathbb{R}^n$ be a vector with the following entries:

$$\omega_i \triangleq \begin{cases} \frac{1}{w_{\mathrm{row}}-1} & \text{if } \lambda_i \geq 0 \\ 1 & \text{if } \lambda_i < 0 \end{cases}.$$

One can easily verify that $\boldsymbol{\omega} \in \mathcal{K}(\mathbf{H})$.

Note: this pseudo-codeword construction is inspired by the canonical completion contruction.

BSC: An Upper Bound on the Threshold (Proof)

- Assume that we have a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC code.
- Moreover, let $\boldsymbol{\omega} \in \mathbb{R}^n$ be a vector with the following entries:

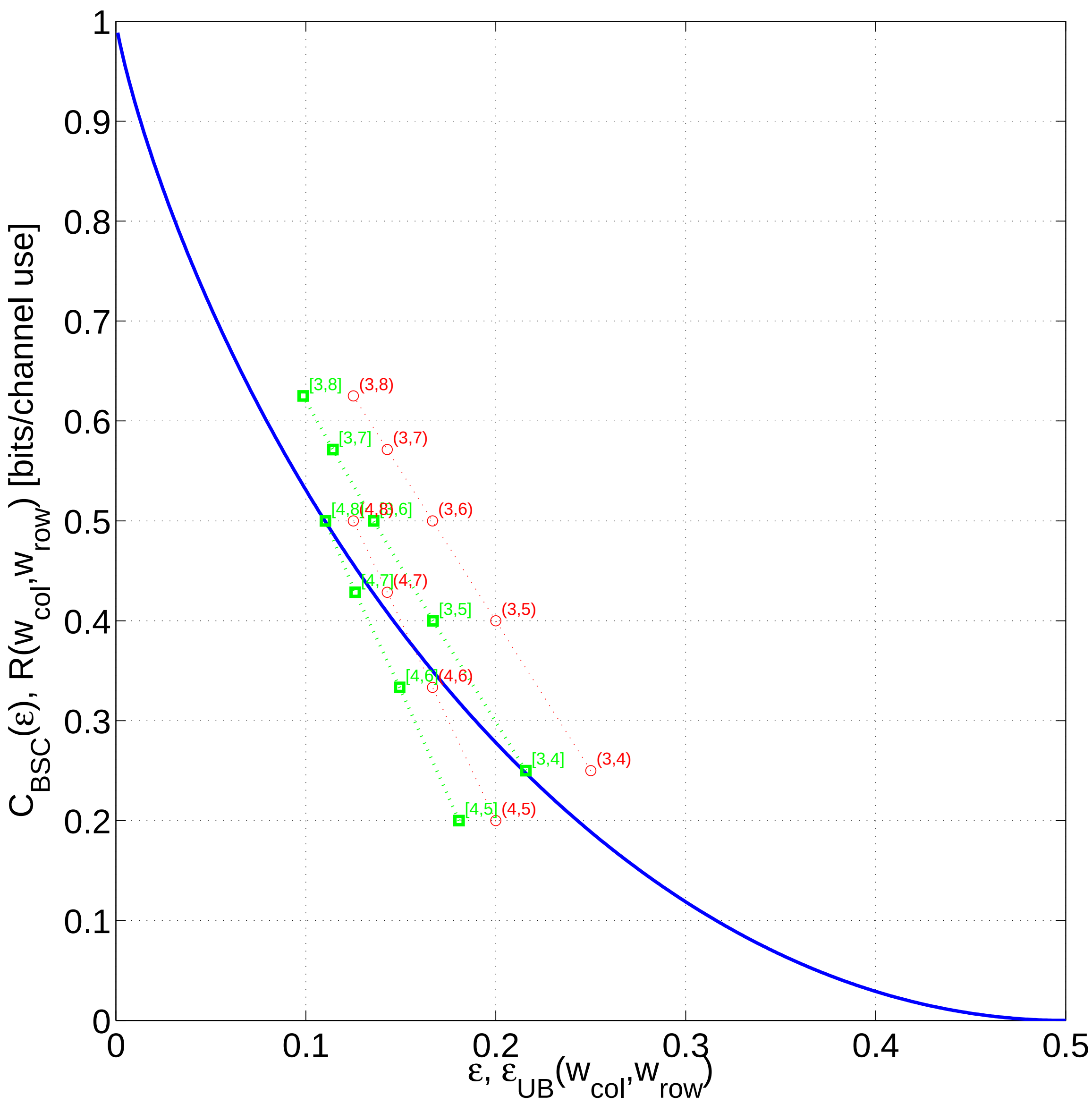
$$\omega_i \triangleq \begin{cases} \frac{1}{w_{\mathrm{row}}-1} & \text{if } \lambda_i \geq 0 \\ 1 & \text{if } \lambda_i < 0 \end{cases}.$$

  One can easily verify that $\boldsymbol{\omega} \in \mathcal{K}(\mathbf{H})$.

  Note: this pseudo-codeword construction is inspired by the canonical completion contruction.

- In the rest of the proof, one shows for which ε this pseudo-codeword leads to a decoding error (details omitted).

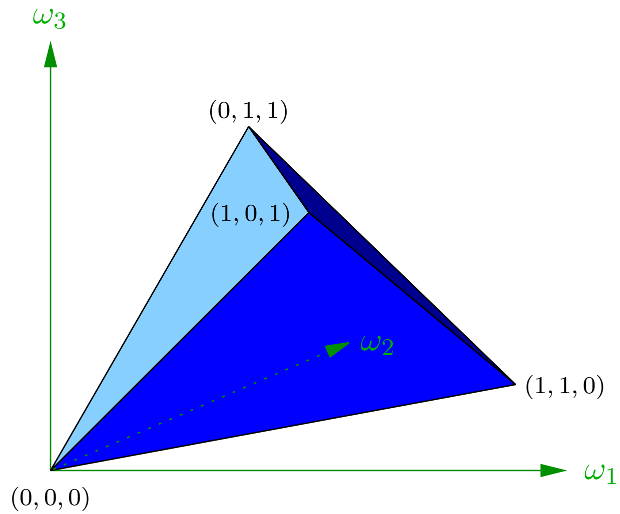
2-Neighborhood-Based Bounds on the Threshold



The fundamental polytope

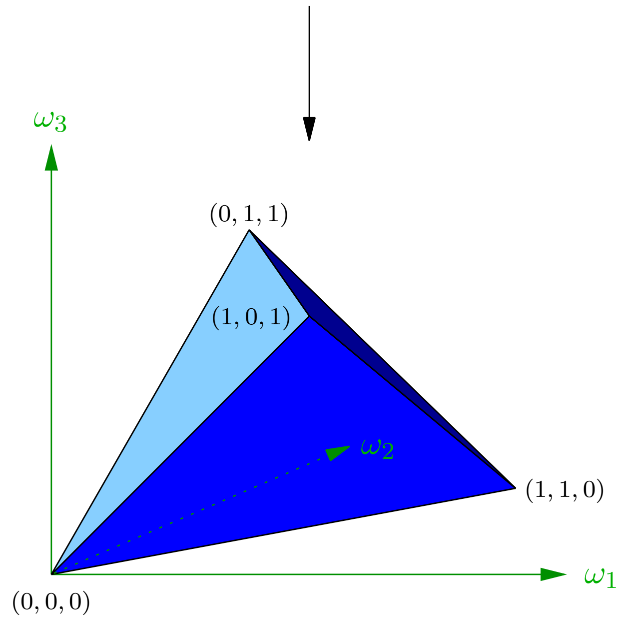
in various contexts

The Fundamental Polytope in Various Contexts



The Fundamental Polytope in Various Contexts

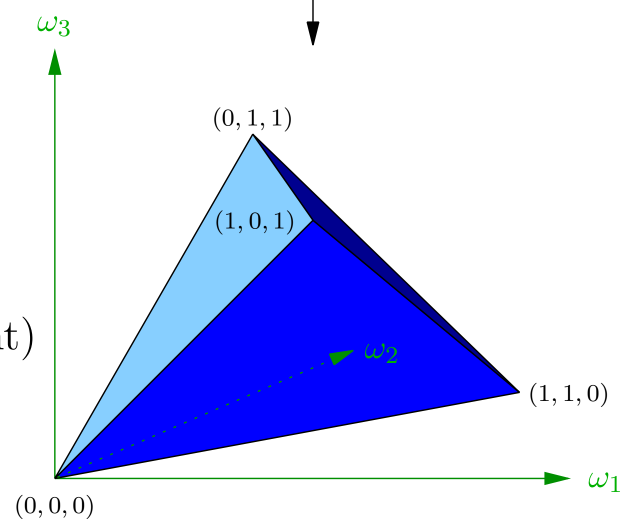
Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)



The Fundamental Polytope in Various Contexts

Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)

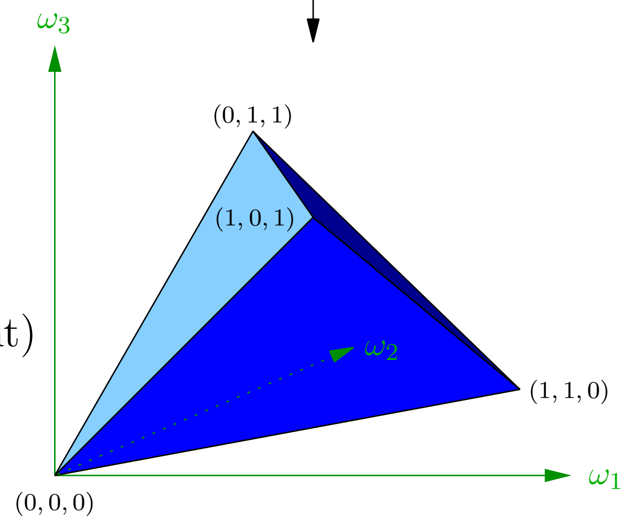
LP Decoding
(Feldman/Karger/Wainwright)
(CISS 2003)



The Fundamental Polytope in Various Contexts

Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)

LP Decoding
(Feldman/Karger/Wainwright)
(CISS 2003)



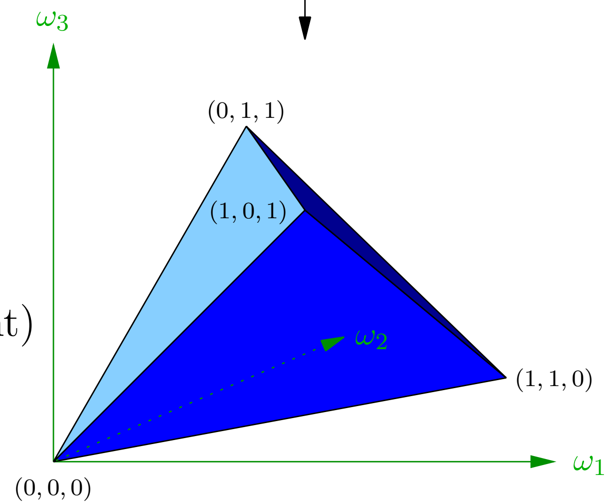
Bethe free energy
(Yedidia/Freeman/Weiss)
(MERL Rep. 2001)

The Fundamental Polytope in Various Contexts

Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)

LP Decoding

(Feldman/Karger/Wainwright)
(CISS 2003)



Bethe free energy

(Yedidia/Freeman/Weiss)
(MERL Rep. 2001)

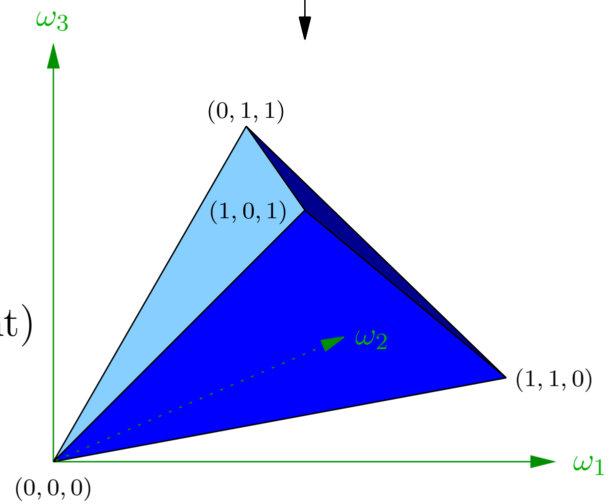
Fundamental cone of cycle codes is the Newton polyhedron
of the edge zeta function of normal factor graph
(Koetter/Li/Vontobel/Walker, ITW2004)

The Fundamental Polytope in Various Contexts

Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)

LP Decoding

(Feldman/Karger/Wainwright)
(CISS 2003)



Bethe free energy

(Yedidia/Freeman/Weiss)
(MERL Rep. 2001)

"The Tropical Geometry of Statistical Models"
Sturmfels, plenary talk at Allerton 2004

Fundamental cone of cycle codes is the Newton polyhedron
of the edge zeta function of normal factor graph
(Koetter/Li/Vontobel/Walker, ITW2004)

References

- More details: see the appendices.
- Papers listed at **`www.pseudocodewords.info`**


Thank you!

Appendices

Communication systems and Shannon's channel coding theorem

Communication System (Part 1)

Communication System (Part 1)

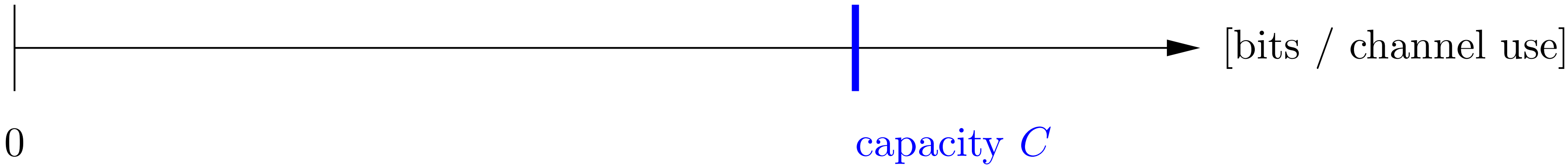
Communication System (Part 1)

Shannon (1948): it is a good idea to use channel codes!

Communication System (Part 1)

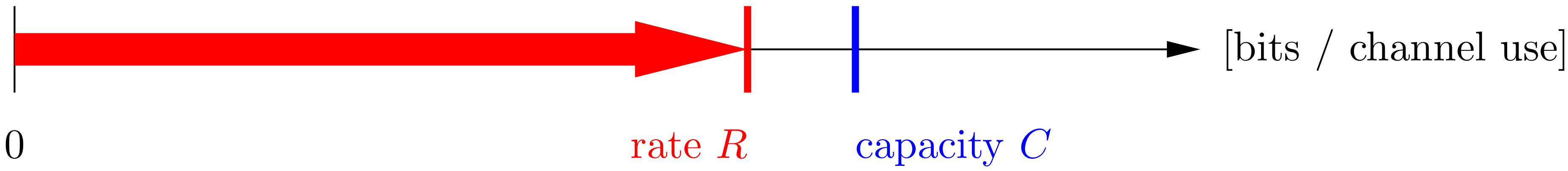
Shannon (1948): it is a good idea to use channel codes!

Communication System (Part 2)



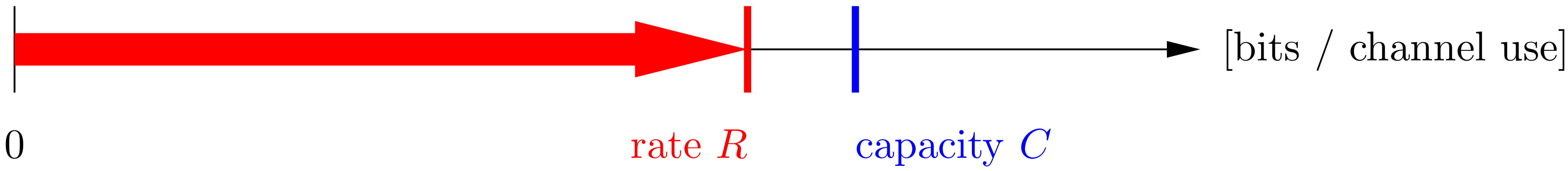
- A channel is characterized by a number C called the capacity.

Communication System (Part 2)



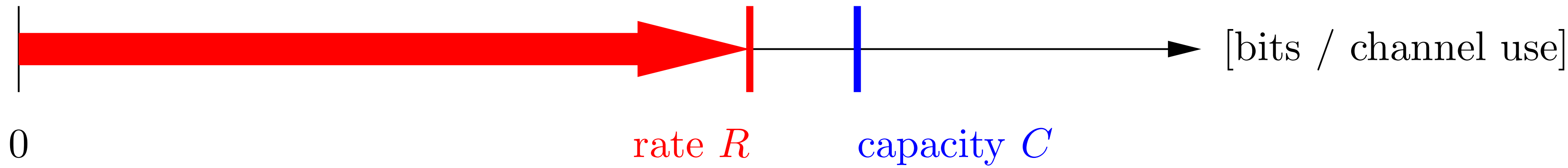
- A channel is characterized by a number C called the capacity.
- A code is characterized by a number R called the rate.

Communication System (Part 2)



- A channel is characterized by a number C called the capacity.
- A code is characterized by a number R called the rate.
- If $R < C$: there are codes, encoders, and decoders such that arbitrarily low error probabilities can be guaranteed (as long as one allows arbitrarily long codes).

Communication System (Part 2)



- A channel is characterized by a number C called the capacity.
- A code is characterized by a number R called the rate.
- If $R < C$: there are codes, encoders, and decoders such that arbitrarily low error probabilities can be guaranteed (as long as one allows arbitrarily long codes).
- Shannon's proof was though non-constructive, i.e. it was not clear at all how to obtain specific well-performing finite-length codes that possess efficient encoders and decoders.

"Traditional" vs. "Modern" Coding and Decoding

	Code design		Decoding
"Traditional"	Reed-Solomon codes etc.	$\longrightarrow$	**?**
"Modern"	**?**	$\longrightarrow$	Message-passing iterative decoding LP decoding

“Traditional” vs. “Modern” Coding and Decoding

	Code design		**Decoding**
”Traditional”	Reed-Solomon codes etc.	$\longrightarrow$	Berlekamp-Massey decoder etc.
”Modern”	**?**	$\longrightarrow$	Message-passing iterative decoding LP decoding

"Traditional" vs. "Modern" Coding and Decoding

	Code design		Decoding
"Traditional"	Reed-Solomon codes etc.	→	Berlekamp-Massey decoder etc.
"Modern"	Codes on Graphs (LDPC/Turbo codes, etc.)	→	Message-passing Iterative decoding LP decoding

“Traditional” vs. “Modern” Coding and Decoding

	Code design		**Decoding**
”Traditional”	Reed-Solomon codes etc.	⟶	Berlekamp-Massey decoder etc.
”Modern”	Codes on Graphs (LDPC/Turbo codes, etc.)	⟶	Message-passing Iterative decoding LP decoding

In both “traditional” and “modern” coding theory, “structure” is an important keyword. By imposing structural constraints

- one usually loses somewhat in generality;
- however, (mathematical) tools become available that can yield big analytical and practical gains.

Communication Model (Part 1)

Information word: $\mathbf{u} = (u_1, \ldots, u_k) \in \mathcal{U}^k$

Sent codeword: $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C} \subseteq \mathcal{X}^n$

Received word: $\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$

Communication Model (Part 1)

Information word: $\mathbf{u} = (u_1, \ldots, u_k) \in \mathcal{U}^k$

Sent codeword: $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C} \subseteq \mathcal{X}^n$

Received word: $\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$

Decoding: Based on $\mathbf{y}$ we would like to estimate the transmitted codeword $\hat{\mathbf{x}}$ or the information word $\hat{\mathbf{u}}$.

Communication Model (Part 1)

Information word:	$\mathbf{u} = (u_1, \ldots, u_k) \in \mathcal{U}^k$
Sent codeword:	$\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C} \subseteq \mathcal{X}^n$
Received word:	$\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$

Decoding: Based on $\mathbf{y}$ we would like to estimate the transmitted codeword $\hat{\mathbf{x}}$ or the information word $\hat{\mathbf{u}}$.

Depending on what criterion we optimize, we obtain different decoding algorithms.

Communication Model (Part 2)

- Min. the block error prob. results in block-wise MAP decoding

$$\hat{\mathbf{u}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{u}\in\mathcal{U}^k} P_{\mathbf{U}|\mathbf{Y}}(\mathbf{u}|\mathbf{y}) = \operatorname*{argmax}_{\mathbf{u}\in\mathcal{U}^k} P_{\mathbf{U},\mathbf{Y}}(\mathbf{u},\mathbf{y}).$$

Communication Model (Part 2)

- Min. the block error prob. results in block-wise MAP decoding

$$\hat{\mathbf{u}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{u}\in\mathcal{U}^k} P_{\mathbf{U}|\mathbf{Y}}(\mathbf{u}|\mathbf{y}) = \operatorname*{argmax}_{\mathbf{u}\in\mathcal{U}^k} P_{\mathbf{U},\mathbf{Y}}(\mathbf{u},\mathbf{y}).$$

- This can also be written as

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}^n} P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}^n} P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}).$$

Communication Model (Part 2)

- Min. the block error prob. results in block-wise MAP decoding

$$\hat{\mathbf{u}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{u}\in\mathcal{U}^k} P_{\mathbf{U}|\mathbf{Y}}(\mathbf{u}|\mathbf{y}) = \operatorname*{argmax}_{\mathbf{u}\in\mathcal{U}^k} P_{\mathbf{U},\mathbf{Y}}(\mathbf{u},\mathbf{y}).$$

- This can also be written as

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}^n} P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}^n} P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}).$$

- If all codewords are equally likely then

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}^n} P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \operatorname*{argmax}_{\mathbf{x}\in\mathcal{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$$

Communication Model (Part 2)

- Min. the block error prob. results in block-wise MAP decoding

$$\hat{\mathbf{u}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\ P_{\mathbf{U}|\mathbf{Y}}(\mathbf{u}|\mathbf{y}) = \underset{\mathbf{u}\in\mathcal{U}^k}{\operatorname{argmax}}\ P_{\mathbf{U},\mathbf{Y}}(\mathbf{u},\mathbf{y}).$$

- This can also be written as

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\ P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\ P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}).$$

- If all codewords are equally likely then

$$\hat{\mathbf{x}}^{\text{block}}_{\text{MAP}}(\mathbf{y}) = \underset{\mathbf{x}\in\mathcal{X}^n}{\operatorname{argmax}}\ P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \underset{\mathbf{x}\in\mathcal{C}}{\operatorname{argmax}}\ P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \triangleq \hat{\mathbf{x}}^{\text{block}}_{\text{ML}}(\mathbf{y}).$$

Binary linear codes

Binary Linear Codes (Part 1)

Let $\mathbf{H}$ be a parity-check matrix, e.g.

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

Binary Linear Codes (Part 1)

Let $\mathbf{H}$ be a parity-check matrix, e.g.

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

The code $\mathcal{C}$ described by $\mathbf{H}$ is then

$$\mathcal{C} = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \;\middle|\; \mathbf{H} \cdot \mathbf{x}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}} \pmod{2} \right\}.$$

Binary Linear Codes (Part 1)

Let $\mathbf{H}$ be a parity-check matrix, e.g.

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

The code $\mathcal{C}$ described by $\mathbf{H}$ is then

$$\mathcal{C} = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \;\middle|\; \mathbf{H} \cdot \mathbf{x}^\mathsf{T} = \mathbf{0}^\mathsf{T} \pmod 2 \right\}.$$

A vector $\mathbf{x} \in \mathbb{F}_2^5$ is a codeword if and only if

$$\mathbf{H} \cdot \mathbf{x}^\mathsf{T} = \mathbf{0}^\mathsf{T} \pmod 2.$$

Binary Linear Codes (Part 2)

This means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following two equations:

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

Binary Linear Codes (Part 2)

This means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following two equations:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad x_1 + x_2 + x_3 = 0 \pmod 2$$

Binary Linear Codes (Part 2)

This means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following two equations:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} x_1 + x_2 + x_3 = 0 \pmod 2 \\ x_2 + x_4 + x_5 = 0 \pmod 2 \end{array}$$

Binary Linear Codes (Part 2)

This means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following two equations:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} x_1 + x_2 + x_3 = 0 \pmod 2 \\ x_2 + x_4 + x_5 = 0 \pmod 2 \end{array}$$

In summary,

$$\begin{aligned} \mathcal{C} &= \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \;\middle|\; \mathbf{H} \cdot \mathbf{x}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}} \pmod 2 \right\} \\ &= \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \;\middle|\; \begin{array}{c} x_1 + x_2 + x_3 = 0 \pmod 2 \\ x_2 + x_4 + x_5 = 0 \pmod 2 \end{array} \right\}. \end{aligned}$$

Binary Linear Codes (Part 3)

Defining the codes $\mathcal{C}_1$ and $\mathcal{C}_2$ where

$$\mathcal{C}_1 = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \;\middle|\; x_1 + x_2 + x_3 = 0 \pmod 2 \right\},$$

Binary Linear Codes (Part 3)

Defining the codes $\mathcal{C}_1$ and $\mathcal{C}_2$ where

$$\mathcal{C}_1 = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \;\middle|\; x_1 + x_2 + x_3 = 0 \pmod 2 \right\},$$

$$\mathcal{C}_2 = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \;\middle|\; x_2 + x_4 + x_5 = 0 \pmod 2 \right\},$$

Binary Linear Codes (Part 3)

Defining the codes $\mathcal{C}_1$ and $\mathcal{C}_2$ where

$$\mathcal{C}_1 = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \;\middle|\; x_1 + x_2 + x_3 = 0 \pmod 2 \right\},$$

$$\mathcal{C}_2 = \left\{ (x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^5 \;\middle|\; x_2 + x_4 + x_5 = 0 \pmod 2 \right\},$$

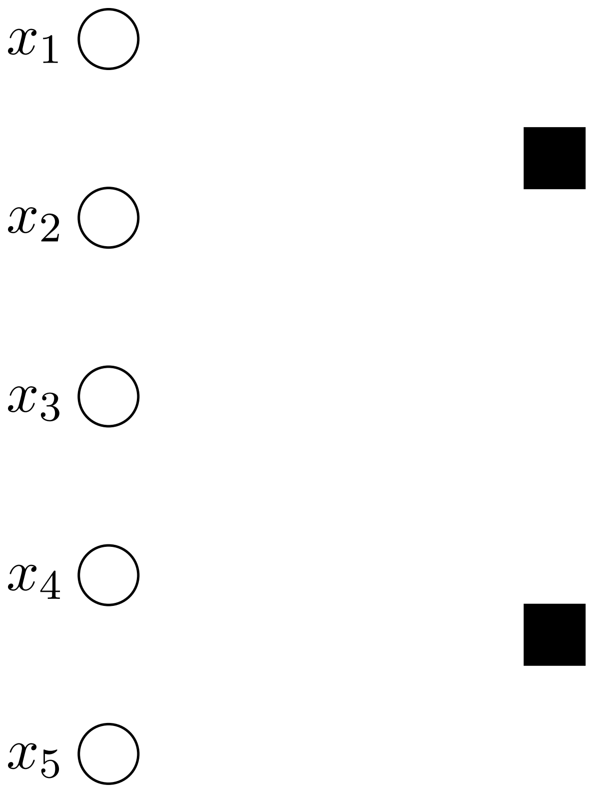
the code $\mathcal{C}$ can be written as the intersection of $\mathcal{C}_1$ and $\mathcal{C}_2$:

$$\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2.$$

Graphical representation of a code

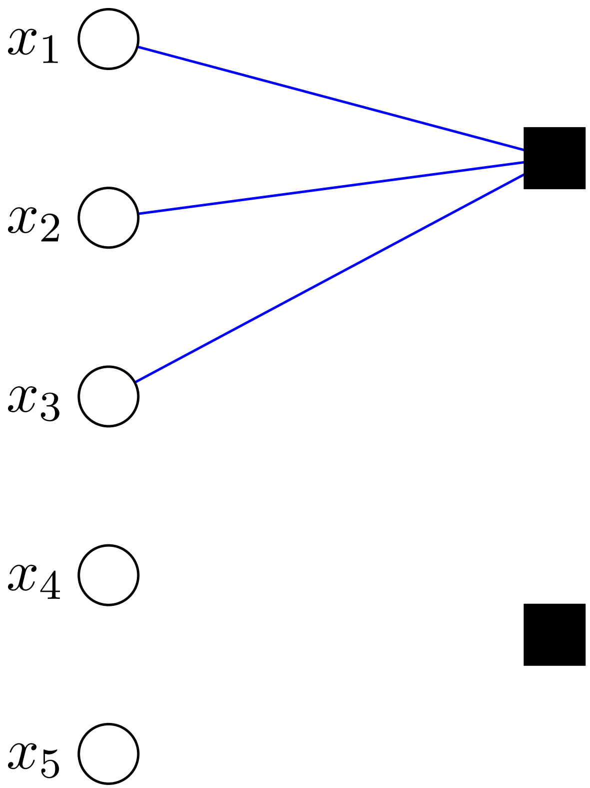
Graphical Representation of a Code

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



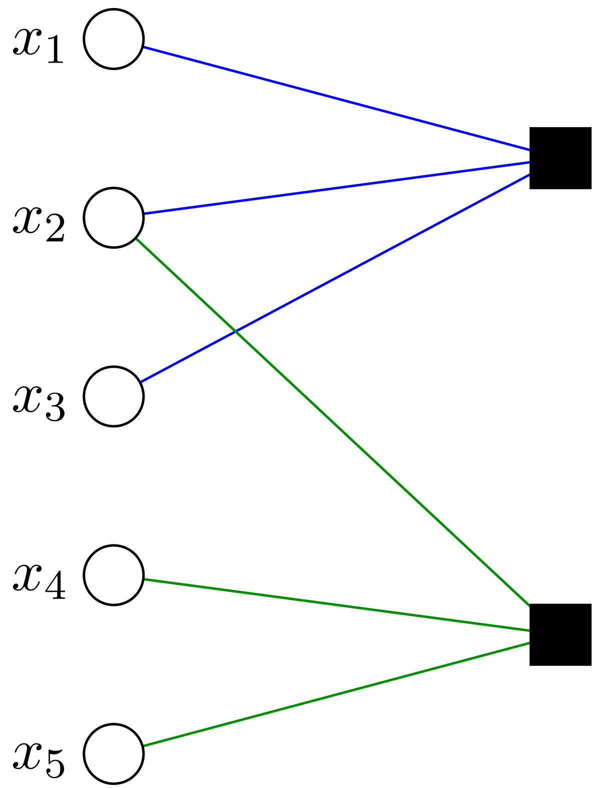
Graphical Representation of a Code

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



Graphical Representation of a Code

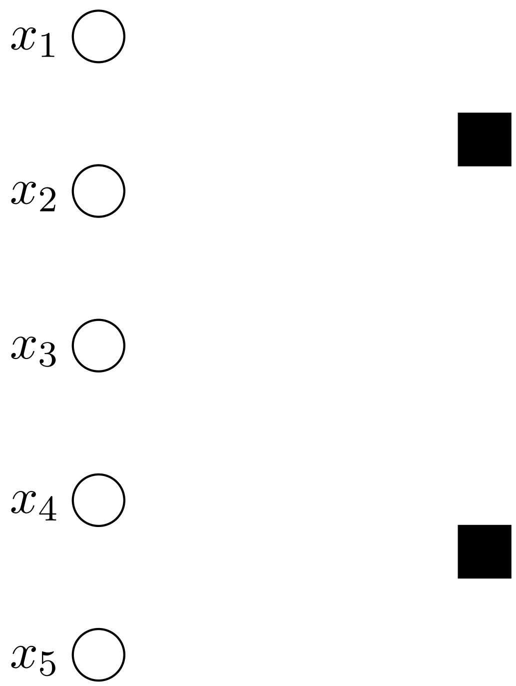
$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$



Graphical Representation of a Code

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

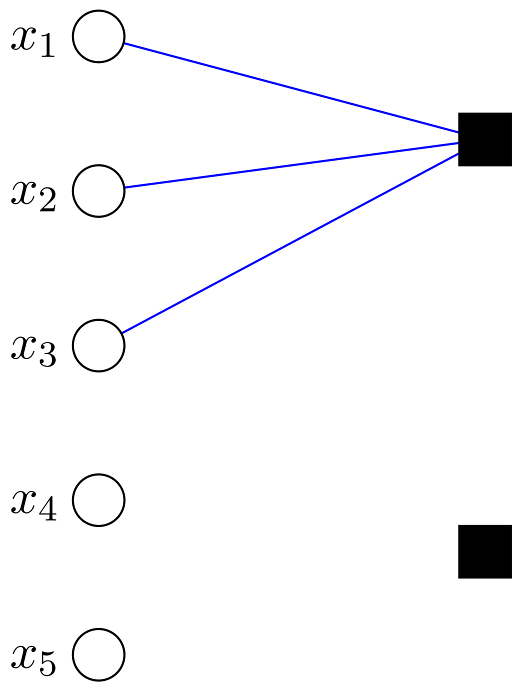
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1



Graphical Representation of a Code

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

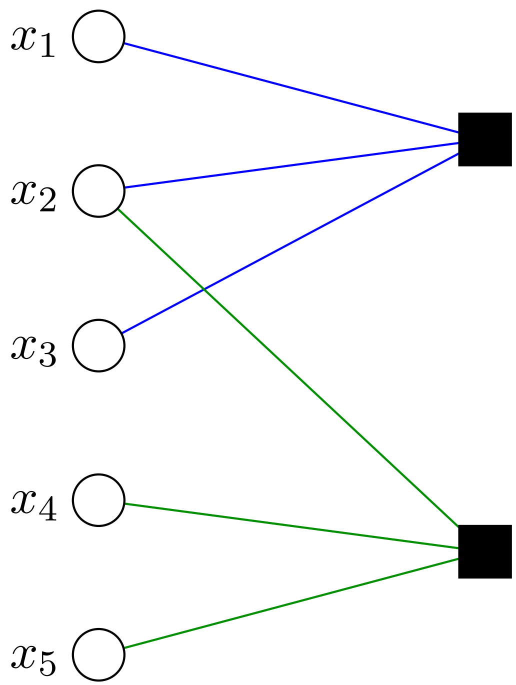
0000	0000	11111111
0000	1111	00001111
0000	1111	11110000
0011	0011	00110011
0101	0101	01010101



Graphical Representation of a Code

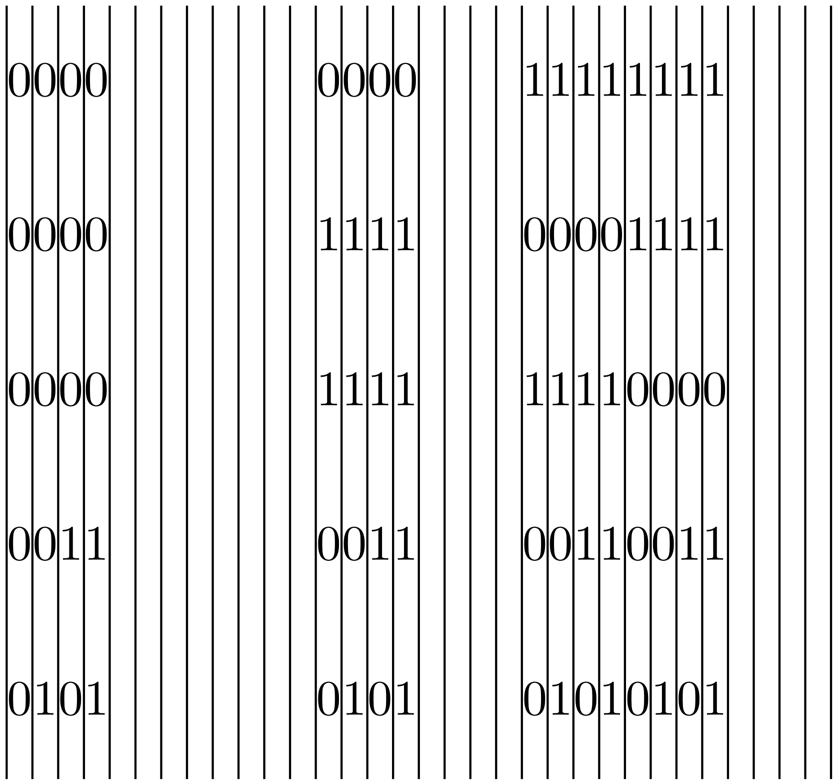
$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1
0	0	1	1	1	1	0	0
0	1	0	1	0	1	0	1
0	0	1	0	0	1	0	0

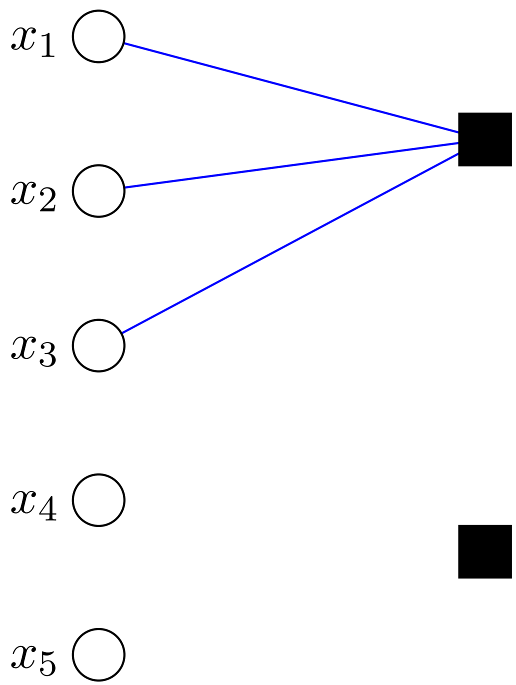


Graphical Representation of a Code

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



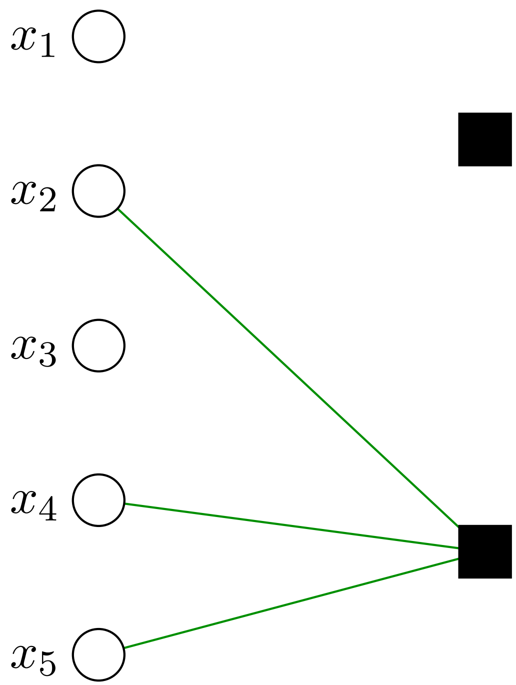
code $\mathcal{C}_1$



Graphical Representation of a Code

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

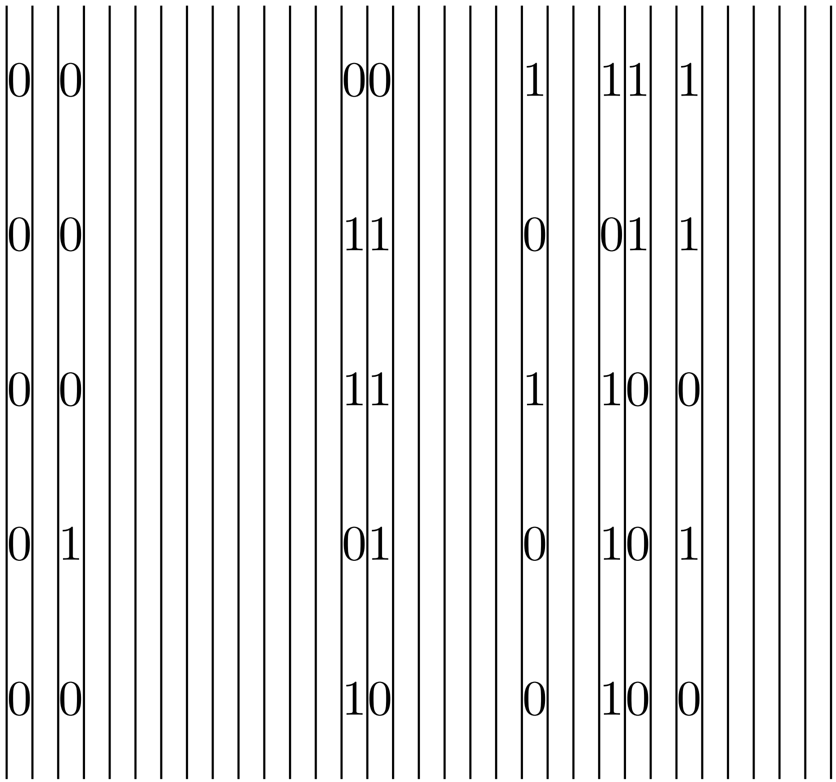
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0
0	1	0	1	1	0	1	0	1	0	1	1	1	0	1	1

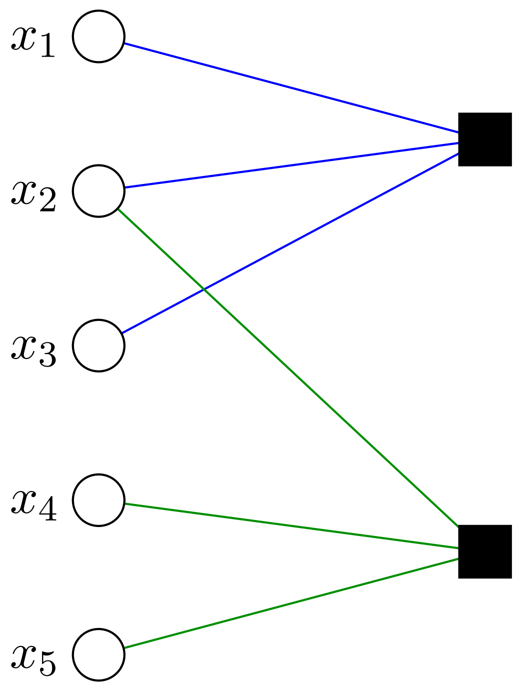


code $\mathcal{C}_2$

Graphical Representation of a Code

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

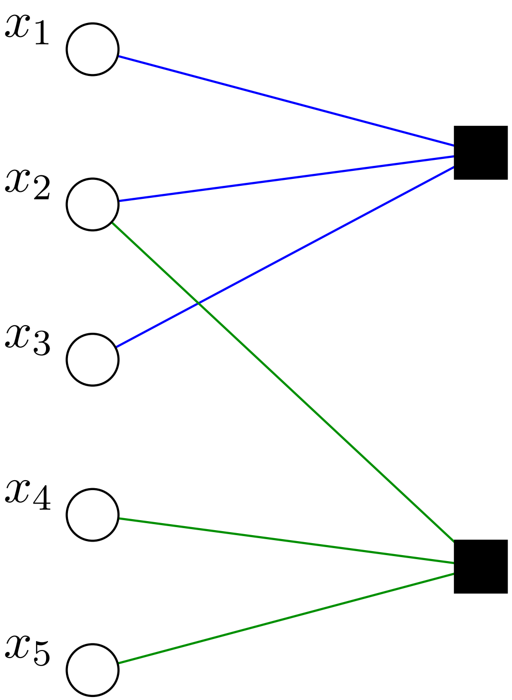




$$\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$$

FG of a Data Communication System based on a parity-check code

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

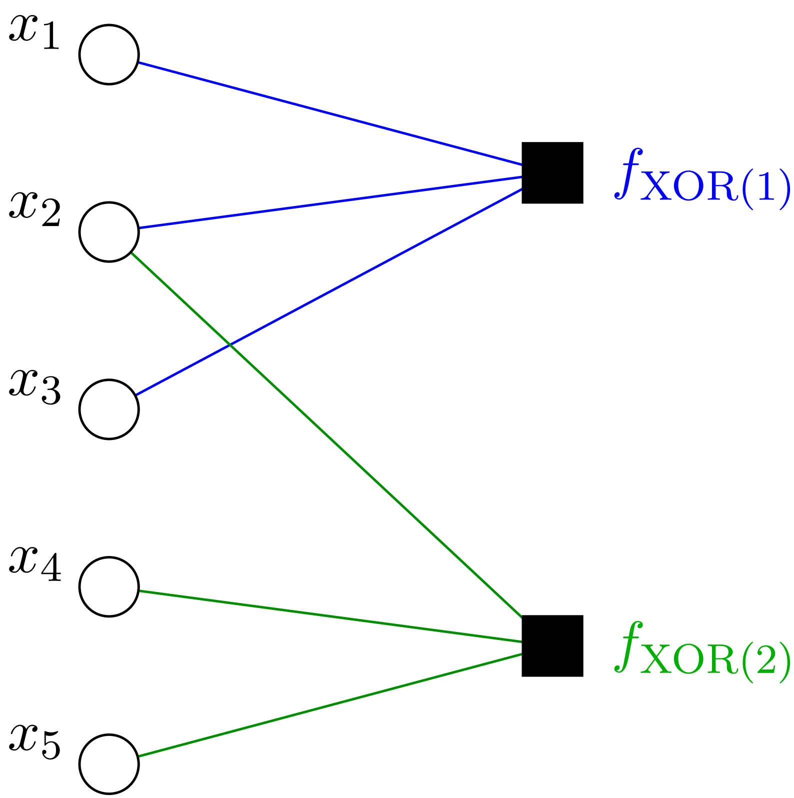


$$f_{\mathrm{XOR}}(x_1, x_2, x_3) \;=\; [x_1 + x_2 + x_3 = 0 \pmod 2]$$

$$f_{\mathrm{XOR}}(x_2, x_4, x_5) \;=\; [x_2 + x_4 + x_5 = 0 \pmod 2]$$

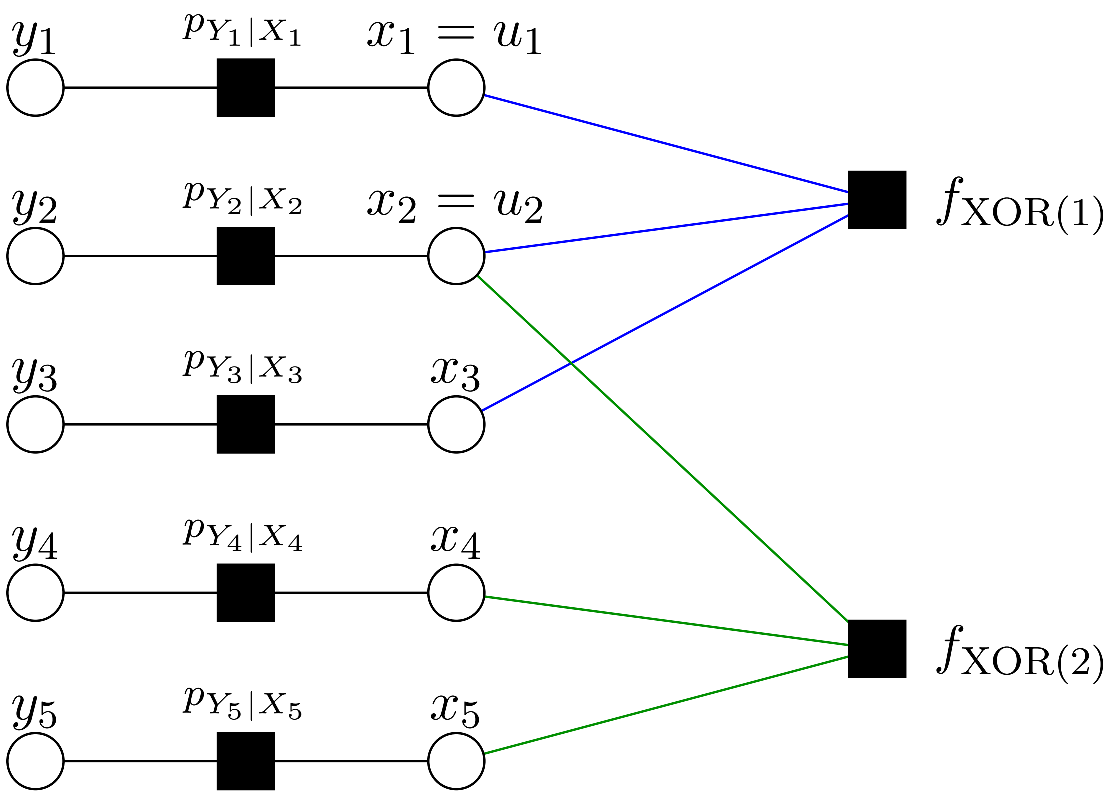
FG of a Data Communication System based on a parity-check code

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



FG of a Data Communication System based on a parity-check code

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



Expressing a decoder as the solution of a linear program

ML Decoding as an *Integer* LP

For memoryless channels, block-wise ML decoding of a binary code can be written as a linear program.

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg\max_{\mathbf{x}\in\mathcal{C}} \; P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$$

ML Decoding as an *Integer* LP

For memoryless channels, block-wise ML decoding of a binary code can be written as a linear program.

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg\max_{\mathbf{x}\in\mathcal{C}} \; P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \arg\min_{\mathbf{x}\in\mathcal{C}} \; \sum_{i=1}^{n} x_i\lambda_i,$$

ML Decoding as an *Integer* LP

For memoryless channels, block-wise ML decoding of a binary code can be written as a linear program.

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg\max_{\mathbf{x}\in\mathcal{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} x_i \lambda_i,$$

where

$$\lambda_i \triangleq \lambda_i(y_i) \triangleq \log \frac{P_{Y|X}(y_i|0)}{P_{Y|X}(y_i|1)}.$$

ML Decoding as an *Integer* LP

Derivation (we assume to have a memoryless channel):

$$\arg\max_{\mathbf{x}\in\mathcal{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$$

ML Decoding as an *Integer* LP

Derivation (we assume to have a memoryless channel):

$$\arg\max_{\mathbf{x}\in\mathcal{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$$

$$= \arg\max_{\mathbf{x}\in\mathcal{C}} \log \prod_{i=1}^{n} P_{Y_i|X_i}(y_i|x_i)$$

ML Decoding as an *Integer* LP

Derivation (we assume to have a memoryless channel):

$$
\begin{aligned}
&\arg\max_{\mathbf{x}\in\mathcal{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \\
&\qquad = \arg\max_{\mathbf{x}\in\mathcal{C}} \; \log \prod_{i=1}^{n} P_{Y_i|X_i}(y_i|x_i) \\
&\qquad = \arg\max_{\mathbf{x}\in\mathcal{C}} \; \sum_{i=1}^{n} \log P_{Y_i|X_i}(y_i|x_i)
\end{aligned}
$$

ML Decoding as an *Integer* LP

Derivation (we assume to have a memoryless channel):

$$\begin{aligned}
&\arg\max_{\mathbf{x}\in\mathcal{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \\
&\qquad = \arg\max_{\mathbf{x}\in\mathcal{C}} \log \prod_{i=1}^{n} P_{Y_i|X_i}(y_i|x_i) \\
&\qquad = \arg\max_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \log P_{Y_i|X_i}(y_i|x_i) \\
&\qquad = \arg\max_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \left(x_i \log \frac{P_{Y_i|X_i}(y_i|1)}{P_{Y_i|X_i}(y_i|0)} + \log P_{Y_i|X_i}(y_i|0) \right)
\end{aligned}$$

ML Decoding as an *Integer* LP

Derivation (we assume to have a memoryless channel):

$$
\begin{aligned}
&\arg\max_{\mathbf{x}\in\mathcal{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \\
&\quad = \arg\max_{\mathbf{x}\in\mathcal{C}} \log \prod_{i=1}^{n} P_{Y_i|X_i}(y_i|x_i) \\
&\quad = \arg\max_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \log P_{Y_i|X_i}(y_i|x_i) \\
&\quad = \arg\max_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \left(x_i \log \frac{P_{Y_i|X_i}(y_i|1)}{P_{Y_i|X_i}(y_i|0)} + \log P_{Y_i|X_i}(y_i|0) \right) \\
&\quad = \arg\max_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} x_i(-\lambda_i)
\end{aligned}
$$

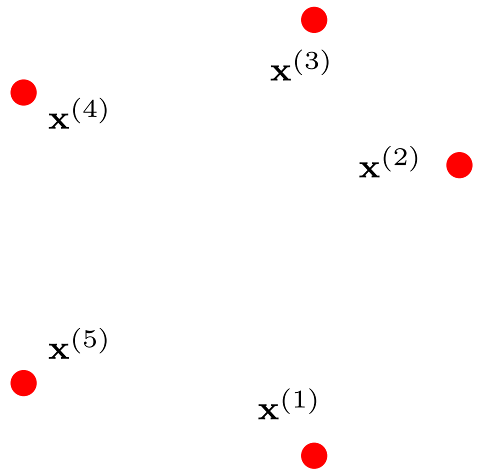
ML Decoding as an *Integer* LP

Derivation (we assume to have a memoryless channel):

$$\begin{aligned}
&\arg\max_{\mathbf{x}\in\mathcal{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \\
&\quad = \arg\max_{\mathbf{x}\in\mathcal{C}} \log \prod_{i=1}^{n} P_{Y_i|X_i}(y_i|x_i) \\
&\quad = \arg\max_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \log P_{Y_i|X_i}(y_i|x_i) \\
&\quad = \arg\max_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \left(x_i \log \frac{P_{Y_i|X_i}(y_i|1)}{P_{Y_i|X_i}(y_i|0)} + \log P_{Y_i|X_i}(y_i|0) \right) \\
&\quad = \arg\max_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} x_i(-\lambda_i) = \arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} x_i\lambda_i .
\end{aligned}$$

ML Decoding as an LP

$$\arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \lambda_i x_i$$



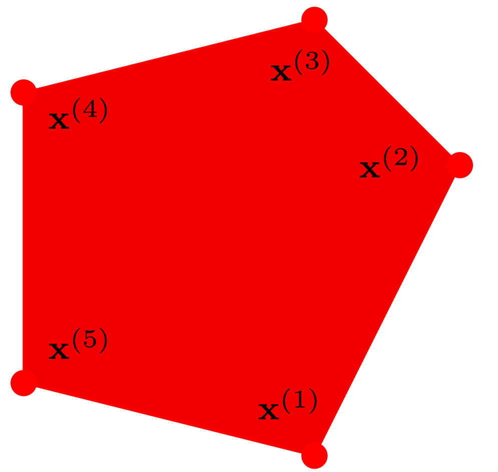
e.g.

$$\mathcal{C} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)}\right\}$$

ML Decoding as an LP

$$\arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \lambda_i x_i$$

$$\arg\min_{\mathbf{x}\in\mathrm{conv}(C)} \sum_{i=1}^{n} \lambda_i x_i$$



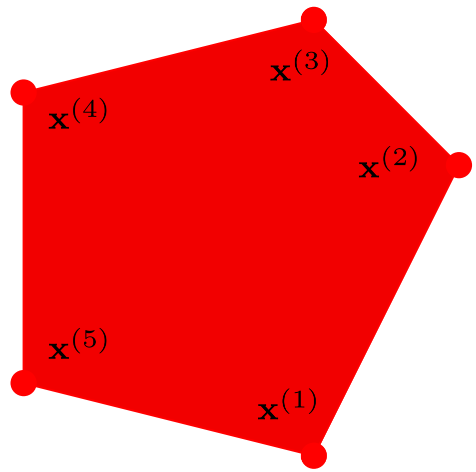
e.g.

$$\mathcal{C} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)}\right\}$$

ML Decoding as an LP

$$\arg\min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{n} \lambda_i x_i$$

$$\stackrel{*}{=} \arg\min_{\mathbf{x}\in\mathrm{conv}\,(C)} \sum_{i=1}^{n} \lambda_i x_i$$

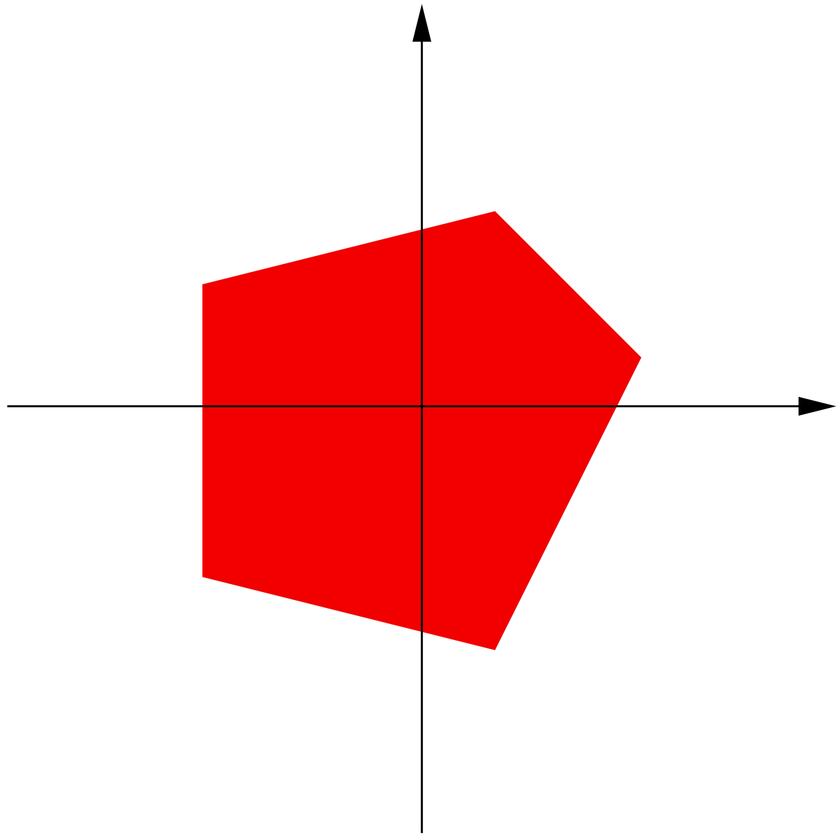


e.g.

$$\mathcal{C} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(5)}\right\}$$

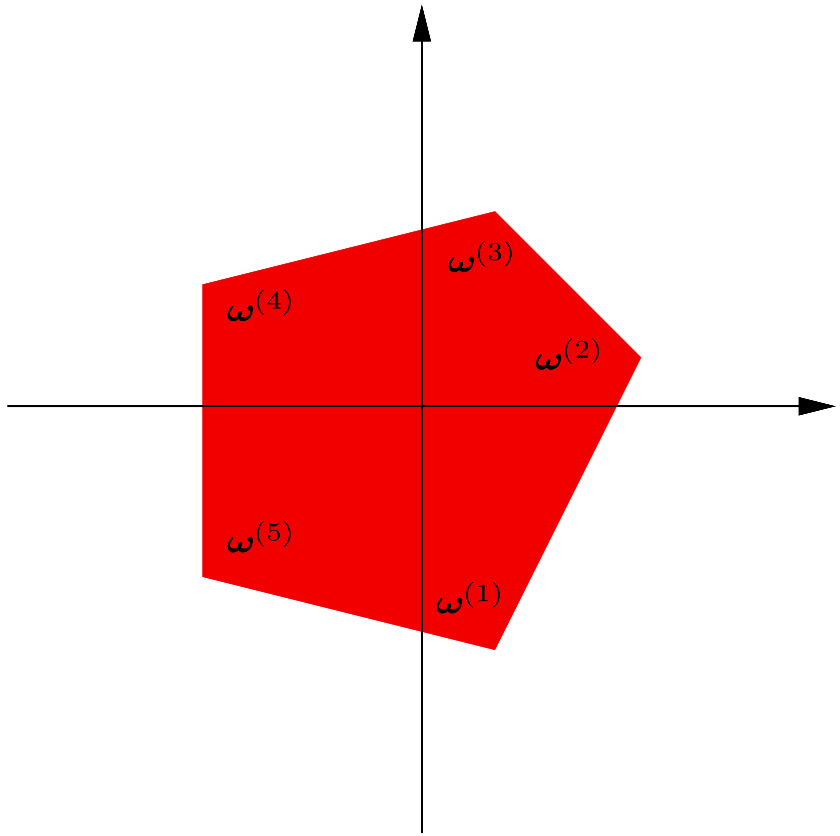
Linear Programs (LPs) (Part 1)

$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$



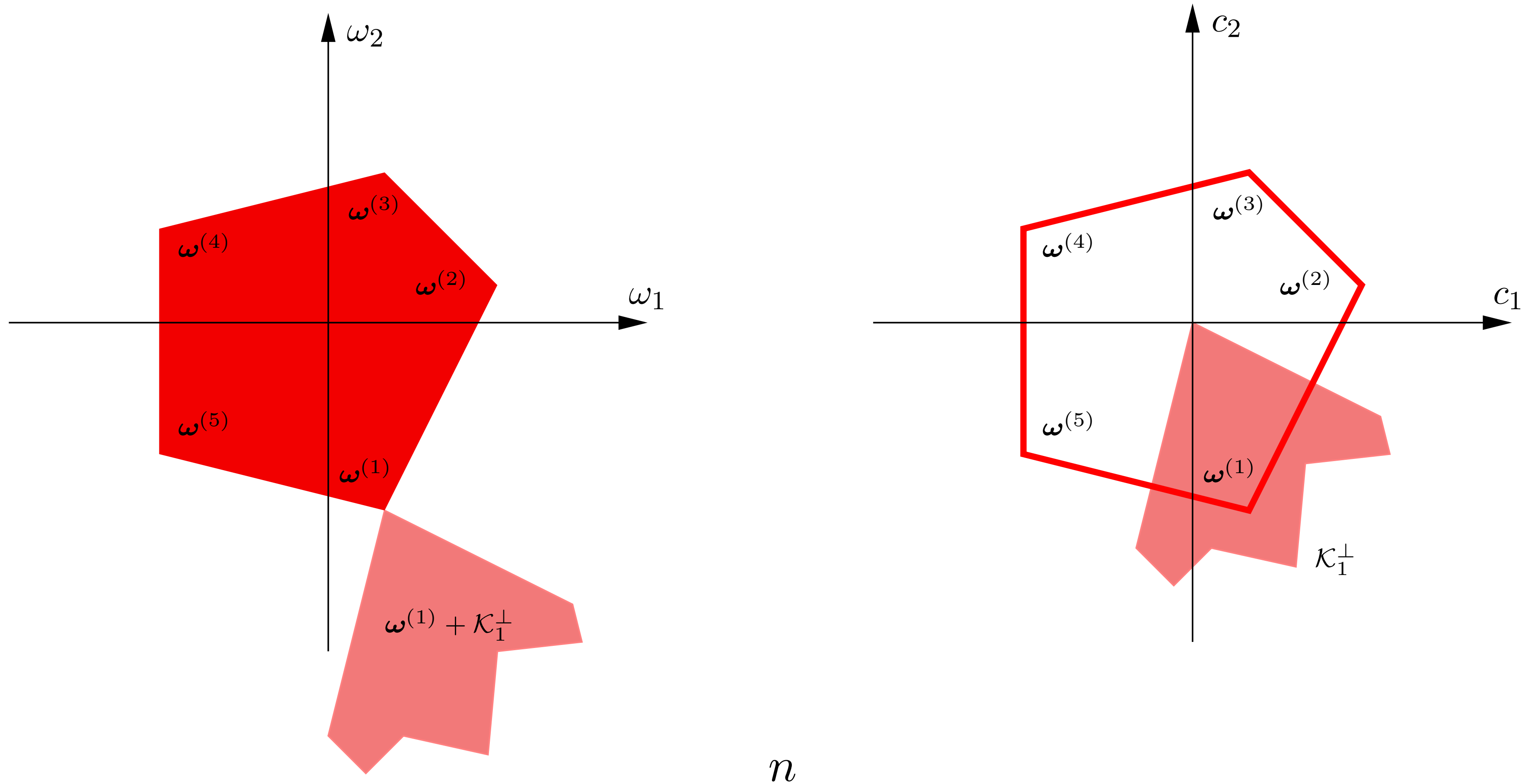
Linear Programs (LPs) (Part 1)

$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$



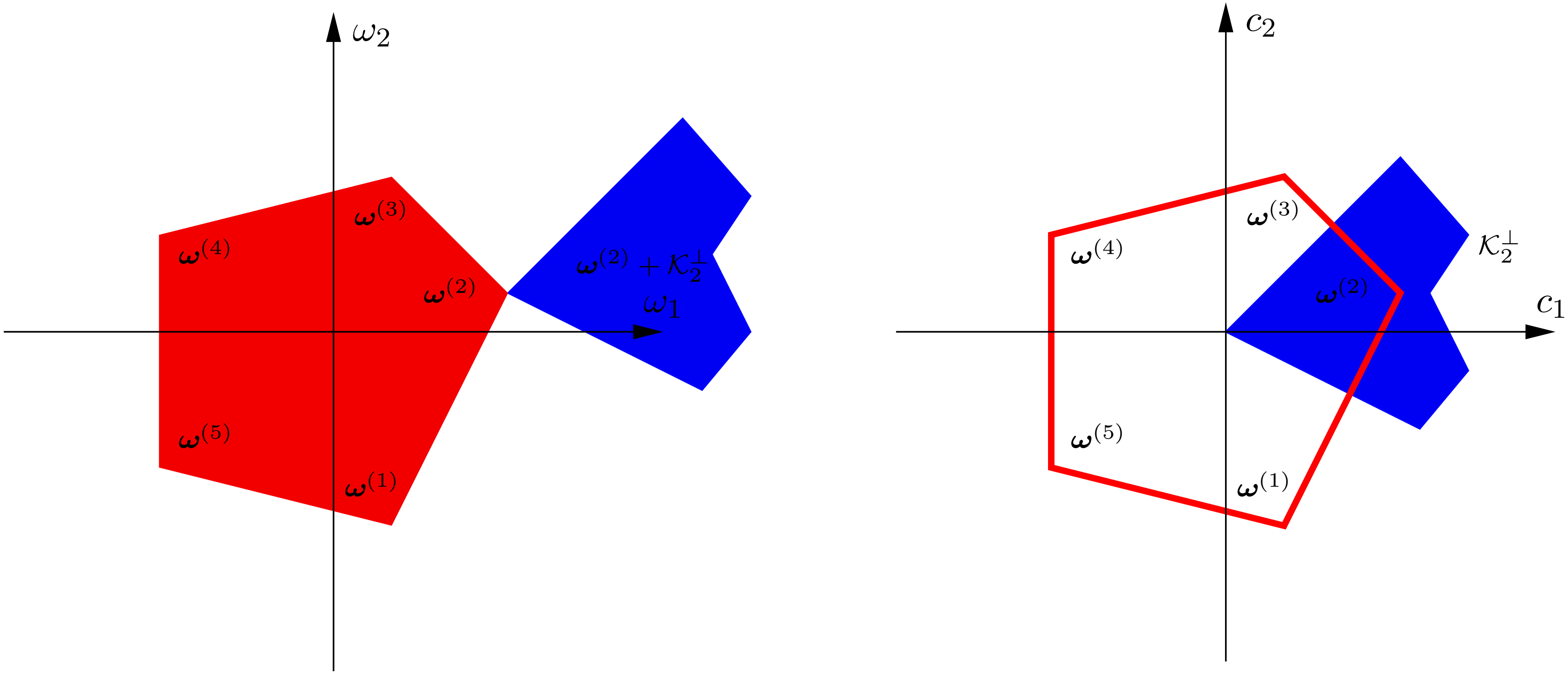
Because the cost function is linear and because $\mathcal{A}$ is a polytope, one of the vertices of $\mathcal{A}$ is always in the solution set.

Linear Programs (LPs) (Part 2)



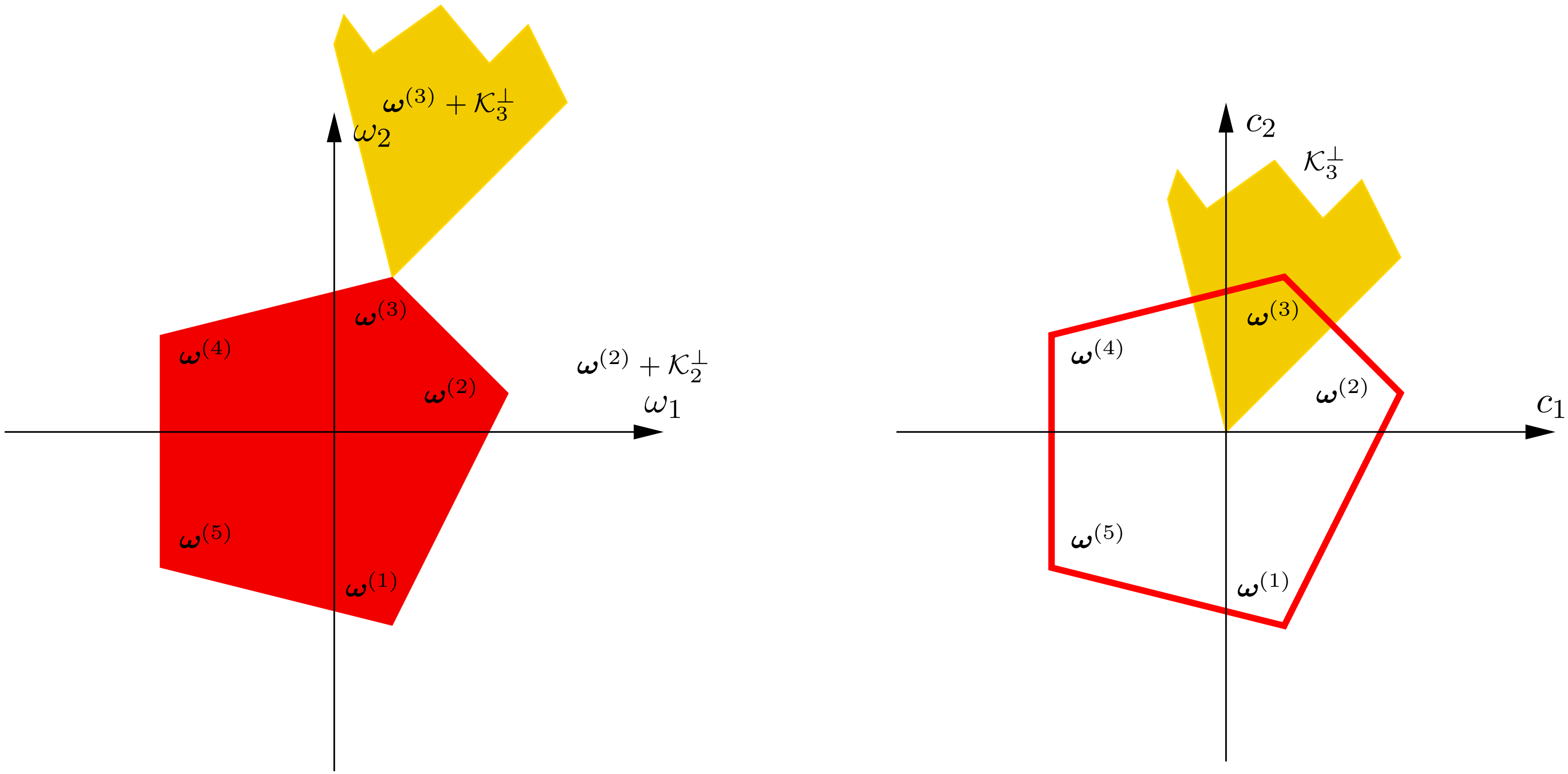
$$\arg\max_{\boldsymbol{\omega}\in\mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

Linear Programs (LPs) (Part 2)



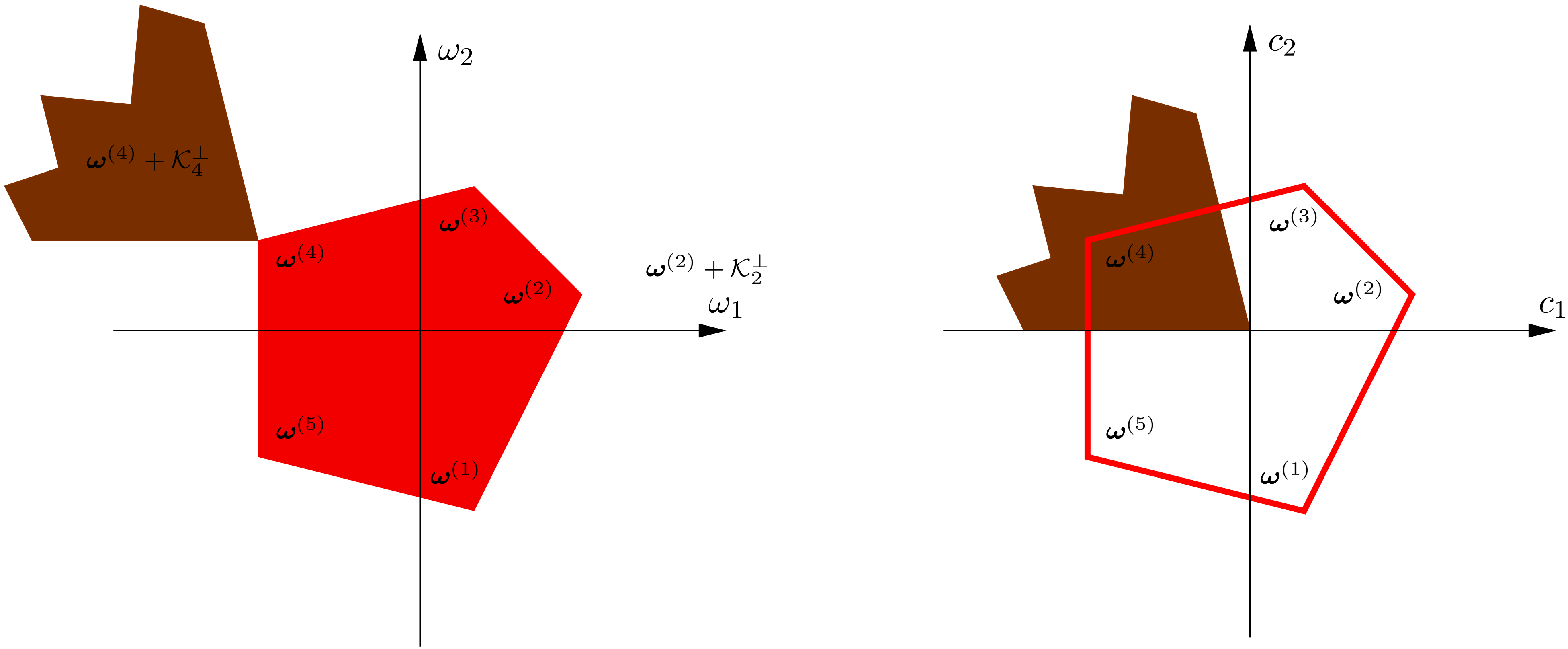
$$\arg\max_{\boldsymbol{\omega}\in\mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

Linear Programs (LPs) (Part 2)



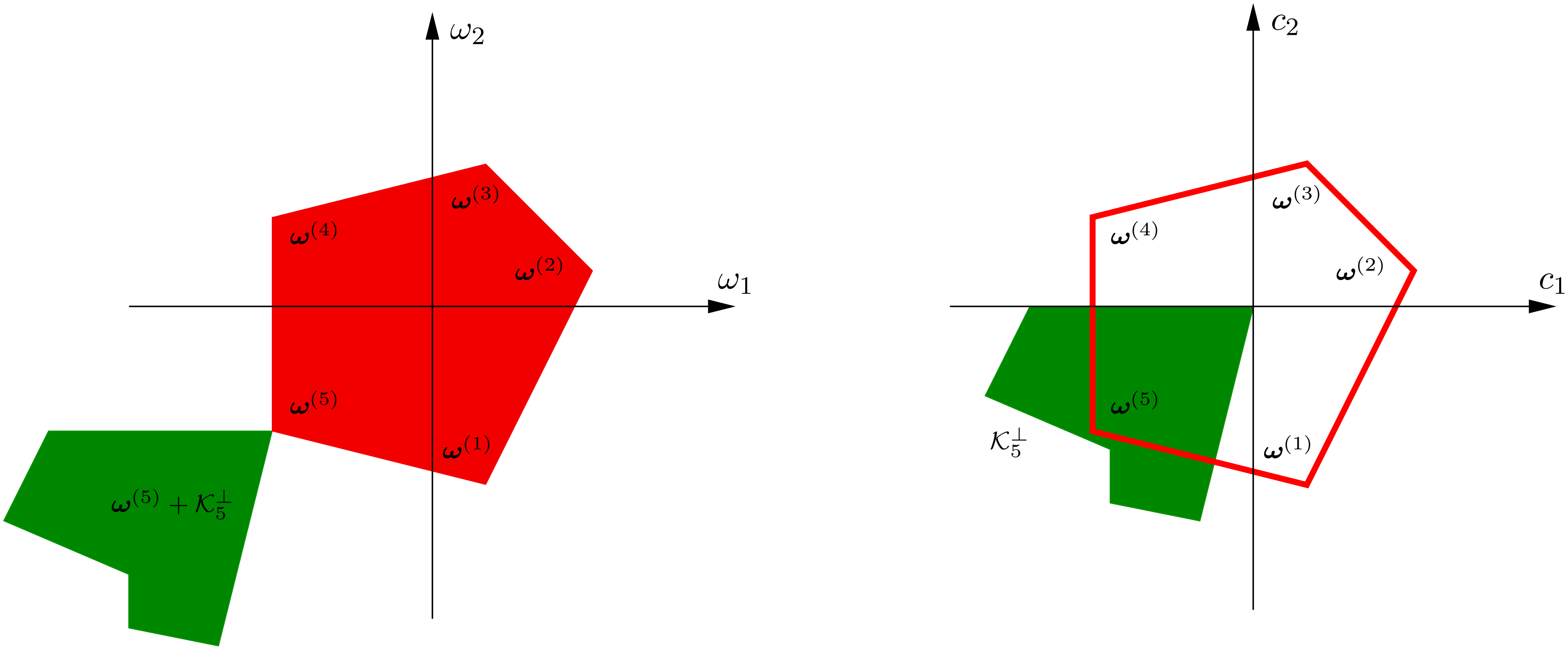
$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

Linear Programs (LPs) (Part 2)



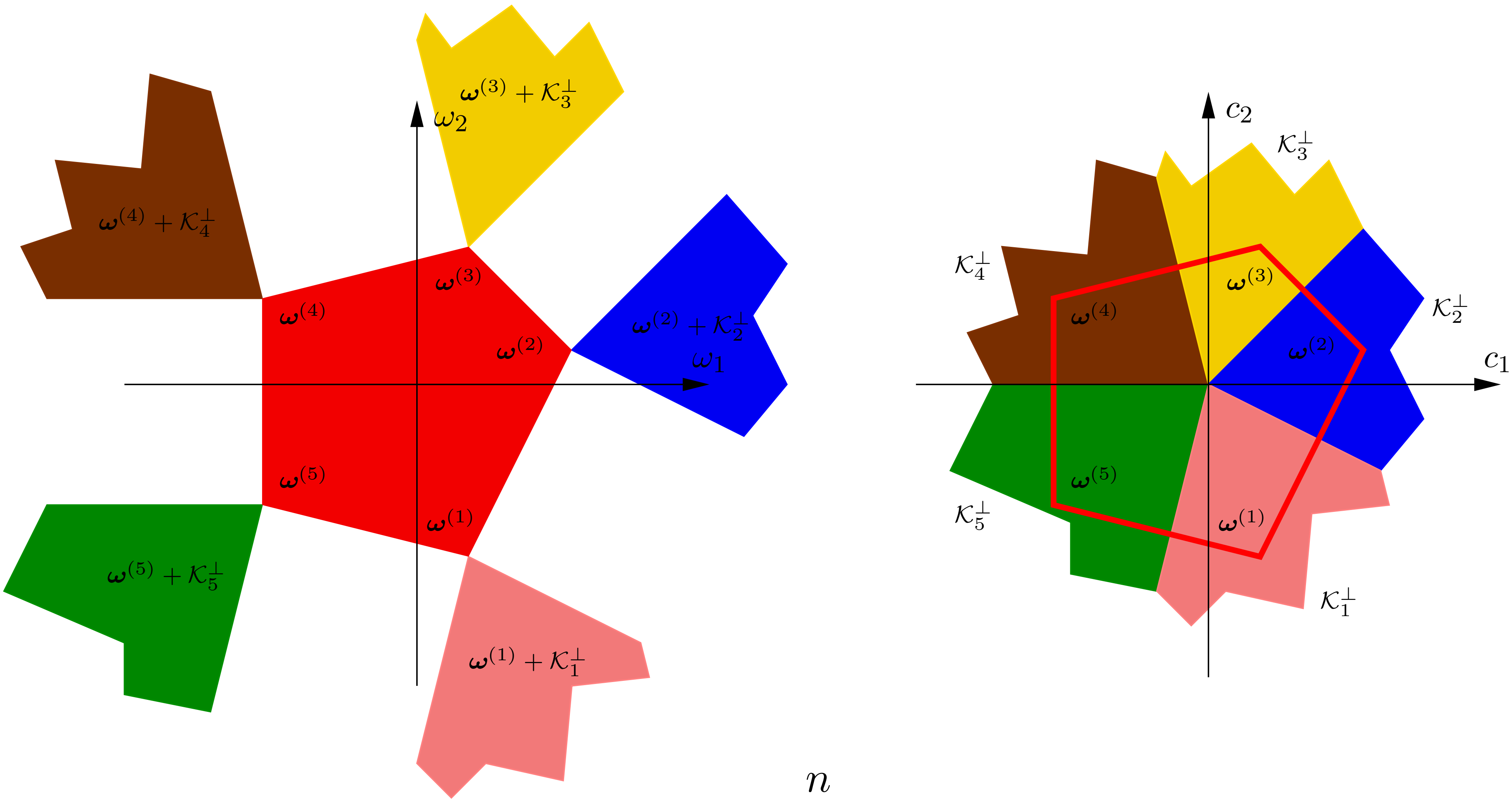
$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

Linear Programs (LPs) (Part 2)



$$\arg\max_{\boldsymbol{\omega}\in\mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

Linear Programs (LPs) (Part 2)



$$\arg\max_{\boldsymbol{\omega}\in\mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

LABS hp

ML Decoding as an LP

$$\hat{\mathbf{x}}^{\text{block}}_{\text{ML}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \text{conv}(\mathcal{C})} \sum_{i=1}^{n} x_i \lambda_i,$$

This is a linear program.

ML Decoding as an LP

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathrm{conv}(\mathcal{C})} \sum_{i=1}^{n} x_i \lambda_i,$$

This is a linear program.

However, the

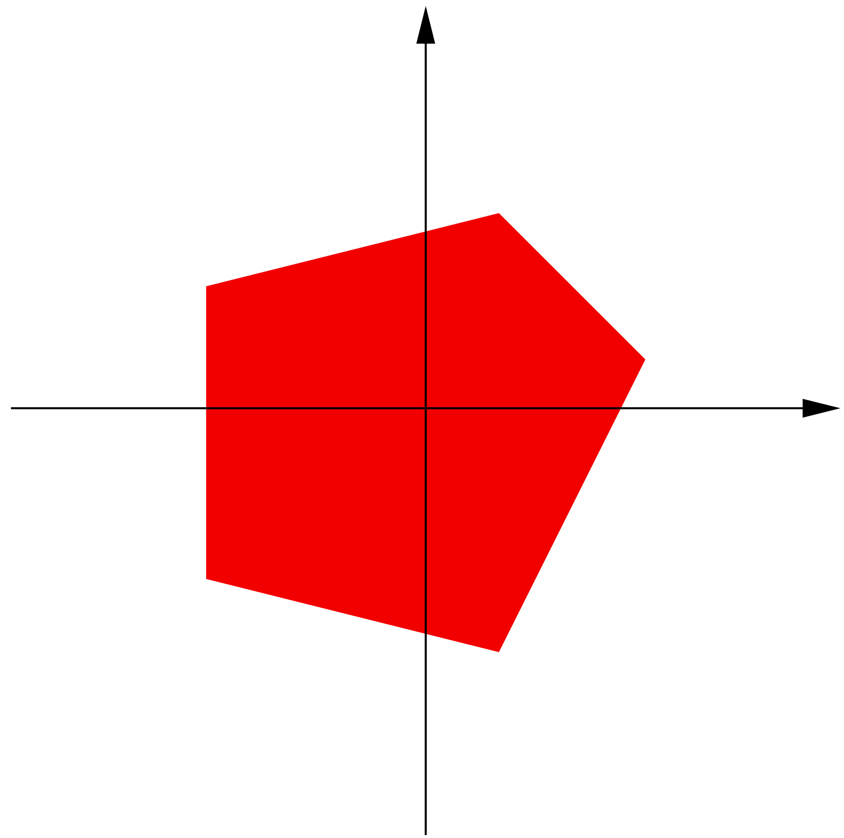
number of variables / equalities / inequalities

needed to describe the polytope $\mathrm{conv}(\mathcal{C})$ is (usually) exponential in n.

Relaxed linear programs and LP decoding

Relaxed Linear Programs (Part 1)

$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

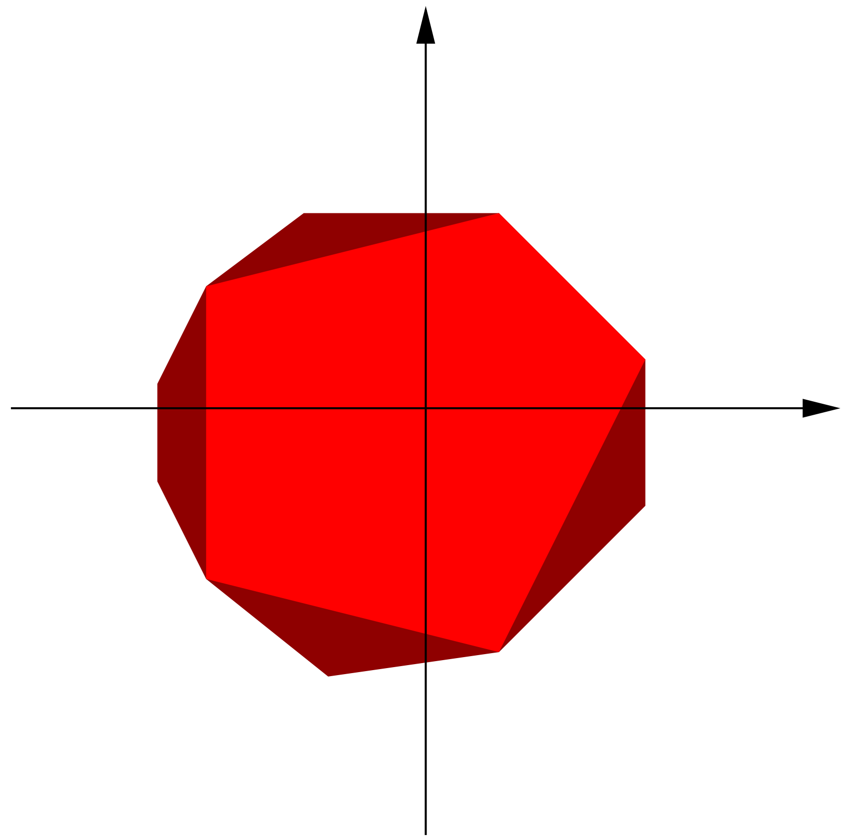


Relaxed Linear Programs (Part 1)

$$\operatorname*{arg\,max}_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

is replaced by

$$\operatorname*{arg\,max}_{\omega \in \mathcal{A}'} \sum_{i=1}^{n} c_i \omega_i$$

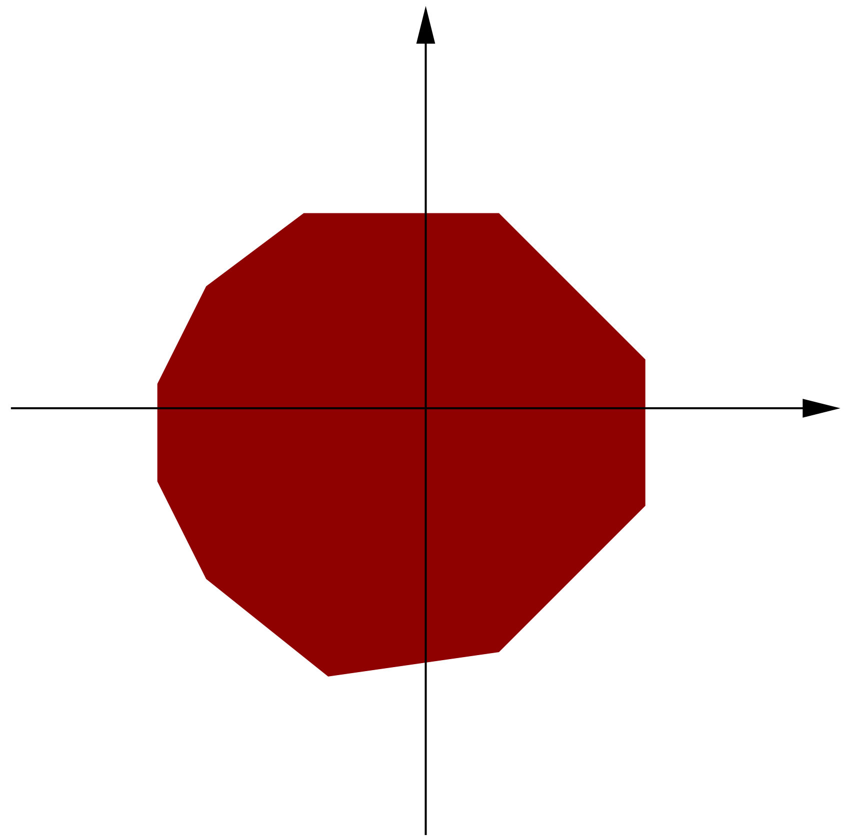


Relaxed Linear Programs (Part 1)

$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

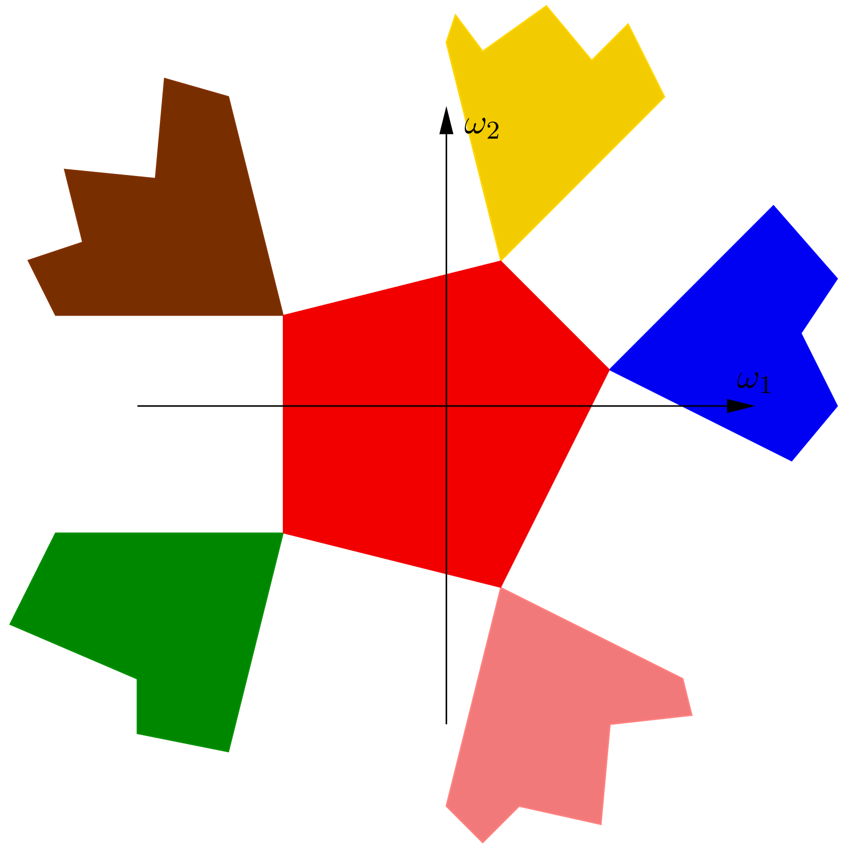
is replaced by

$$\arg\max_{\omega \in \mathcal{A}'} \sum_{i=1}^{n} c_i \omega_i$$



Relaxed Linear Programs (Part 2)

$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

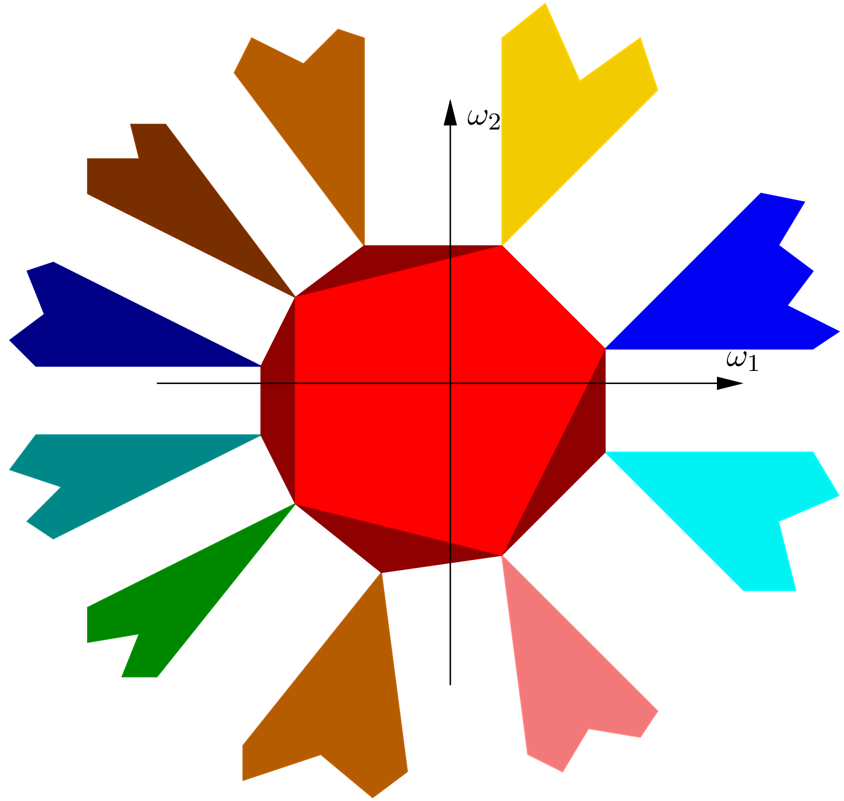


Relaxed Linear Programs (Part 2)

$$\arg\max_{\omega \in \mathcal{A}} \sum_{i=1}^{n} c_i \omega_i$$

is replaced by

$$\arg\max_{\omega \in \mathcal{A}'} \sum_{i=1}^{n} c_i \omega_i$$



LP Decoding (Part 1)

$$\hat{\boldsymbol{\omega}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C})} \sum_{i=1}^{n} \omega_i \lambda_i .$$

LP Decoding (Part 1)

$$\hat{\boldsymbol{\omega}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C})} \sum_{i=1}^{n} \omega_i \lambda_i .$$

A standard approach in optimization theory is then to relax the set $\mathrm{conv}(\mathcal{C})$ to a set $\mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ whose description complexity is much lower:

$$\hat{\boldsymbol{\omega}}_{\mathrm{LP}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \mathrm{relax}(\mathrm{conv}(\mathcal{C}))} \sum_{i=1}^{n} \omega_i \lambda_i .$$

Linear Programming Decoding (Part 4)

How do we obtain a suitable relaxation?

Linear Programming Decoding (Part 4)

How do we obtain a suitable relaxation? The following approach was proposed by Feldman / Karger / Wainwright and seems to work well for LDPC codes.

Linear Programming Decoding (Part 4)

How do we obtain a suitable relaxation? The following approach was proposed by Feldman / Karger / Wainwright and seems to work well for LDPC codes.

Before showing how this relaxation works, let us remember how we define a code using a parity-check matrix.

Let $\mathbf{H}$ be a parity-check matrix, e.g.

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

Linear Programming Decoding (Part 4)

How do we obtain a suitable relaxation? The following approach was proposed by Feldman / Karger / Wainwright and seems to work well for LDPC codes.

Before showing how this relaxation works, let us remember how we define a code using a parity-check matrix.

Let $\mathbf{H}$ be a parity-check matrix, e.g.

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

A vector $\mathbf{x} \in \mathbb{F}_2^5$ is a codeword if and only if

$$\mathbf{H}\mathbf{x}^\mathsf{T} = \mathbf{0}^\mathsf{T}.$$

Linear Programming Decoding (Part 5)

In our case this means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following three equations:

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

Linear Programming Decoding (Part 5)

In our case this means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following three equations:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad x_1 + x_2 + x_3 = 0 \pmod 2$$

Linear Programming Decoding (Part 5)

In our case this means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following three equations:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} x_1 + x_2 + x_3 = 0 \pmod 2 \\ x_2 + x_4 + x_5 = 0 \pmod 2 \end{array}$$

Linear Programming Decoding (Part 5)

In our case this means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following three equations:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} x_1 + x_2 + x_3 = 0 \pmod 2 \\ x_2 + x_4 + x_5 = 0 \pmod 2 \\ x_3 + x_4 + x_5 = 0 \pmod 2 \end{array}$$

Linear Programming Decoding (Part 5)

In our case this means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following three equations:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{aligned} x_1 + x_2 + x_3 &= 0 \pmod 2 \\ x_2 + x_4 + x_5 &= 0 \pmod 2 \\ x_3 + x_4 + x_5 &= 0 \pmod 2 \end{aligned}$$

Therefore, $\mathcal{C}$ can be seen as the intersection of three codes

$$\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2 \cap \mathcal{C}_3,$$

Linear Programming Decoding (Part 5)

In our case this means that $\mathbf{x}$ is a codeword if and only if $\mathbf{x}$ fulfills the following three equations:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{aligned} x_1 + x_2 + x_3 &= 0 \pmod 2 \\ x_2 + x_4 + x_5 &= 0 \pmod 2 \\ x_3 + x_4 + x_5 &= 0 \pmod 2 \end{aligned}$$

Therefore, $\mathcal{C}$ can be seen as the intersection of three codes

$$\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2 \cap \mathcal{C}_3,$$

where

$$\begin{aligned} \mathcal{C}_1 &\triangleq \left\{ \mathbf{x} \in \mathbb{F}_2^5 \mid \mathbf{h}_1 \mathbf{x}^\mathsf{T} = 0 \pmod 2 \right\}, \\ \mathcal{C}_2 &\triangleq \left\{ \mathbf{x} \in \mathbb{F}_2^5 \mid \mathbf{h}_2 \mathbf{x}^\mathsf{T} = 0 \pmod 2 \right\}, \\ \mathcal{C}_3 &\triangleq \left\{ \mathbf{x} \in \mathbb{F}_2^5 \mid \mathbf{h}_3 \mathbf{x}^\mathsf{T} = 0 \pmod 2 \right\}. \end{aligned}$$

Linear Programming Decoding (Part 6)

Let the relaxation $\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

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

Linear Programming Decoding (Part 6)

Let the relaxation $\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

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

Linear Programming Decoding (Part 6)

Let the relaxation $\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1)$$

Linear Programming Decoding (Part 6)

Let the relaxation $\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \end{array}$$

Linear Programming Decoding (Part 6)

Let the relaxation $\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_3) \end{array}$$

Linear Programming Decoding (Part 6)

Let the relaxation $\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_3) \end{array}$$

Therefore,

$$\mathrm{relax}(\mathrm{conv}(\mathcal{C}))$$

This relaxation turns out to have many desirable properties. Note that the points in $\mathcal{P}(\mathbf{H})$ are called pseudo-codewords.

Linear Programming Decoding (Part 6)

Let the relaxation $\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{aligned} \boldsymbol{\omega} &\in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} &\in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} &\in \mathrm{conv}(\mathcal{C}_3) \end{aligned}$$

Therefore,

$$\mathrm{relax}(\mathrm{conv}(\mathcal{C})) \triangleq \underbrace{\mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_3)}.$$

This relaxation turns out to have many desirable properties. Note that the points in $\mathcal{P}(\mathbf{H})$ are called pseudo-codewords.

Linear Programming Decoding (Part 6)

Let the relaxation $\text{relax}(\mathcal{C}) \triangleq \text{relax}(\text{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \text{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \text{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \text{conv}(\mathcal{C}_3) \end{array}$$

Therefore,

$$\text{relax}(\text{conv}(\mathcal{C})) \triangleq \underbrace{\text{conv}(\mathcal{C}_1) \cap \text{conv}(\mathcal{C}_2) \cap \text{conv}(\mathcal{C}_3)}_{\text{Fundamental polytope } \mathcal{P}(\mathbf{H})}.$$

This relaxation turns out to have many desirable properties. Note that the points in $\mathcal{P}(\mathbf{H})$ are called pseudo-codewords.

Linear Programming Decoding (Part 6)

Let the relaxation $\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_3) \end{array}$$

Therefore,

$$\mathrm{conv}(\mathcal{C}) \subseteq \mathrm{relax}(\mathrm{conv}(\mathcal{C})) \triangleq \underbrace{\mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_3)}_{\text{Fundamental polytope } \mathcal{P}(\mathbf{H})}.$$

This relaxation turns out to have many desirable properties. Note that the points in $\mathcal{P}(\mathbf{H})$ are called pseudo-codewords.

Linear Programming Decoding (Part 6)

Let the relaxation $\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{relax}(\mathrm{conv}(\mathcal{C}))$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_3) \end{array}$$

Therefore,

$$\mathcal{C} \subset \mathrm{conv}(\mathcal{C}) \subseteq \mathrm{relax}(\mathrm{conv}(\mathcal{C})) \triangleq \underbrace{\mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_3)}_{\text{Fundamental polytope } \mathcal{P}(\mathbf{H})}.$$

This relaxation turns out to have many desirable properties. Note that the points in $\mathcal{P}(\mathbf{H})$ are called pseudo-codewords.

Block-wise ML Decoding vs. LP Decoding

Block-wise ML decoding:

LP decoding:

Block-wise ML Decoding vs. LP Decoding

Block-wise ML decoding:

$$\hat{\mathbf{x}}^{\text{block}}_{\text{ML}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \text{conv}(\mathcal{C})} \sum_{i=1}^{n} x_i \lambda_i .$$

LP decoding:

Block-wise ML Decoding vs. LP Decoding

Block-wise ML decoding:

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathrm{conv}(\mathcal{C})} \sum_{i=1}^{n} x_i \lambda_i .$$

LP decoding:

$$\hat{\boldsymbol{\omega}}_{\mathrm{LP}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \mathcal{P}(\mathbf{H})} \sum_{i=1}^{n} \omega_i \lambda_i .$$

Block-wise ML Decoding vs. LP Decoding

Block-wise ML decoding:

$$\hat{\mathbf{x}}_{\mathrm{ML}}^{\mathrm{block}}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathrm{conv}\left(\cap_{j=1}^{m} \mathcal{C}_j\right)} \sum_{i=1}^{n} x_i \lambda_i .$$

LP decoding:

$$\hat{\boldsymbol{\omega}}_{\mathrm{LP}}(\mathbf{y}) = \arg \min_{\boldsymbol{\omega} \in \cap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)} \sum_{i=1}^{n} \omega_i \lambda_i .$$

Fundamental Polytope

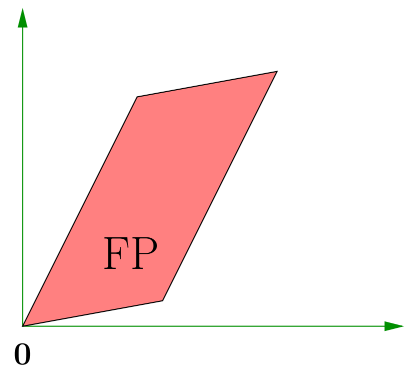
$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{array}{l} \Rightarrow \mathcal{C}_1 \\ \Rightarrow \mathcal{C}_2 \\ \Rightarrow \mathcal{C}_3 \end{array}$$

$$\Rightarrow \mathcal{C} = \bigcap_{j=1}^{m} \mathcal{C}_j$$

Fundamental Polytope

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{matrix} \Rightarrow \mathcal{C}_1 \\ \Rightarrow \mathcal{C}_2 \\ \Rightarrow \mathcal{C}_3 \end{matrix} \qquad \begin{matrix} \Rightarrow \mathrm{conv}(\mathcal{C}_1) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_2) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_3) \end{matrix}$$

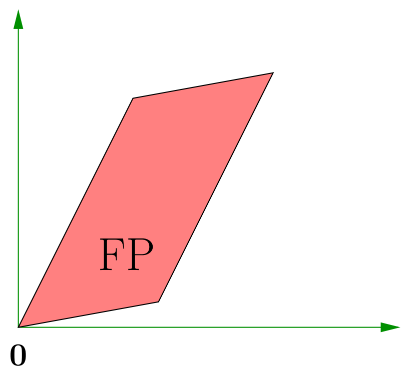
$$\Rightarrow \mathcal{C} = \bigcap_{j=1}^{m} \mathcal{C}_j \qquad \Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)}_{\text{Fundamental polytope}}$$



Fundamental Polytope / Cone (Part 1)

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{array}{l} \Rightarrow \operatorname{conv}(\mathcal{C}_1) \\ \Rightarrow \operatorname{conv}(\mathcal{C}_2) \\ \Rightarrow \operatorname{conv}(\mathcal{C}_3) \end{array}$$

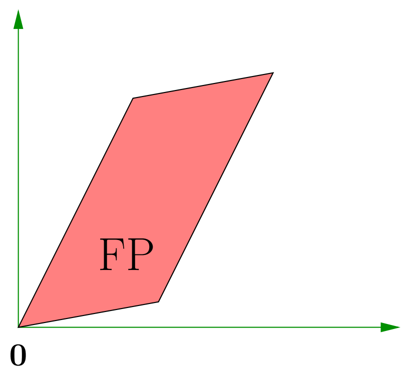
$$\Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \operatorname{conv}(\mathcal{C}_j)}_{\text{Fundamental polytope}}$$

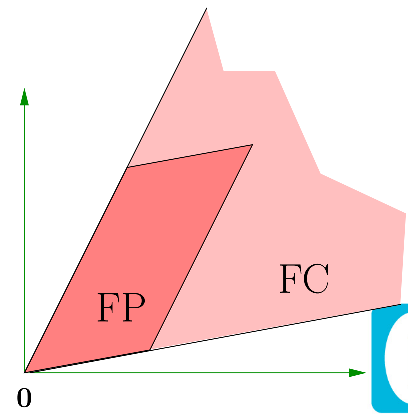


Fundamental Polytope / Cone (Part 1)

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{array}{l} \Rightarrow \mathrm{conv}(\mathcal{C}_1) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_2) \\ \Rightarrow \mathrm{conv}(\mathcal{C}_3) \end{array}$$

$$\Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)}_{\text{Fundamental polytope}}$$

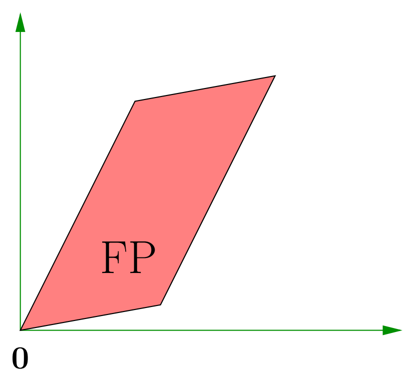


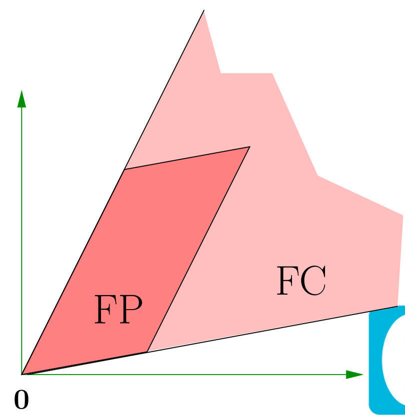


Fundamental Polytope / Cone (Part 1)

$$
\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}
\begin{array}{ll}
\Rightarrow \mathrm{conv}(\mathcal{C}_1) & \Rightarrow \mathrm{conic}(\mathcal{C}_1) \\
\Rightarrow \mathrm{conv}(\mathcal{C}_2) & \Rightarrow \mathrm{conic}(\mathcal{C}_2) \\
\Rightarrow \mathrm{conv}(\mathcal{C}_3) & \Rightarrow \mathrm{conic}(\mathcal{C}_3)
\end{array}
$$

$$
\Rightarrow \underbrace{\mathcal{P}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j)}_{\text{Fundamental polytope}} \quad \Rightarrow \underbrace{\mathcal{K}(\mathbf{H}) = \bigcap_{j=1}^{m} \mathrm{conic}(\mathcal{C}_j)}_{\text{Fundamental cone}}
$$





Convex hull of simple codes

Convex Hull of Simple Codes (Part 1)

Let $\mathcal{C}$ be defined by the parity-check matrix

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

Then

$$\mathcal{C} = \{(0,0),\ (1,1)\}$$

and

$$\operatorname{conv}(\mathcal{C}) = \left\{ \boldsymbol{\omega} \in [0,1]^2 \;\middle|\; \begin{array}{r} -\omega_1 + \omega_2 \geq 0 \\ +\omega_1 - \omega_2 \geq 0 \end{array} \right\},$$

where $[0,1] = \{r \in \mathbb{R} \mid 0 \leq r \leq 1\}$.

Convex Hull of Simple Codes (Part 2)

Let $\mathcal{C}$ be defined by the parity-check matrix

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

Then

$$\mathcal{C} = \{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1)\}$$

and

$$\operatorname{conv}(\mathcal{C}) = \left\{ \boldsymbol{\omega} \in [0,1]^3 \;\middle|\; \begin{array}{l} -\omega_1 + \omega_2 + \omega_3 \geq 0 \\ +\omega_1 - \omega_2 + \omega_3 \geq 0 \\ +\omega_1 + \omega_2 - \omega_3 \geq 0 \\ -\omega_1 - \omega_2 - \omega_3 \geq -2 \end{array} \right\}.$$

Conic Hull of Simple Codes (Part 1)

Let $\mathcal{C}$ be defined by the parity-check matrix

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

Then

$$\mathcal{C} = \big\{(0,0),\ (1,1)\big\}$$

and

$$\operatorname{conic}(\mathcal{C}) = \left\{ \boldsymbol{\omega} \in \mathbb{R}_+^2 \;\middle|\; \begin{array}{l} -\omega_1 + \omega_2 \geq 0 \\ +\omega_1 - \omega_2 \geq 0 \end{array} \right\},$$

where $\mathbb{R}_+ = \{r \in \mathbb{R} \mid r \geq 0\}$.

Conic Hull of Simple Codes (Part 2)

Let $\mathcal{C}$ be defined by the parity-check matrix

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

Then

$$\mathcal{C} = \{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1)\}$$

and

$$\operatorname{conic}(\mathcal{C}) = \left\{ \boldsymbol{\omega} \in \mathbb{R}^3_+ \;\middle|\; \begin{array}{c} -\omega_1 + \omega_2 + \omega_3 \geq 0 \\ +\omega_1 - \omega_2 + \omega_3 \geq 0 \\ +\omega_1 + \omega_2 - \omega_3 \geq 0 \end{array} \right\}.$$

A Simple Code (Part 1)

Let us consider the length-3 code $\mathcal{C}$ defined by the parity-check matrix

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

The code $\mathcal{C}$ can be written as $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2 \cap \mathcal{C}_3$ with

$$\mathcal{C}_1 = \{(0,0,0),\ (1,1,0),\ (0,0,1),\ (1,1,1)\}$$
$$\mathcal{C}_2 = \{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1)\}$$
$$\mathcal{C}_3 = \{(0,0,0),\ (0,1,1),\ (1,0,0),\ (1,1,1)\}$$

A Simple Code (Part 2)

The fundamental polytope is $\mathcal{P}(\mathbf{H}) = \operatorname{conv}(\mathcal{C}_1) \cap \operatorname{conv}(\mathcal{C}_2) \cap \operatorname{conv}(\mathcal{C}_3)$ with

$$\operatorname{conv}(\mathcal{C}_1) = \operatorname{conv}\left(\{(0,0,0),\ (1,1,0),\ (0,0,1),\ (1,1,1)\}\right)$$

$$= \left\{ \boldsymbol{\omega} \in [0,1]^3 \;\middle|\; \begin{array}{l} -\omega_1 + \omega_2 \geq 0 \\ +\omega_1 - \omega_2 \geq 0 \end{array} \right\}$$

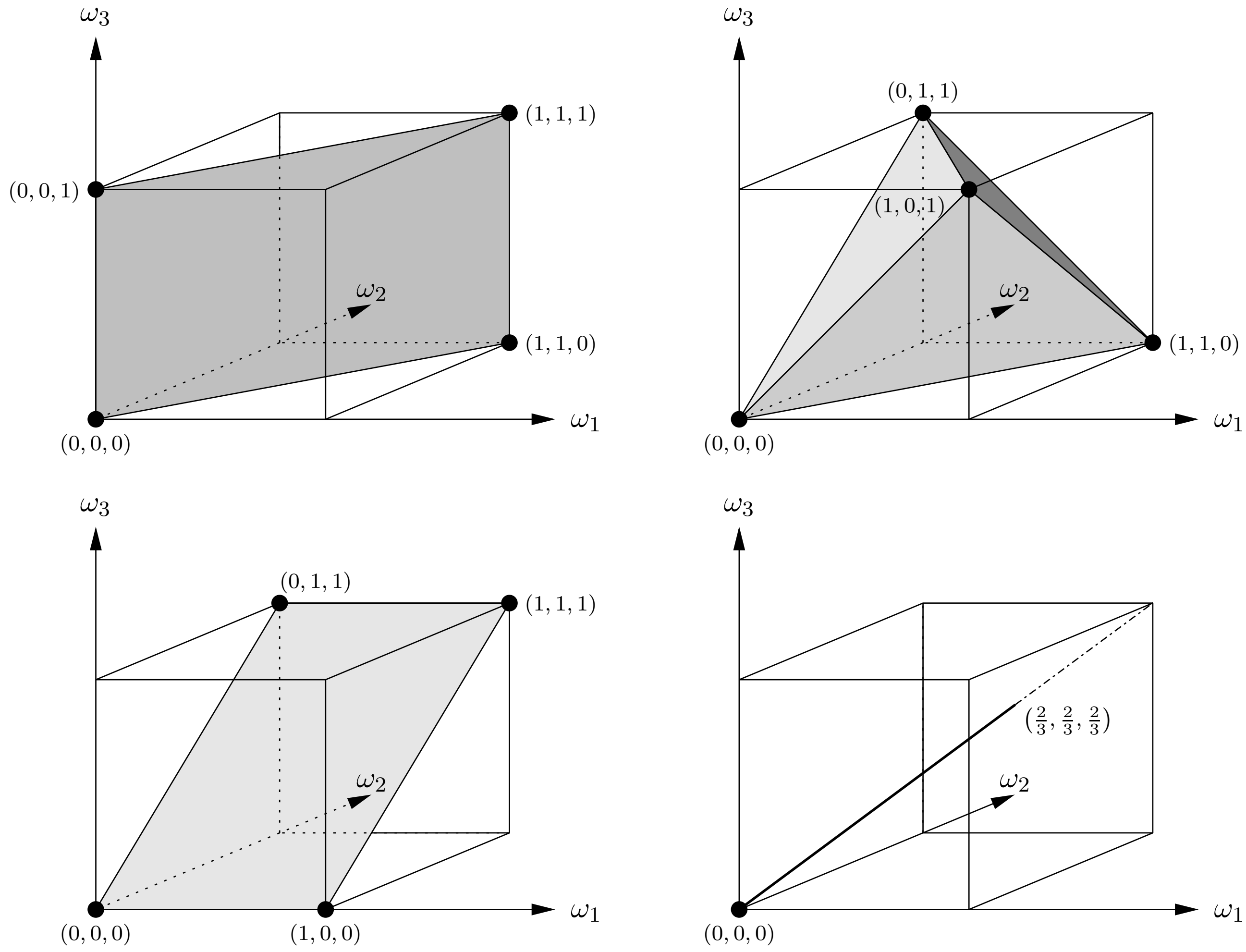
$$\operatorname{conv}(\mathcal{C}_2) = \operatorname{conv}\left(\{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1)\}\right)$$

$$= \left\{ \boldsymbol{\omega} \in [0,1]^3 \;\middle|\; \begin{array}{l} -\omega_1 + \omega_2 + \omega_3 \geq 0 \\ +\omega_1 - \omega_2 + \omega_3 \geq 0 \\ +\omega_1 + \omega_2 - \omega_3 \geq 0 \\ -\omega_1 - \omega_2 - \omega_3 \geq -2 \end{array} \right\}$$

$$\operatorname{conv}(\mathcal{C}_3) = \operatorname{conv}\left(\{(0,0,0),\ (0,1,1),\ (1,0,0),\ (1,1,1)\}\right)$$

$$= \left\{ \boldsymbol{\omega} \in [0,1]^3 \;\middle|\; \begin{array}{l} -\omega_2 + \omega_3 \geq 0 \\ +\omega_2 - \omega_3 \geq 0 \end{array} \right\}$$

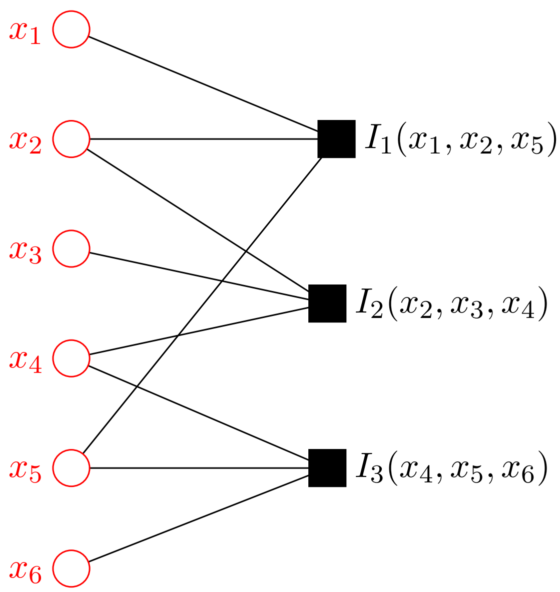
A Simple Code (Part 3)



$\mathrm{conv}(\mathcal{C}_1)$	$\mathrm{conv}(\mathcal{C}_2)$
$\mathrm{conv}(\mathcal{C}_3)$	$\mathcal{P}(\mathbf{H})$

Pseudo-codewords and Tanner graphs

Tanner / Factor graphs



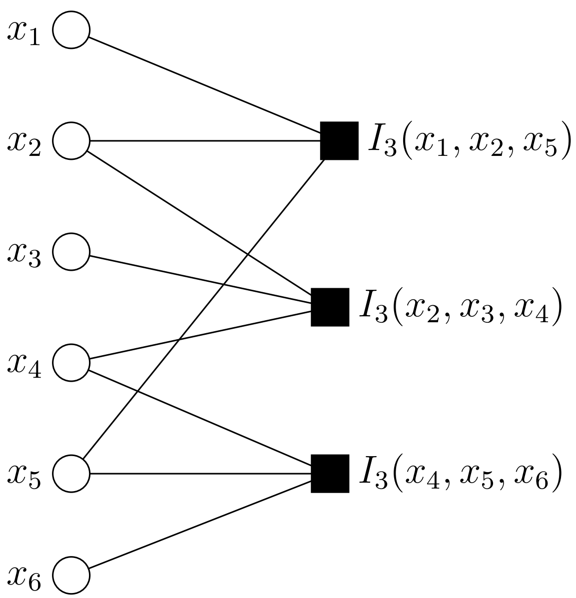
$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

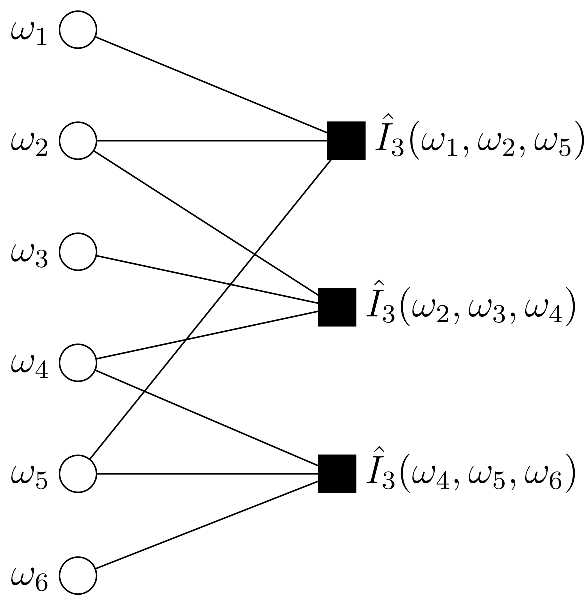
Codeword indicator function:

$$\begin{aligned} I_1(x_1, x_2, x_5) \cdot I_2(x_2, x_3, x_4) \cdot I_3(x_4, x_5, x_6) &= \big[(x_1, x_2, x_5) \in \mathcal{C}_1\big] \cdot \\ &\quad \big[(x_2, x_3, x_4) \in \mathcal{C}_2\big] \cdot \\ &\quad \big[(x_4, x_5, x_6) \in \mathcal{C}_3\big] \end{aligned}$$

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

Pseudo-Codewords / Fundamental Polytope





Codeword indicator function:

$$I_1(x_1, x_2, x_5) \cdot I_2(x_2, x_3, x_4) \cdot I_3(x_4, x_5, x_6)$$
$$= \big[(x_1, x_2, x_5) \in \mathcal{C}_1\big] \cdot \big[(x_2, x_3, x_4) \in \mathcal{C}_2\big] \cdot \big[(x_4, x_5, x_6) \in \mathcal{C}_3\big]$$

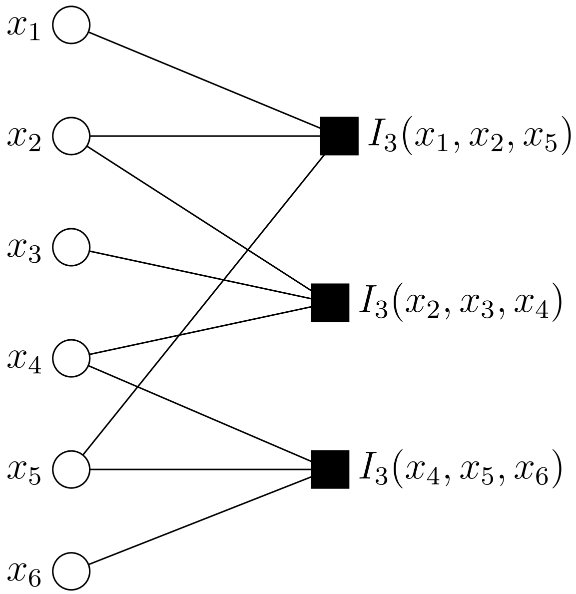
Note: $x_i \in \{0, 1\}$

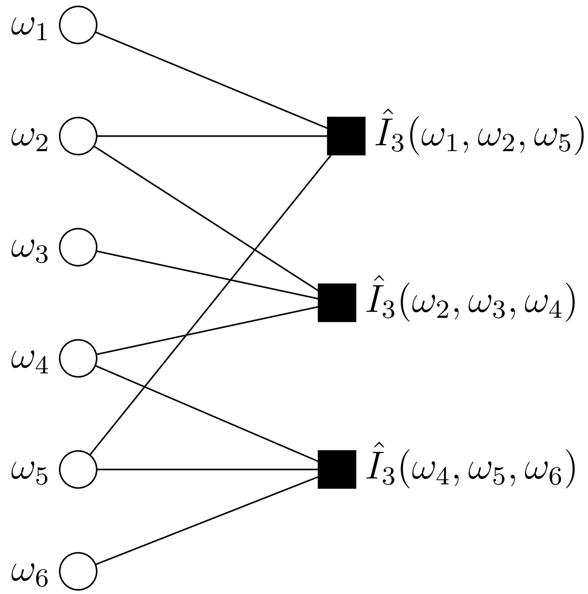
Pseudo-codeword indicator function:

$$\hat{I}_1(\omega_1, \omega_2, \omega_5) \cdot \hat{I}_2(\omega_2, \omega_3, \omega_4) \cdot \hat{I}_3(\omega_4, \omega_5, \omega_6)$$
$$= \big[(\omega_1, \omega_2, \omega_5) \in \mathrm{conv}(\mathcal{C}_1)\big] \cdot \big[(\omega_2, \omega_3, \omega_4) \in \mathrm{conv}(\mathcal{C}_2)\big] \cdot \big[(\omega_4, \omega_5, \omega_6) \in \mathrm{conv}(\mathcal{C}_3)\big]$$

Note: $0 \leq \omega_i \leq 1$

Pseudo-Codewords / Fundamental Cone





Codeword indicator function:

$$
\begin{aligned}
I_1(x_1, x_2, x_5) \cdot I_2(x_2, x_3, x_4) \cdot I_3(x_4, x_5, x_6)
&= \big[(x_1, x_2, x_5) \in \mathcal{C}_1\big] \cdot \\
&\quad \big[(x_2, x_3, x_4) \in \mathcal{C}_2\big] \cdot \\
&\quad \big[(x_4, x_5, x_6) \in \mathcal{C}_3\big]
\end{aligned}
$$

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

Pseudo-codeword indicator function:

$$
\begin{aligned}
\hat{I}_1(\omega_1, \omega_2, \omega_5) \cdot \hat{I}_2(\omega_2, \omega_3, \omega_4) \cdot \hat{I}_3(\omega_4, \omega_5, \omega_6)
&= \big[(\omega_1, \omega_2, \omega_5) \in \mathrm{conic}(\mathcal{C}_1)\big] \cdot \\
&\quad \big[(\omega_2, \omega_3, \omega_4) \in \mathrm{conic}(\mathcal{C}_2)\big] \cdot \\
&\quad \big[(\omega_4, \omega_5, \omega_6) \in \mathrm{conic}(\mathcal{C}_3)\big]
\end{aligned}
$$

Note: $0 \leq \omega_i$

Pseudo-Codewords / Fundamental Cone

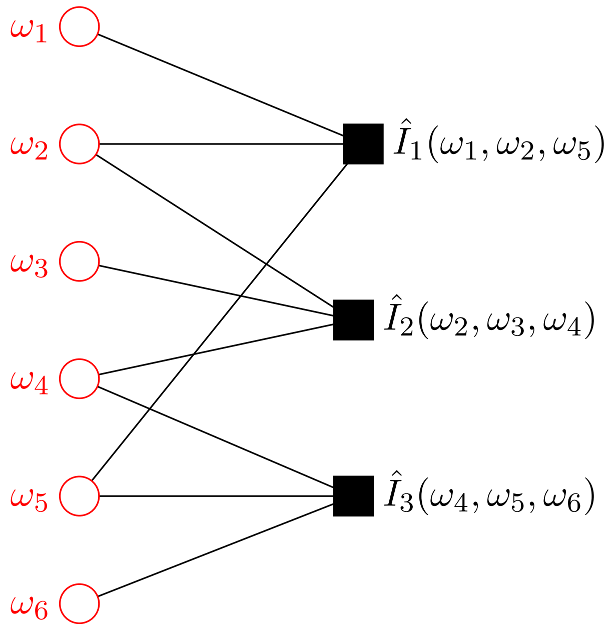
E.g.

$$\big[(\omega_1, \omega_2, \omega_5) \in \mathrm{conic}(\mathcal{C}_1)\big] = 1$$

if and only if

$$\omega_1 \leq \omega_2 + \omega_5$$
$$\omega_2 \leq \omega_1 + \omega_5$$
$$\omega_5 \leq \omega_1 + \omega_2$$

$$\omega_1 \geq 0$$
$$\omega_2 \geq 0$$
$$\omega_3 \geq 0$$



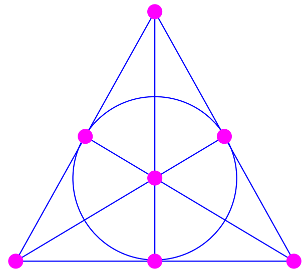
Pseudo-codeword indicator function:

$$\begin{aligned}\hat{I}_1(\omega_1, \omega_2, \omega_5) \cdot \hat{I}_2(\omega_2, \omega_3, \omega_4) \cdot \hat{I}_3(\omega_4, \omega_5, \omega_6) &= \big[(\omega_1, \omega_2, \omega_5) \in \mathrm{conic}(\mathcal{C}_1)\big] \cdot \\ &\quad \big[(\omega_2, \omega_3, \omega_4) \in \mathrm{conic}(\mathcal{C}_2)\big] \cdot \\ &\quad \big[(\omega_4, \omega_5, \omega_6) \in \mathrm{conic}(\mathcal{C}_3)\big]\end{aligned}$$

Note: $0 \leq \omega_i$

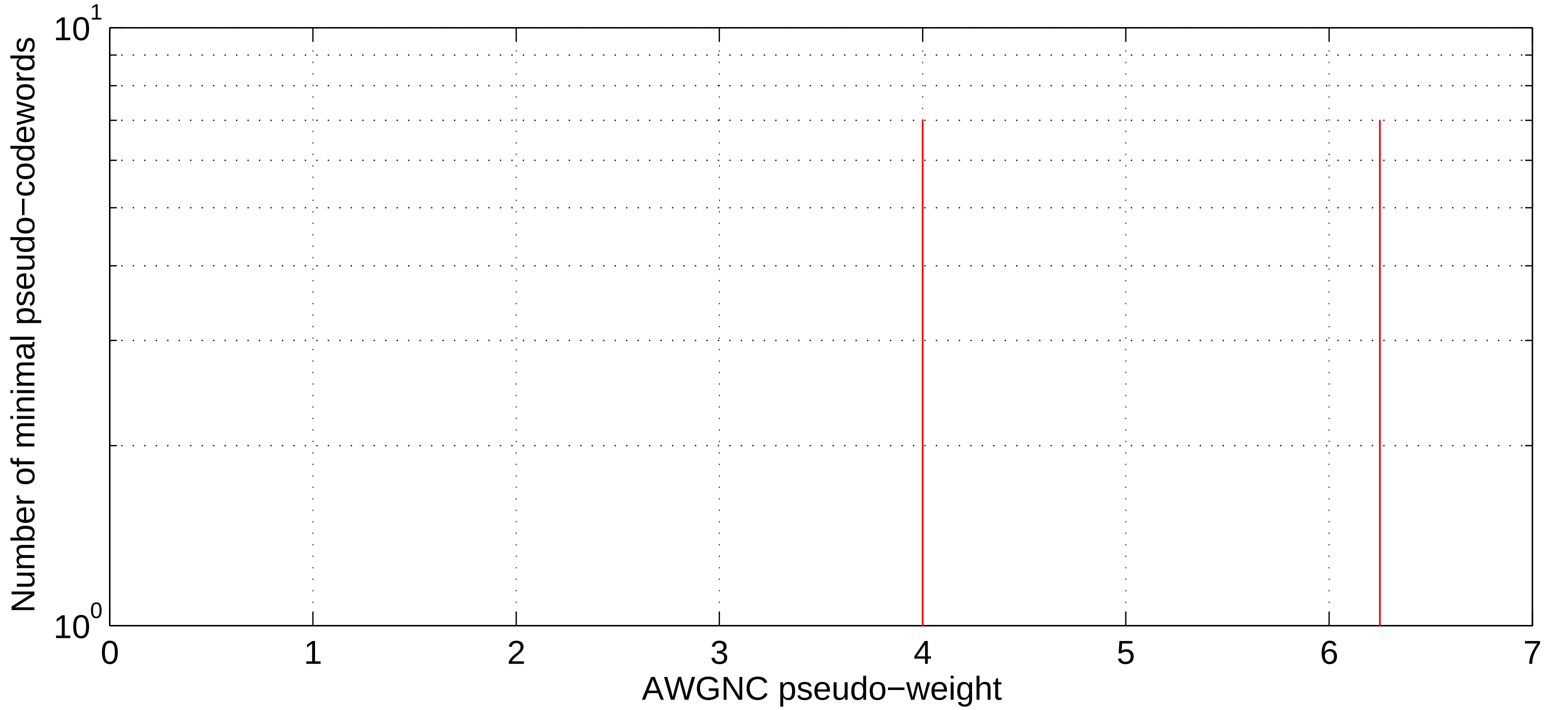
Pseudo-codeword spectra

Pseudo-Codeword Spectra (Part 1)



Consider the PG(2,2)-based $[7, 3, 4]$ binary linear code.

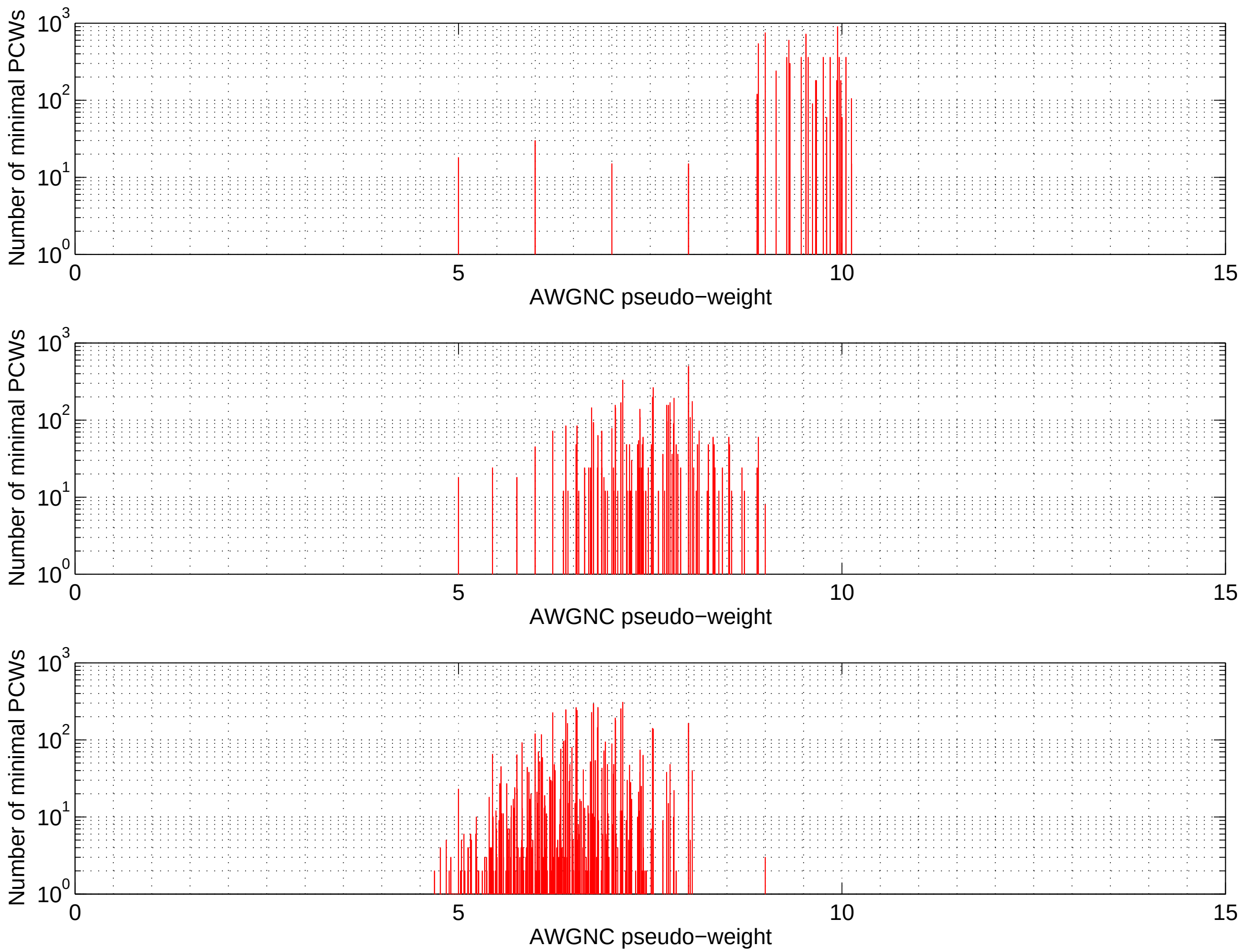
Here is its minimal pseudo-codeword spectrum:



Pseudo-Codeword Spectra (Part 2)

Consider the EG(2,4)-based $[15, 7, 5]$ binary linear code.

Here are some minimal pseudo-codeword spectra for different parity-check matrices of this code:



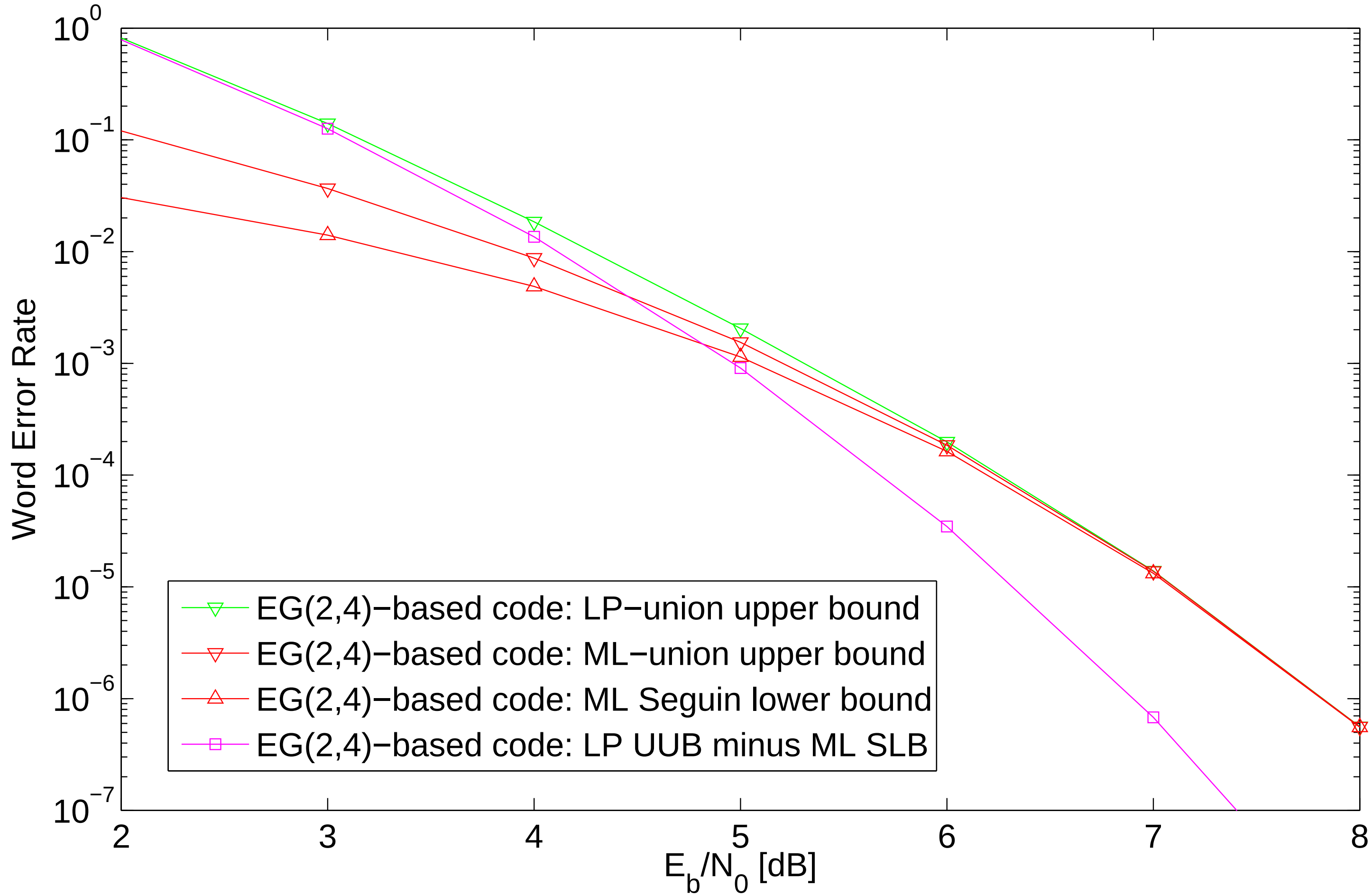
PCM of size 15×15

PCM of size 9×15

PCM of size 8×15

Pseudo-Codeword Spectra (Part 3)

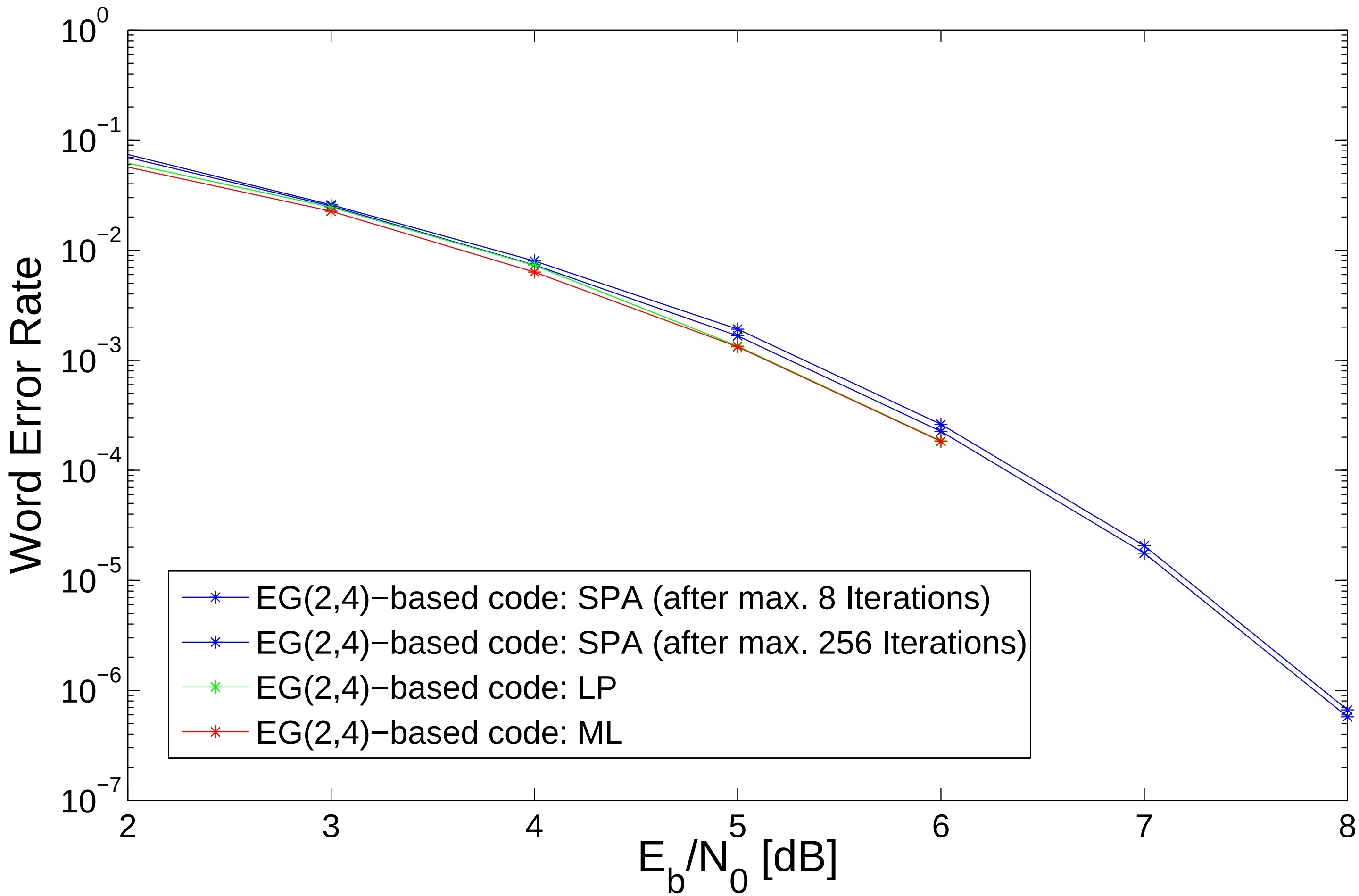
Consider the EG(2,4)-based $[15, 7, 5]$ binary linear code. The following plot shows upper and lower bounds on the word error rate of LP and ML decoding.



LABS hp

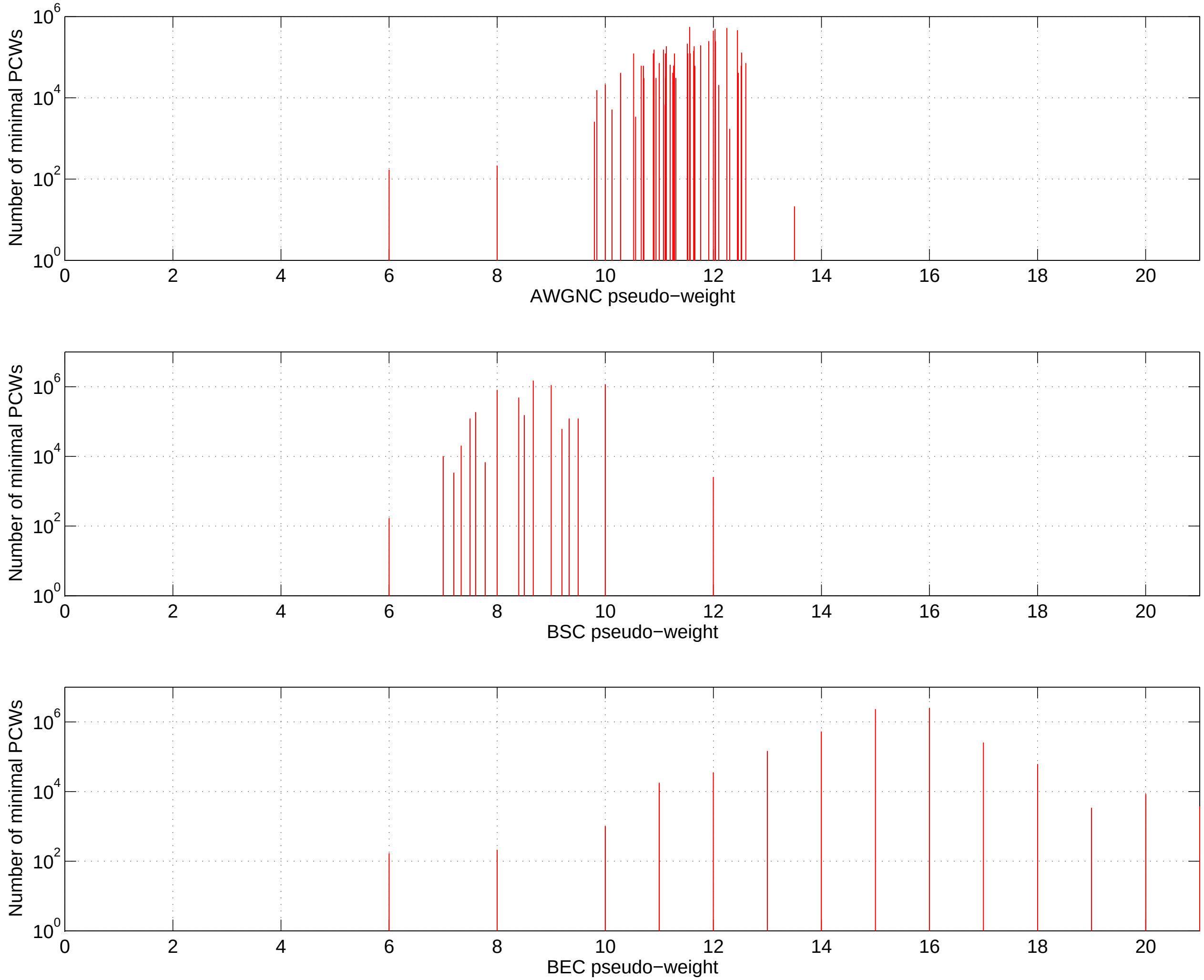
Pseudo-Codeword Spectra (Part 4)

Consider the EG(2,4)-based $[15, 7, 5]$ binary linear code. The following plot shows the word error rate for different decoding algorithms. (Note: LP/ML WER curves for small WER can be obtained from bounds shown in the previous plot.)



Pseudo-Codeword Spectra (Part 5)

Consider the PG(2,4)-based $[21, 11, 6]$ binary linear code.



Pseudo-Codeword Spectra (Part 6)

Some remarks:

- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes

Pseudo-Codeword Spectra (Part 6)

Some remarks:

- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes
  - where the minimal BEC pseudo-weight grows with growing block length,

Pseudo-Codeword Spectra (Part 6)

Some remarks:

- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes
  - where the minimal BEC pseudo-weight grows with growing block length,
  - but where the minimual AWGNC pseudo-weight is bounded from above.

Pseudo-Codeword Spectra (Part 6)

Some remarks:

- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes
  - where the minimal BEC pseudo-weight grows with growing block length,
  - but where the minimual AWGNC pseudo-weight is bounded from above.

$\Rightarrow$ It is important which channel is used!

Pseudo-Codeword Spectra (Part 6)

Some remarks:

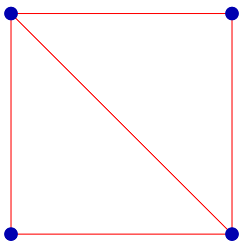
- Haley / Grant paper (ISIT 2005) presented a class of LDPC codes
    - where the minimal BEC pseudo-weight grows with growing block length,
    - but where the minimual AWGNC pseudo-weight is bounded from above.

  $\Rightarrow$ It is important which channel is used!

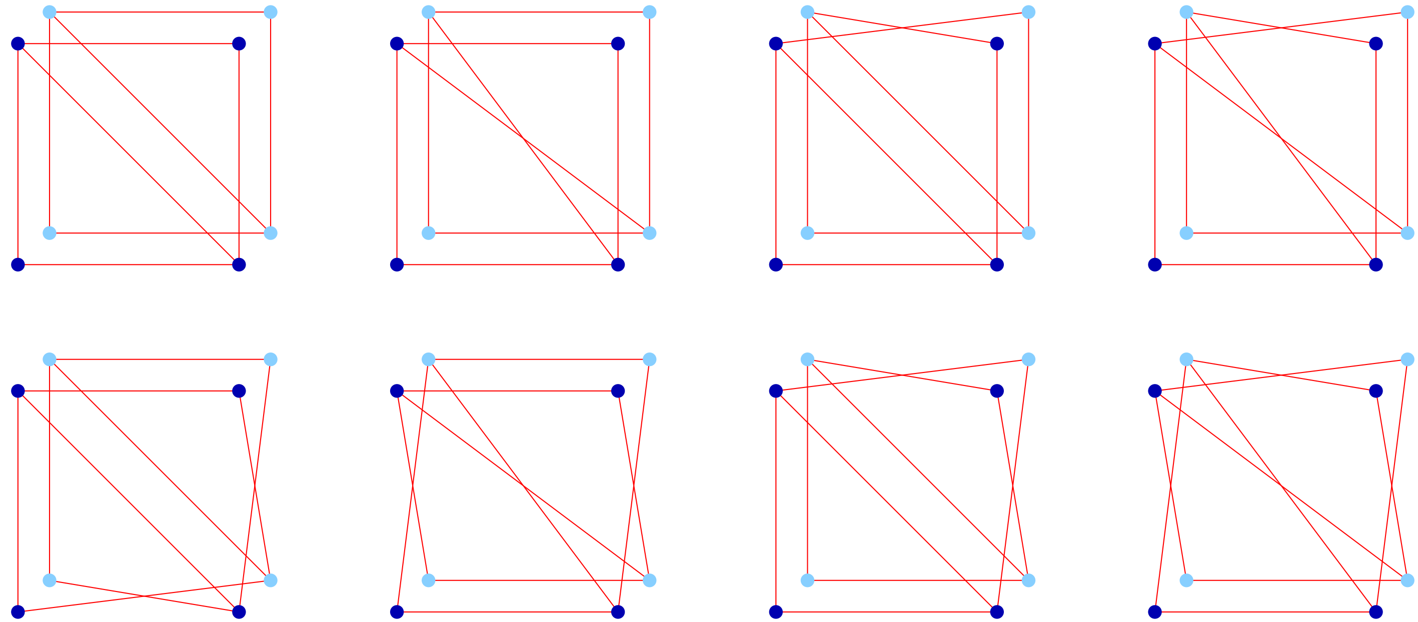
- Chertkov / Stepanov paper (ISIT 2007) presented an intesting heuristic for approximating the pseudo-weight spectra of minimal codewords for a given code.

Graph-cover interpretation of pseudo-codewords

Graph Covers (Part 1)



original graph

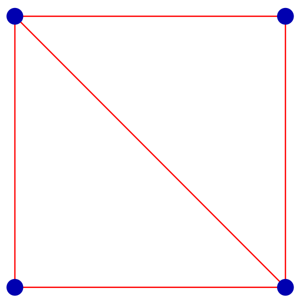


sample of possible
double covers of
the original graph

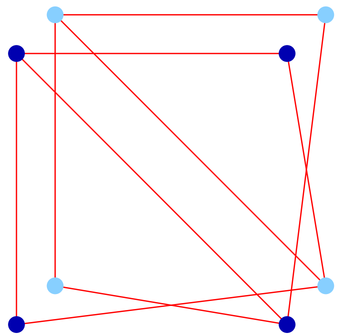
Definition: A double cover of a graph is . . .

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

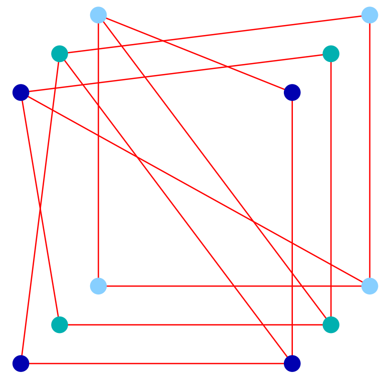
Graph Covers (Part 2)



original graph



(a possible)
double cover of
the original graph

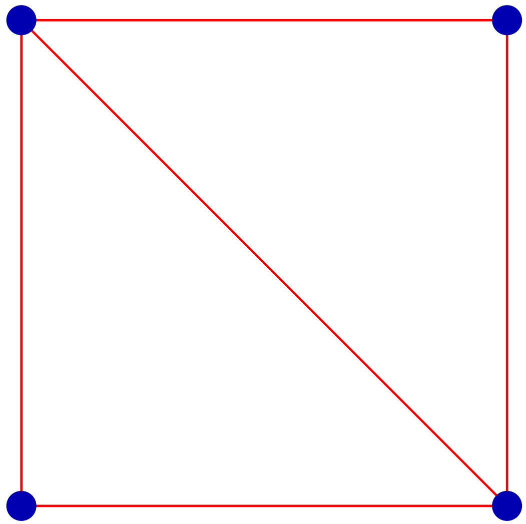


(a possible)
triple cover of
the original graph

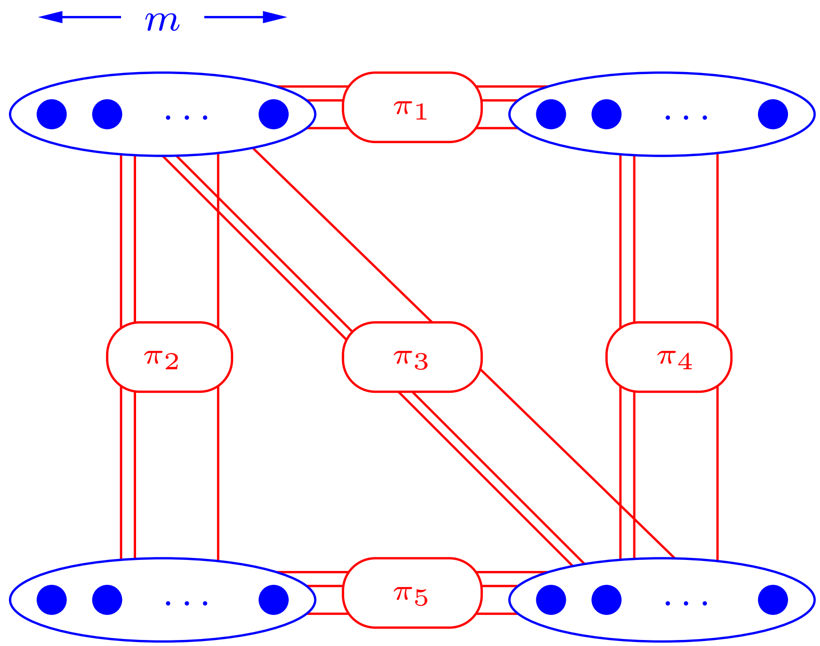
...

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

Graph Covers (Part 3)



original graph



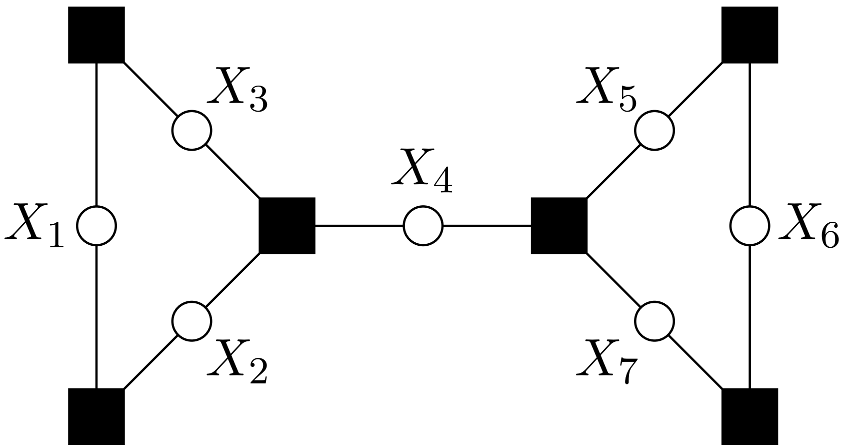
(possible)
m-fold cover of
original graph

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.

Codewords in Graph Covers (Part 1)

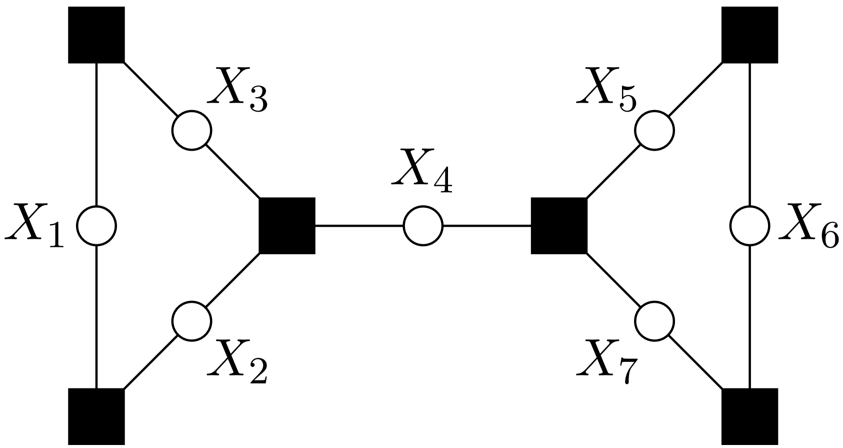
We can also consider covers of Tanner/factor graphs. Here is e.g. a possible double cover of some Tanner/factor graph.



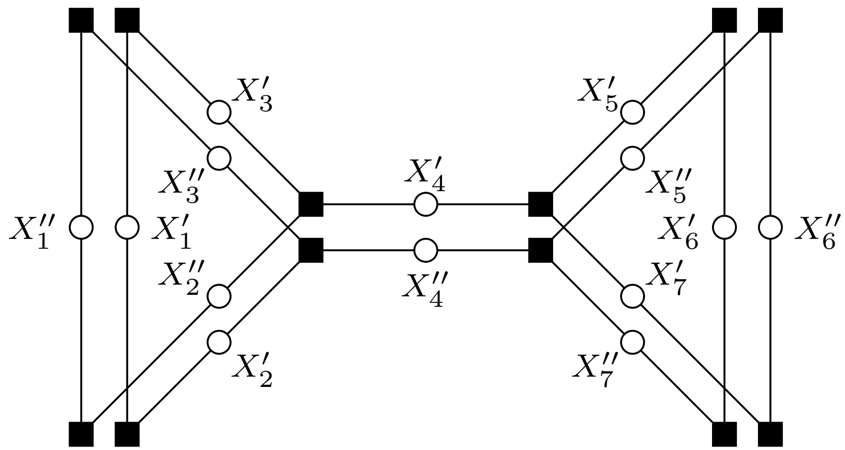
Base factor/Tanner graph
of a length-7 code

Codewords in Graph Covers (Part 1)

We can also consider covers of Tanner/factor graphs. Here is e.g. a possible double cover of some Tanner/factor graph.



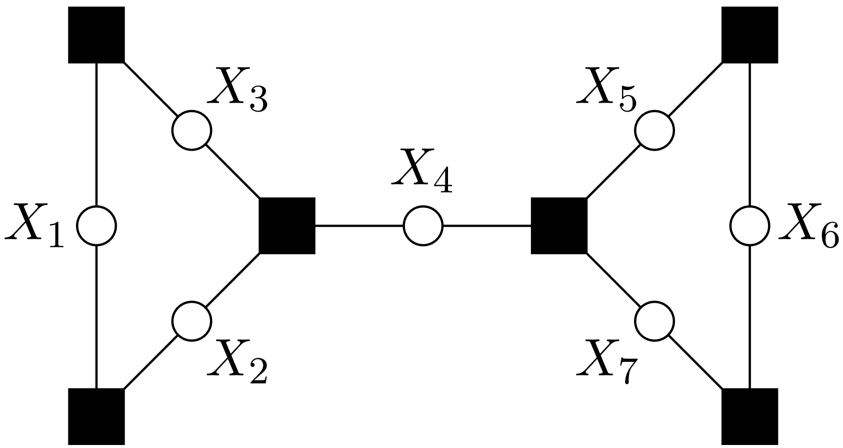
Base factor/Tanner graph
of a length-7 code



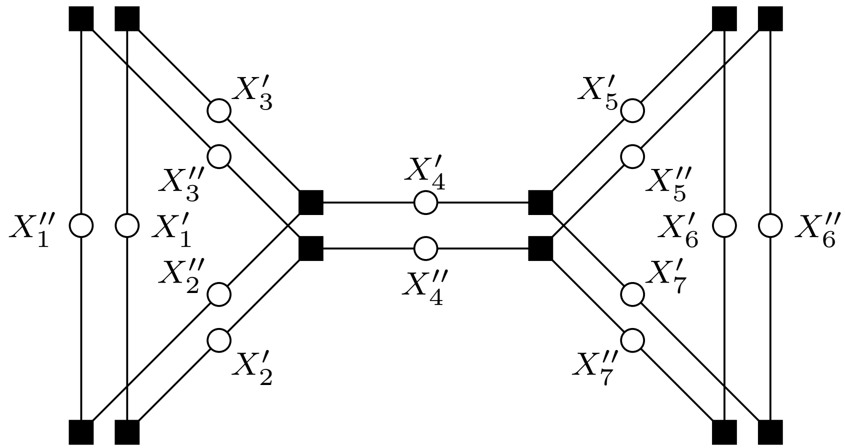
Possible double cover of
the base Tanner/factor graph

Codewords in Graph Covers (Part 1)

We can also consider covers of Tanner/factor graphs. Here is e.g. a possible double cover of some Tanner/factor graph.



Base factor/Tanner graph
of a length-7 code

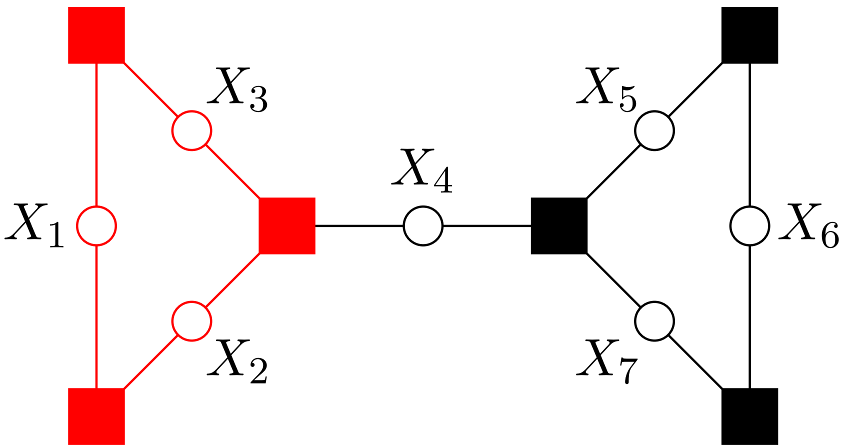


Possible double cover of
the base Tanner/factor graph

Let us study the codes defined by the graph covers of the base Tanner/factor graph.

Codewords in Graph Covers (Part 2)

Obviously, any codeword in the base Tanner/factor graph can be lifted to a codeword in the double cover of the base Tanner/factor graph.

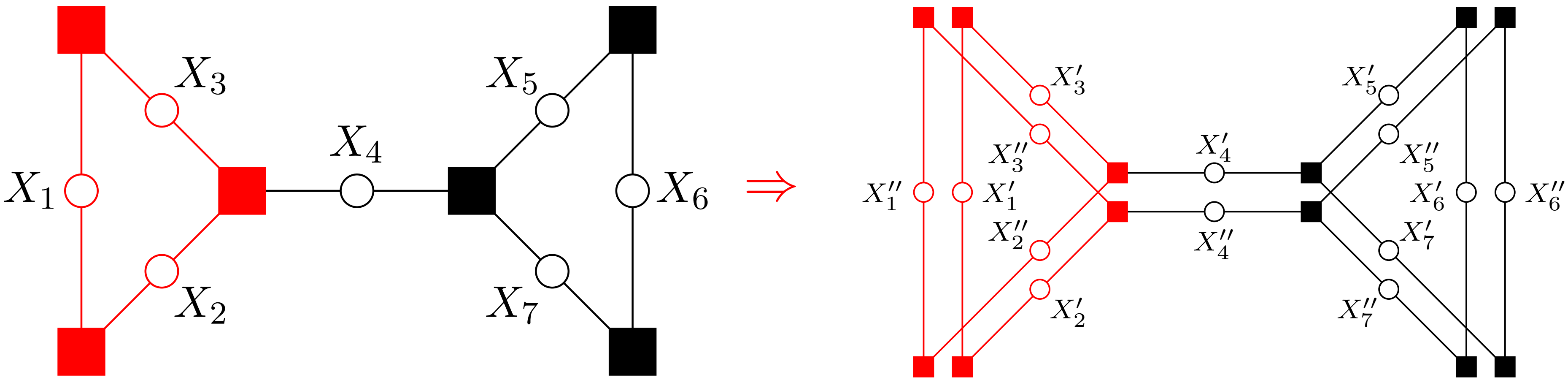


$\Rightarrow$

$$(1, 1, 1, 0, 0, 0, 0)$$

Codewords in Graph Covers (Part 2)

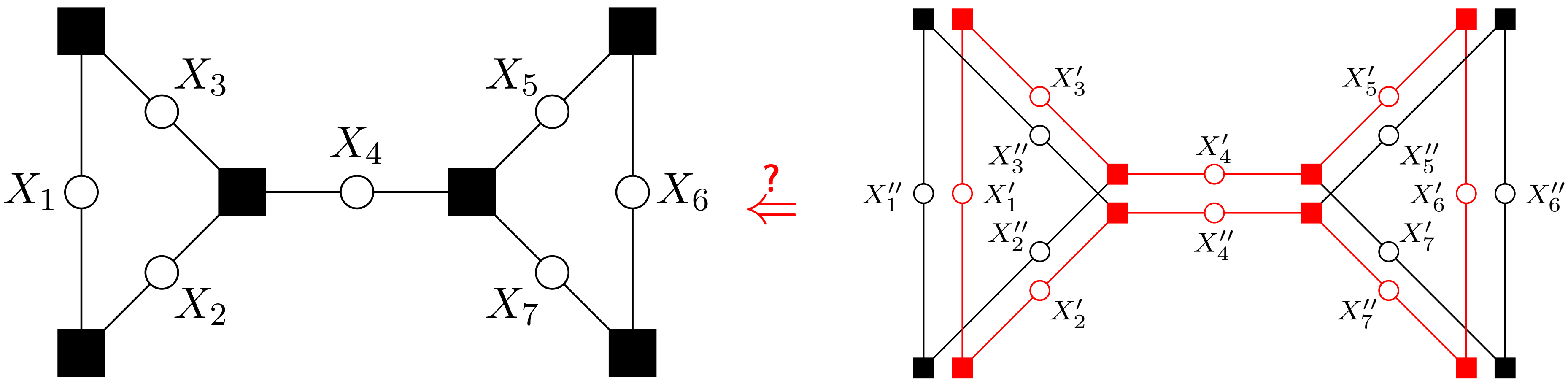
Obviously, any codeword in the base Tanner/factor graph can be lifted to a codeword in the double cover of the base Tanner/factor graph.



$(1, 1, 1, 0, 0, 0, 0)$ $\qquad$ $(1{:}1,\ 1{:}1,\ 1{:}1,\ 0{:}0,\ 0{:}0,\ 0{:}0,\ 0{:}0)$

Codewords in Graph Covers (Part 3)

But in the double cover of the base Tanner/factor graph there are also codewords that are not liftings of codewords in the base Tanner/factor graph!

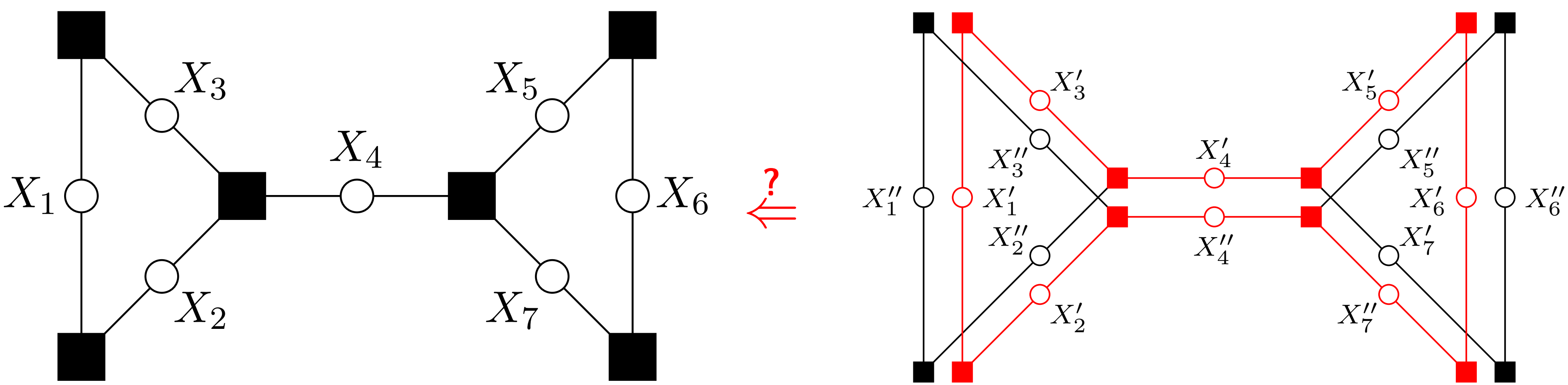


?

$$(1{:}0,\ 1{:}0,\ 1{:}0,\ 1{:}1,\ 1{:}0,\ 1{:}0,\ 0{:}1)$$

Codewords in Graph Covers (Part 3)

But in the double cover of the base Tanner/factor graph there are also codewords that are not liftings of codewords in the base Tanner/factor graph!



What about

$$\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{2}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right) ?$$

$$(1{:}0,\ 1{:}0,\ 1{:}0,\ 1{:}1,\ 1{:}0,\ 1{:}0,\ 0{:}1)$$

Codewords in Graph Covers (Part 4)

Theorem:

Codewords in Graph Covers (Part 4)

Theorem:

- Let $\mathcal{P} \triangleq \mathcal{P}(\mathbf{H})$ be the fundamental polytope of a parity-check matrix $\mathbf{H}$.

Codewords in Graph Covers (Part 4)

Theorem:

- Let $\mathcal{P} \triangleq \mathcal{P}(\mathbf{H})$ be the fundamental polytope of a parity-check matrix $\mathbf{H}$.
- Let $\mathcal{P}'$ be the set of all vectors obtained through codewords in finite covers.

Codewords in Graph Covers (Part 4)

Theorem:

- Let $\mathcal{P} \triangleq \mathcal{P}(\mathbf{H})$ be the fundamental polytope of a parity-check matrix $\mathbf{H}$.
- Let $\mathcal{P}'$ be the set of all vectors obtained through codewords in finite covers.

Then, $\mathcal{P}'$ is dense in $\mathcal{P}$, i.e.

$$\mathcal{P}' = \mathcal{P} \cap \mathbb{Q}^n$$

$$\mathcal{P} = \mathrm{closure}(\mathcal{P}').$$

Codewords in Graph Covers (Part 4)

Theorem:

- Let $\mathcal{P} \triangleq \mathcal{P}(\mathbf{H})$ be the fundamental polytope of a parity-check matrix $\mathbf{H}$.
- Let $\mathcal{P}'$ be the set of all vectors obtained through codewords in finite covers.

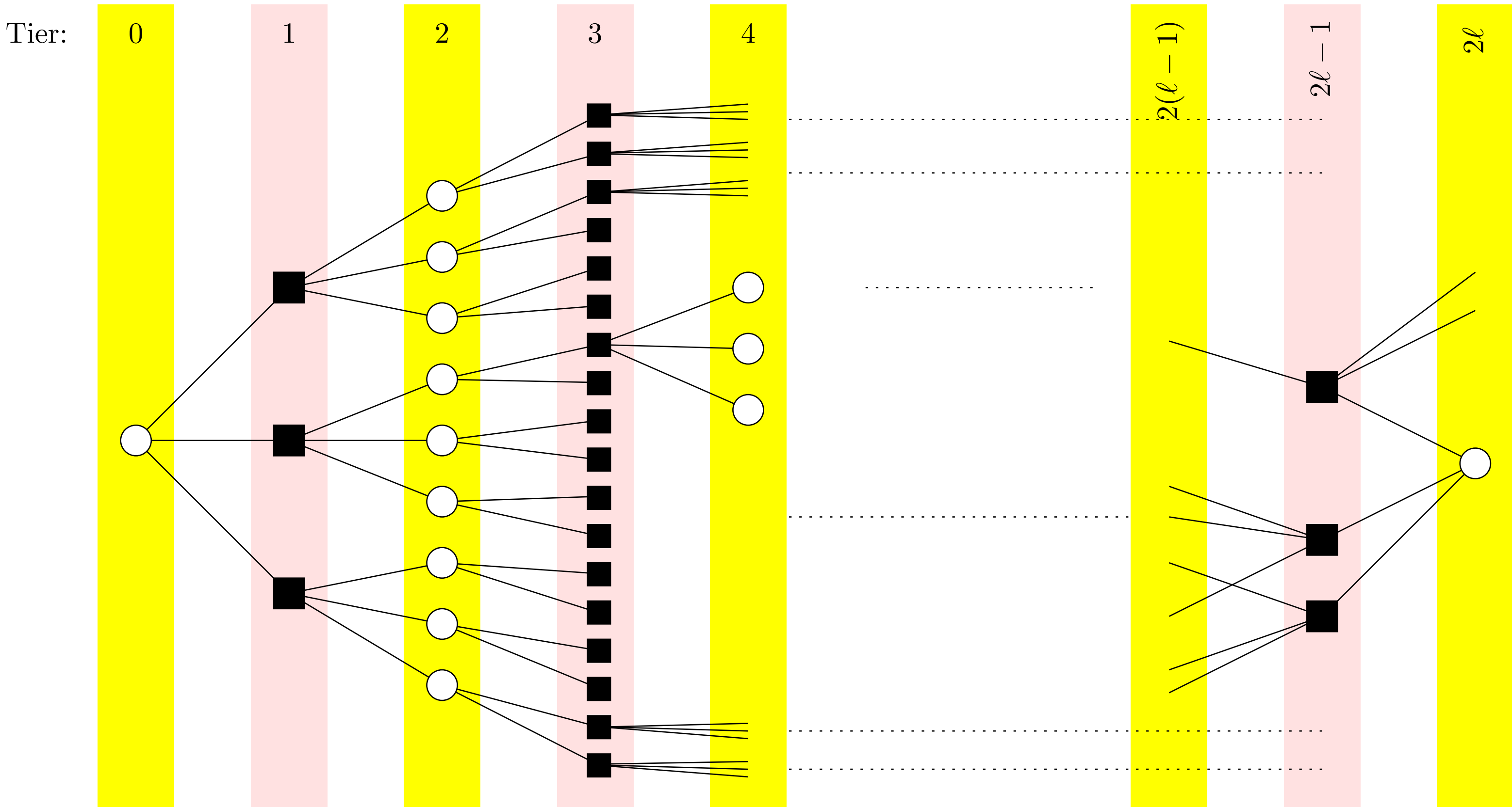
Then, $\mathcal{P}'$ is dense in $\mathcal{P}$, i.e.

$$\mathcal{P}' = \mathcal{P} \cap \mathbb{Q}^n$$
$$\mathcal{P} = \mathrm{closure}(\mathcal{P}').$$

Moreover, note that all vertices of $\mathcal{P}$ are vectors with rational entries and are therefore also in $\mathcal{P}'$.

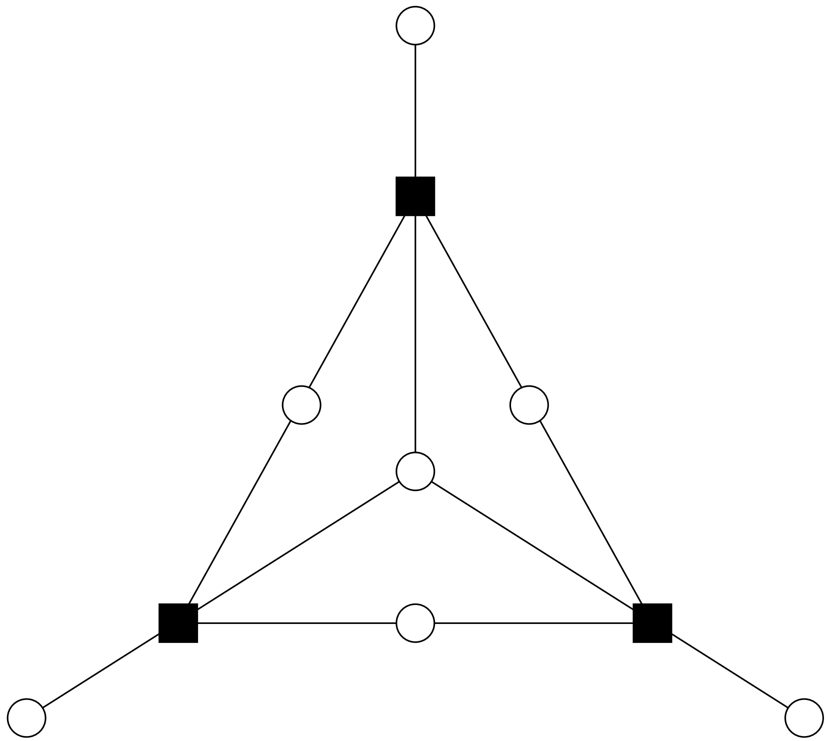
The canonical completion

Trying to Construct a Codeword



Pseudo-Codewords: the Canonical Completion

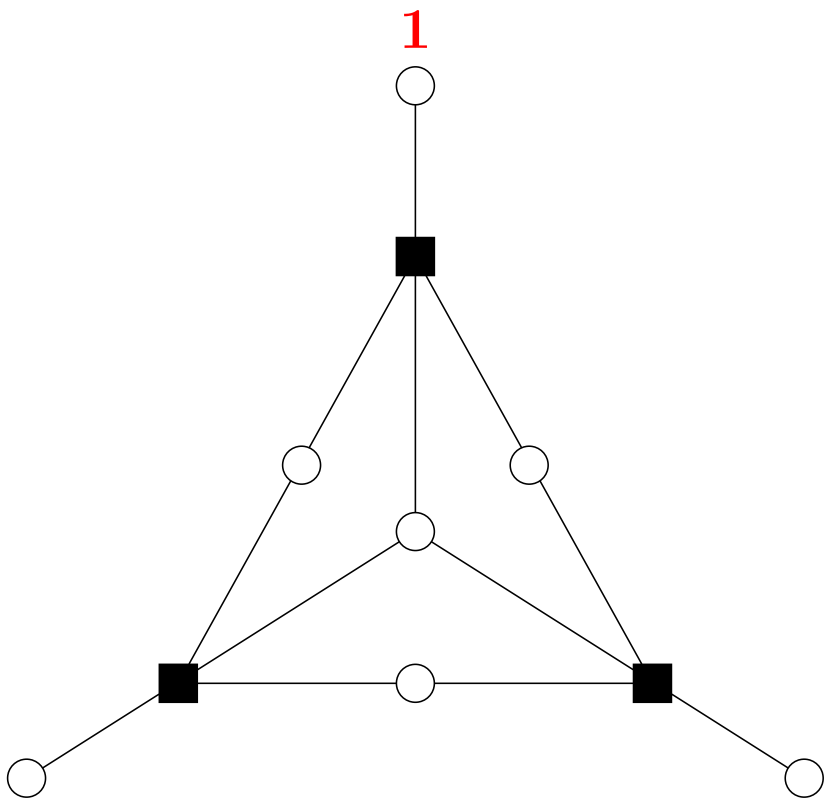
Example: $[7, 4, 3]$ binary Hamming code.



Note that all checks have degree $k = 4$. $\Rightarrow$ completion factor $\frac{1}{k-1} = \frac{1}{3}$.

Pseudo-Codewords: the Canonical Completion

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

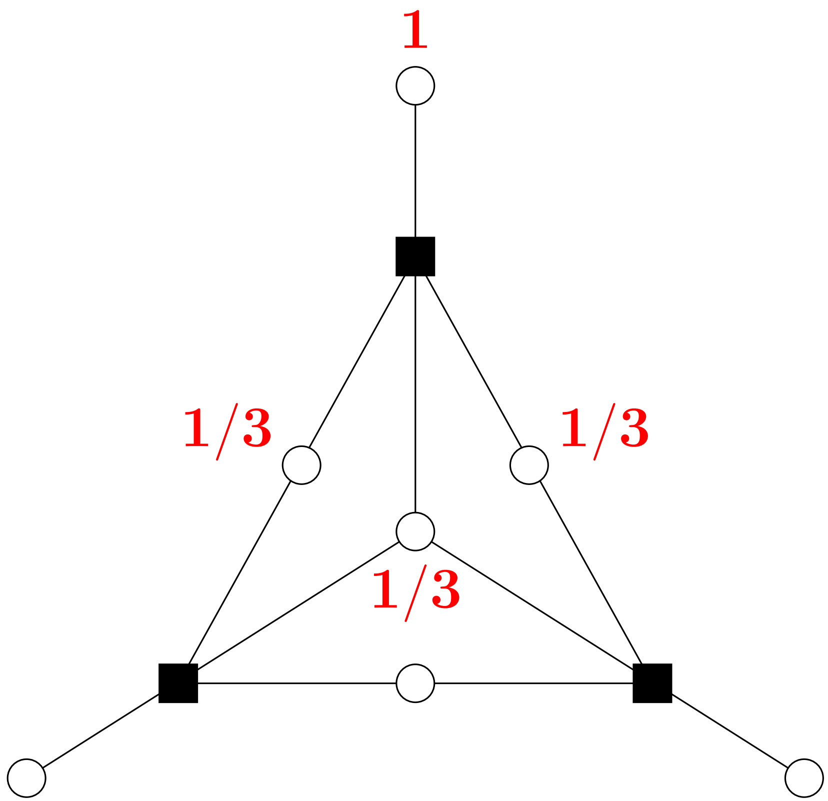


Note that all checks have degree $k = 4$. $\Rightarrow$ completion factor $\frac{1}{k-1} = \frac{1}{3}$.

LABS hp

Pseudo-Codewords: the Canonical Completion

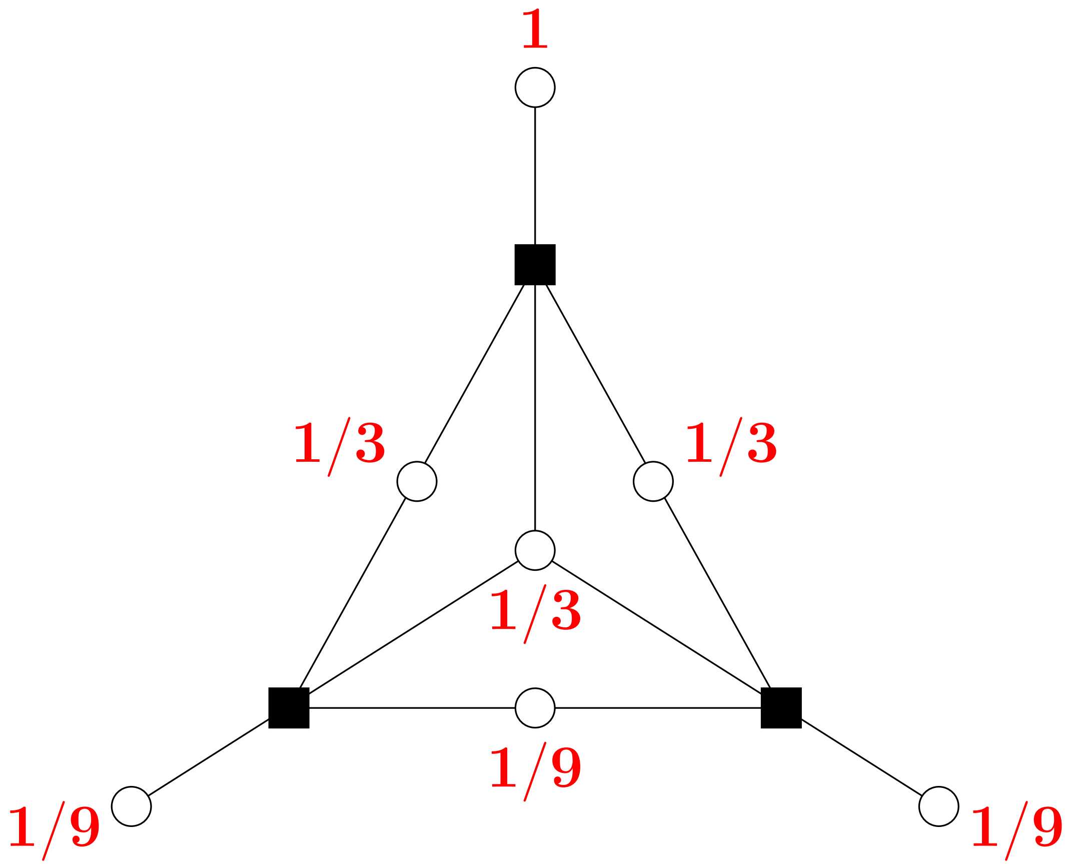
Example: $[7, 4, 3]$ binary Hamming code.



Note that all checks have degree $k = 4$. $\Rightarrow$ completion factor $\frac{1}{k-1} = \frac{1}{3}$.

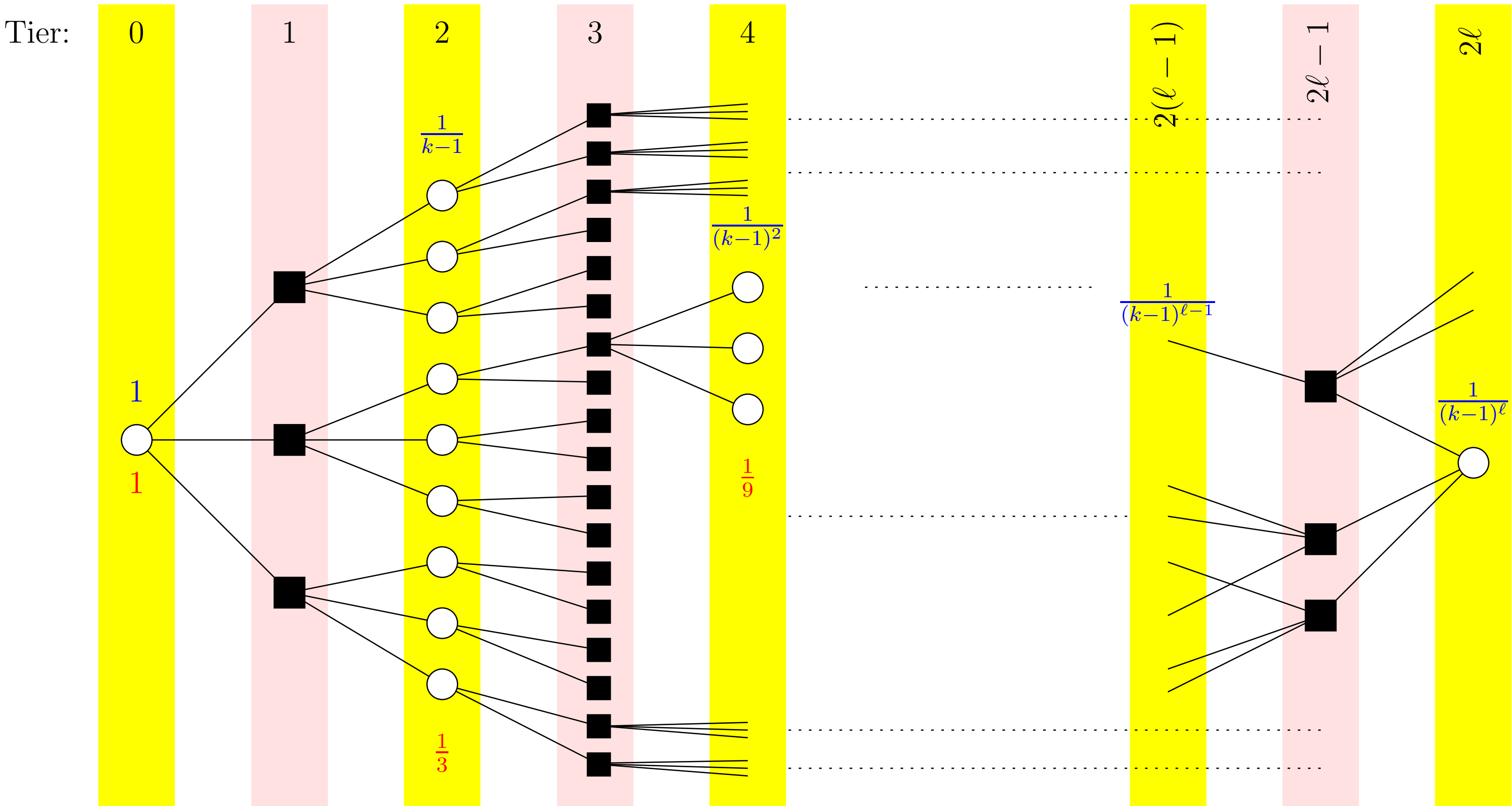
Pseudo-Codewords: the Canonical Completion

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

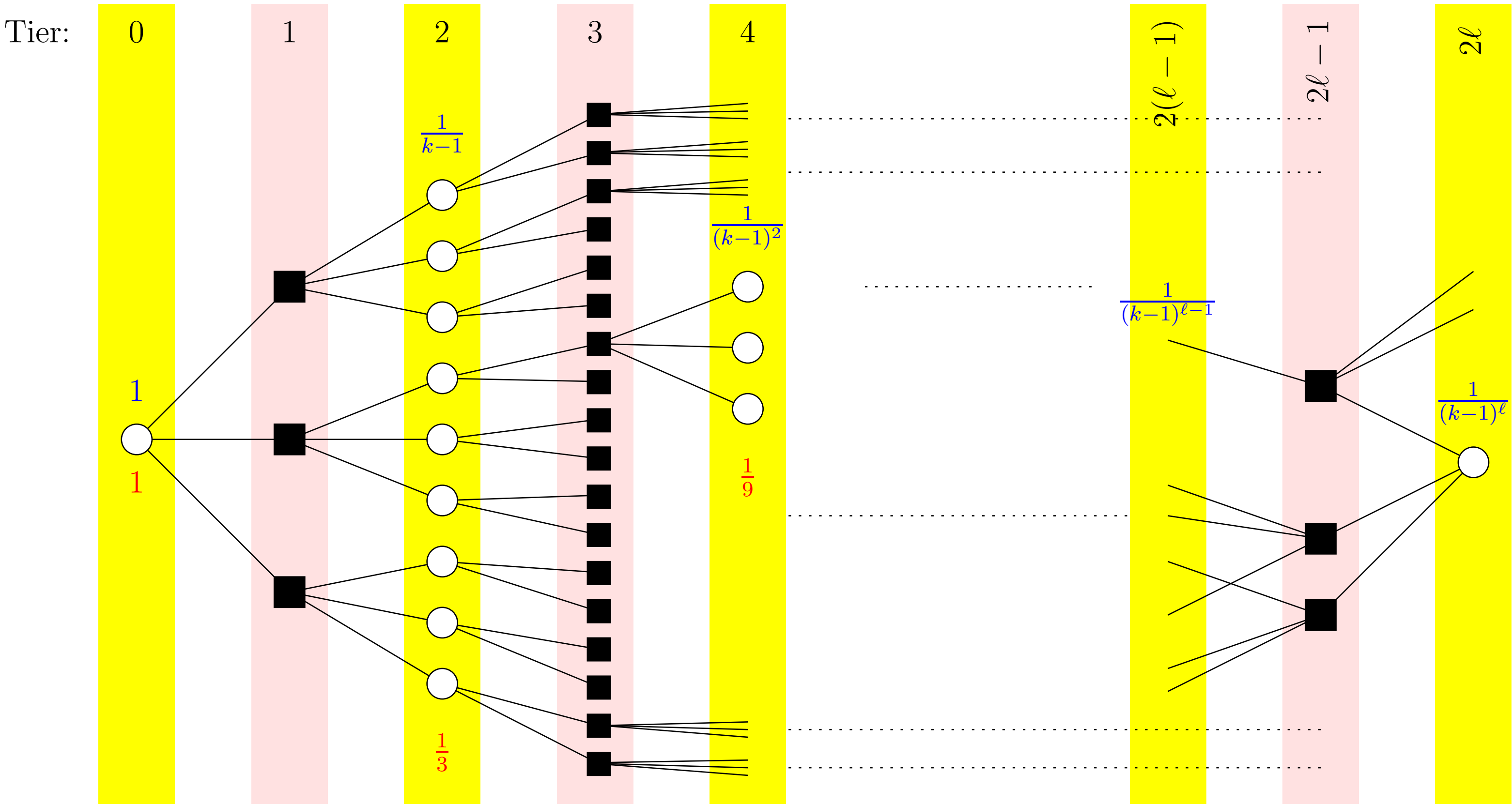


Note that all checks have degree $k = 4$. $\Rightarrow$ completion factor $\frac{1}{k-1} = \frac{1}{3}$.

Pseudo-Codewords: the Canonical Completion



Pseudo-Codewords: the Canonical Completion



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

An Upper Bound on the Minimum Pseudo-Weight based on Can. Compl.

An Upper Bound on the Minimum Pseudo-Weight based on Can. Compl.

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

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

where

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

An Upper Bound on the Minimum Pseudo-Weight based on Can. Compl.

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

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

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 $\{\mathcal{C}_i\}$ of (j,k)-regular LDPC codes of increasing length satisfies

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

Influence

of redundant rows in the parity-check matrix

and of cycles in the Tanner graph

A Tanner Graph with Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \cdots \end{matrix}$$

A Tanner Graph with Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \cdots \end{matrix}$$

$$\tilde{\mathbf{H}} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{12}) \\ \cdots \end{matrix}$$

A Tanner Graph with Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \cdots \end{array}$$

$$\tilde{\mathbf{H}} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{12}) \\ \cdots \end{array}$$

If the support of the blue and the green line coincide in at least two position then we have

$$\mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \;\supseteq\; \mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_{12}).$$

A Tanner Graph without Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \cdots \end{matrix}$$

A Tanner Graph without Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_2) \\ \cdots \end{array}$$

$$\tilde{\mathbf{H}} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \operatorname{conv}(\mathcal{C}_{12}) \\ \cdots \end{array}$$

A Tanner Graph without Four-Cycles

Observation:

$$\mathbf{H} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \cdots \end{array}$$

$$\tilde{\mathbf{H}} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \cdots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \quad \Rightarrow \quad \begin{array}{c} \cdots \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{12}) \\ \cdots \end{array}$$

If the support of the blue and the green line coincide in at most one position then we have

$$\mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \;=\; \mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_{12}).$$

Tanner Graphs with/without Four-Cycles

Proposition: It seems to be favorable to have no four-cycles in the Tanner graph: "we get some inequalities for free!"

Tanner Graphs with/without Four-Cycles

Proposition: It seems to be favorable to have no four-cycles in the Tanner graph: "we get some inequalities for free!"

Note: this argument can be easily extended to Tanner graphs with no six-cycles, no eight-cycles, etc.

Obtaining tighter Relaxations

Let the relaxation $\mathrm{relax}(\mathcal{C})$ of $\mathcal{C}$ be the set of all vectors $\boldsymbol{\omega} \in \mathbb{R}^5$ that fulfill three conditions:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_3) \end{array}$$

Therefore,

$$\mathrm{relax}(\mathcal{C}) \triangleq \mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_3).$$

How well can we do by adding more (redundand) lines to the parity-check matrix?

Obtaining tighter Relaxations (Part 2)

What about taking a parity-check matrix $\mathbf{H}'$ that contains all the non-zero codewords from the dual code?

$$\mathbf{H}' = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_1) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_2) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_3) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{12}) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{13}) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{23}) \\ \boldsymbol{\omega} \in \mathrm{conv}(\mathcal{C}_{123}) \end{array}$$

$$\mathrm{relax}'(\mathcal{C}) \triangleq \mathrm{conv}(\mathcal{C}_1) \cap \mathrm{conv}(\mathcal{C}_2) \cap \mathrm{conv}(\mathcal{C}_3) \cap \mathrm{conv}(\mathcal{C}_{12}) \cap \mathrm{conv}(\mathcal{C}_{13}) \cap \mathrm{conv}(\mathcal{C}_{23}) \cap \mathrm{conv}(\mathcal{C}_{123}).$$

Obtaining tighter Relaxations (Part 3)

Translating a theorem from matroid theory we get the following result:

Theorem (Seymour 1981) We have

$$\mathrm{relax}'(\mathcal{C}) = \mathrm{conv}(\mathcal{C})$$

if and only if there is no way to shorten and puncture $\mathcal{C}$ such that we get the codes F_7^*, $M(K_5)$, or R_{10}.

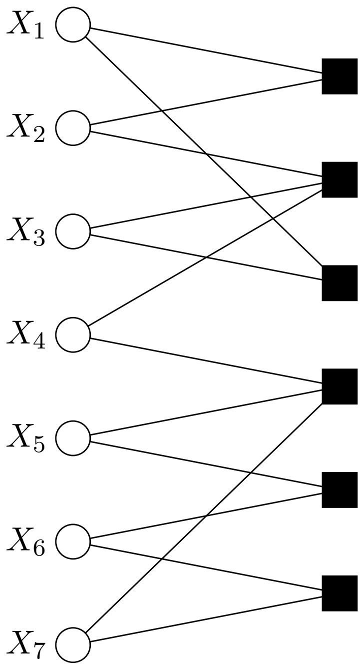
F_7^*:	$[7, 3, 4]$ code
$M(K_5)$:	$[10, 6, 3]$ code
R_{10}:	$[10, 5, 4]$ code

Pseudo-codwords and the edge zeta function

Tanner/Factor Graph of a Cycle Code

Cycle codes are codes which have a Tanner/factor graph where all bit nodes have degree two. (Equivalently, the parity-check matrix has two ones per column.)

Example:

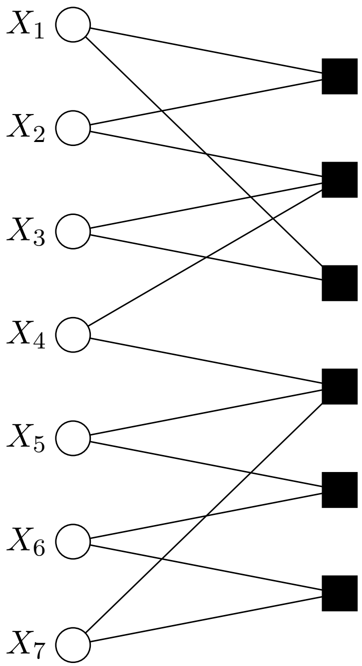


Tanner/factor graph

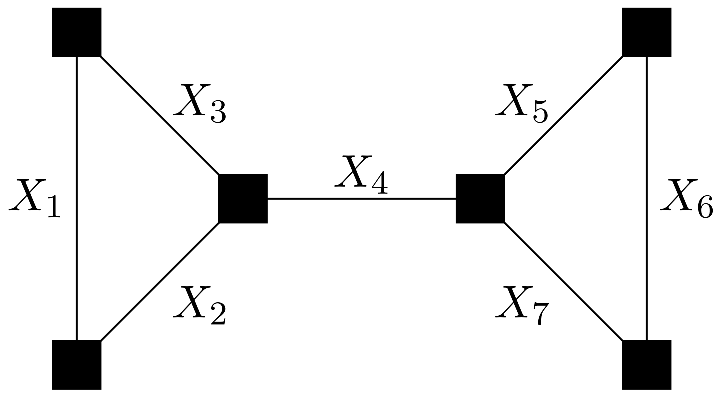
Tanner/Factor Graph of a Cycle Code

Cycle codes are codes which have a Tanner/factor graph where all bit nodes have degree two. (Equivalently, the parity-check matrix has two ones per column.)

Example:



Tanner/factor graph

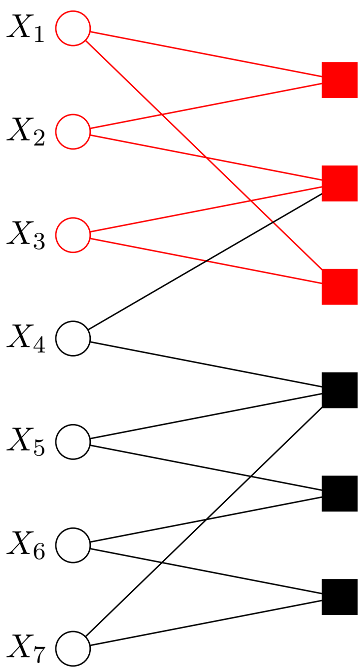


Corresponding normal factor graph

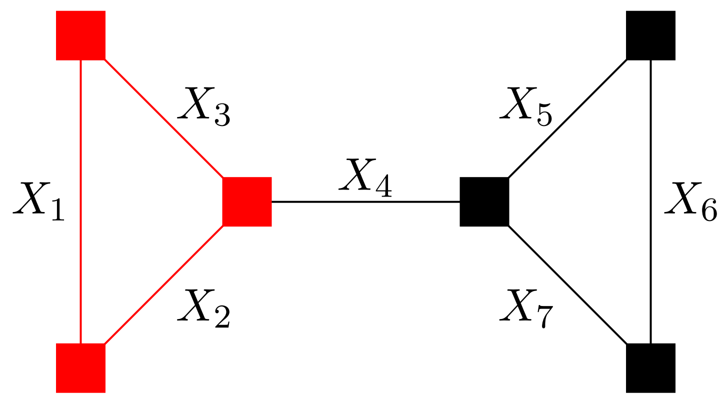
Tanner/Factor Graph of a Cycle Code

Cycle codes are called cycle codes because codewords correspond to simple cycles (or to the symmetric difference set of simple cycles) in the Tanner/factor graph.

Example:



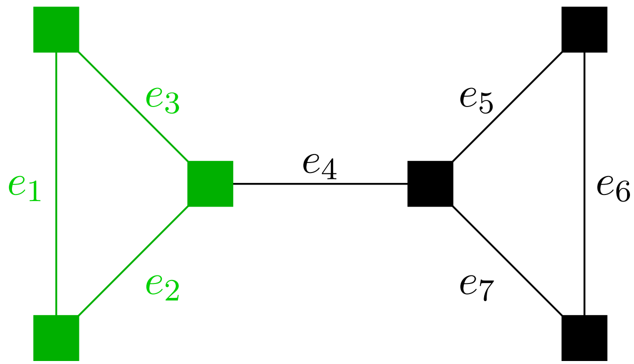
Tanner/factor graph



Corresponding normal factor graph

The Edge Zeta Function of a Graph

Definition (Hashimoto, see also Stark/Terras):



Here: $\Gamma = (e_1, e_2, e_3)$

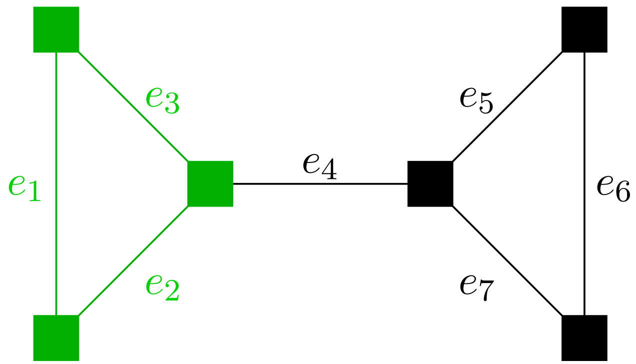
Let Γ be a path in a graph X with edge-set E; write

$$\Gamma = (e_{i_1}, \ldots, e_{i_k})$$

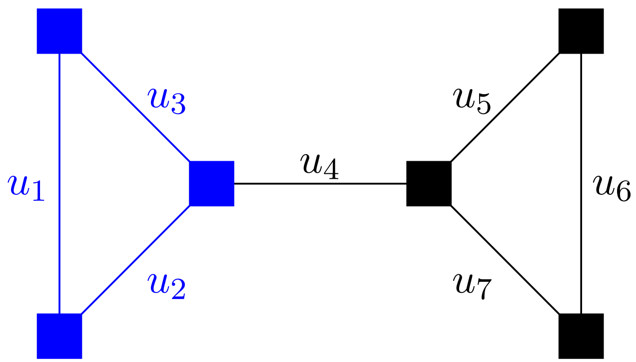
to indicate that Γ begins with the edge e_{i_1} and ends with the edge e_{i_k}.

The Edge Zeta Function of a Graph

Definition (Hashimoto, see also Stark/Terras):



Here: $\Gamma = (e_1, e_2, e_3)$



Here: $g(\Gamma) = u_1 u_2 u_3$

Let Γ be a path in a graph X with edge-set E; write

$$\Gamma = (e_{i_1}, \ldots, e_{i_k})$$

to indicate that Γ begins with the edge e_{i_1} and ends with the edge e_{i_k}.

The monomial of Γ is given by

$$g(\Gamma) \triangleq u_{i_1} \cdots u_{i_k},$$

where the u_i's are indeterminates.

The Edge Zeta Function of a Graph

Definition (Hashimoto, see also Stark/Terras):

The edge zeta function of X is defined to be the power series

$$\zeta_X(u_1, \ldots, u_n) \in \mathbb{Z}[[u_1, \ldots, u_n]]$$

given by

$$\zeta_X(u_1, \ldots, u_n) = \prod_{[\Gamma] \in A(X)} \frac{1}{1 - g(\Gamma)},$$

where $A(X)$ is the collection of equivalence classes of backtrackless, tailless, primitive cycles in X.

Note: unless X contains only one cycle, the set $A(X)$ will be countably infinite.

The Edge Zeta Function of a Graph

Theorem (Bass):

- The edge zeta function $\zeta_X(u_1, \ldots, u_n)$ is a rational function.
- More precisely, for any directed graph $\vec{X}$ of X, we have

$$\zeta_X(u_1, \ldots, u_n) = \frac{1}{\det\left(\mathbf{I} - \mathbf{U}\mathbf{M}(\vec{X})\right)} = \frac{1}{\det\left(\mathbf{I} - \mathbf{M}(\vec{X})\mathbf{U}\right)}$$

where

- $\mathbf{I}$ is the identity matrix of size $2n$,
- $\mathbf{U} = \text{diag}(u_1, \ldots, u_n, u_1, \ldots, u_n)$ is a diagonal matrix of indeterminants.
- $\mathbf{M}(\vec{X})$ is a $2n \times 2n$ matrix derived from some directed graph version $\vec{X}$ of X.

Relationship Pseudo-Codewords and Edge Zeta Function (Part 1: Theorem)

Theorem:

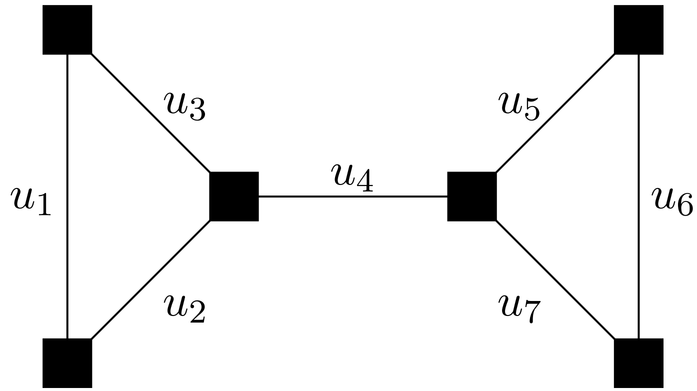
- Let C be a cycle code defined by a parity-check matrix $\mathbf{H}$ having normal graph $N \triangleq N(\mathbf{H})$.
- Let $n = n(N)$ be the number of edges of N.
- Let $\zeta_N(u_1, \ldots, u_n)$ be the edge zeta function of N.
- Then

the monomial $u_1^{p_1} \ldots u_n^{p_n}$ has a nonzero coefficient in the Taylor series expansion of ζ_N

if and only if

the corresponding exponent vector $(p_1, \ldots, p_n)$ is an unscaled pseudo-codeword for C.

Relationship Pseudo-Codewords and Edge Zeta Function (Part 2: Example)

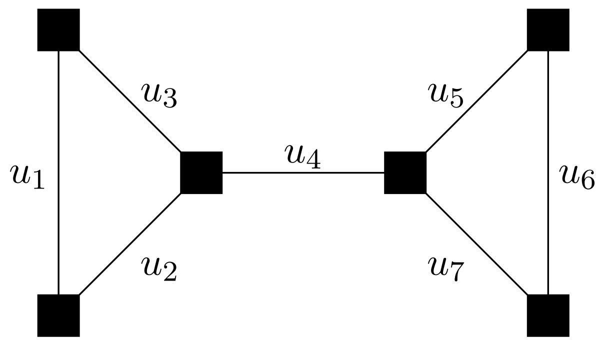


This normal graph N has the following inverse edge zeta function:

$$\zeta_N(u_1,\ldots,u_7) = \frac{1}{\det(\mathbf{I}_{14} - \mathbf{UM})}$$

$$= \frac{1}{\begin{array}{l} 1 - 2u_1u_2u_3 + u_1^2u_2^2u_3^2 - 2u_5u_6u_7 + 4u_1u_2u_3u_5u_6u_7 - 2u_1^2u_2^2u_3^2u_5u_6u_7 \\ -4u_1u_2u_3u_4^2u_5u_6u_7 + 4u_1^2u_2^2u_3^2u_4^2u_5u_6u_7 + u_5^2u_6^2u_7^2 - 2u_1u_2u_3u_5^2u_6^2u_7^2 \\ +u_1^2u_2^2u_3^2u_5^2u_6^2u_7^2 + 4u_1u_2u_3u_4^2u_5^2u_6^2u_7^2 - 4u_1^2u_2^2u_3^2u_4^2u_5^2u_6^2u_7^2 \end{array}}$$

Relationship Pseudo-Codewords and Edge Zeta Function (Part 3: Example)



The Taylor series exansion is

$$\zeta_N(u_1, \ldots, u_7)$$

$$= 1 + 2u_1u_2u_3 + 3u_1^2u_2^2u_3^2 + 2u_5u_6u_7$$

$$+ 4u_1u_2u_3u_5u_6u_7 + 6u_1^2u_2^2u_3^2u_5u_6u_7$$

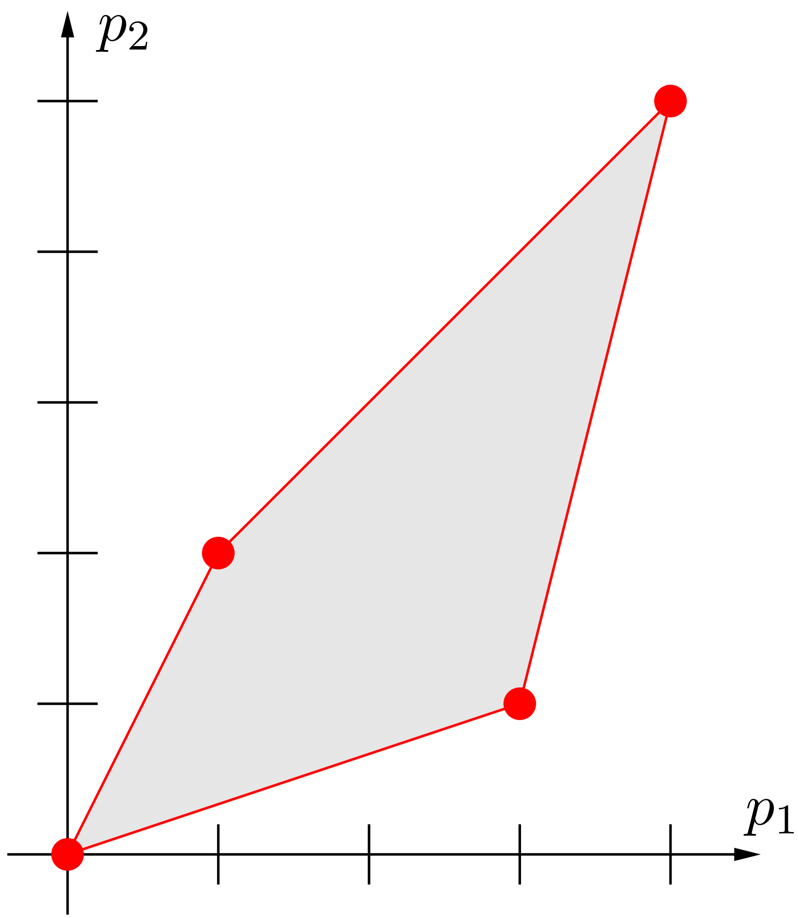
$$+ 4u_1u_2u_3u_4^2u_5u_6u_7 + 12u_1^2u_2^2u_3^2u_4^2u_5u_6u_7$$

$$+ \cdots$$

We get the following exponent vectors:

$(0,0,0,0,0,0,0)$	codeword
$(1,1,1,0,0,0,0)$	codeword
$(2,2,2,0,0,0,0)$	pseudo-codeword (in $\mathbb{Z}$-span)
$(0,0,0,0,1,1,1)$	codeword
$(1,1,1,0,1,1,1)$	codeword
$(2,2,2,0,1,1,1)$	pseudo-codeword (in $\mathbb{Z}$-span)
$(1,1,1,2,1,1,1)$	pseudo-codeword (not in $\mathbb{Z}$-span)
$(2,2,2,2,1,1,1)$	pseudo-codeword (in $\mathbb{Z}$-span)

The Newton Polytope of a Polynomial



Here: $P(u_1, u_2)$
$= u_1^0 u_2^0 + 3u_1^1 u_2^2 + 4u_1^3 u_2^1 - 2u_1^4 u_2^5$

Definition:
The Newton polytope of a polynomial $P(u_1, \ldots, u_n)$ in n indeterminates is the convex hull of the points in n-dimensional space given by the exponent vectors of the nonzero monomials appearing in $P(u_1, \ldots, u_n)$.

Similarly, we can associate a polyhedron to a power series.

Characterizing the Fundamental Cone Through the Zeta Function

Collecting the results from the previous slides we get:

Proposition: Let $\mathcal{C}$ be some cycle code with parity-check matrix $\mathbf{H}$ and normal factor graph $N(\mathbf{H})$.

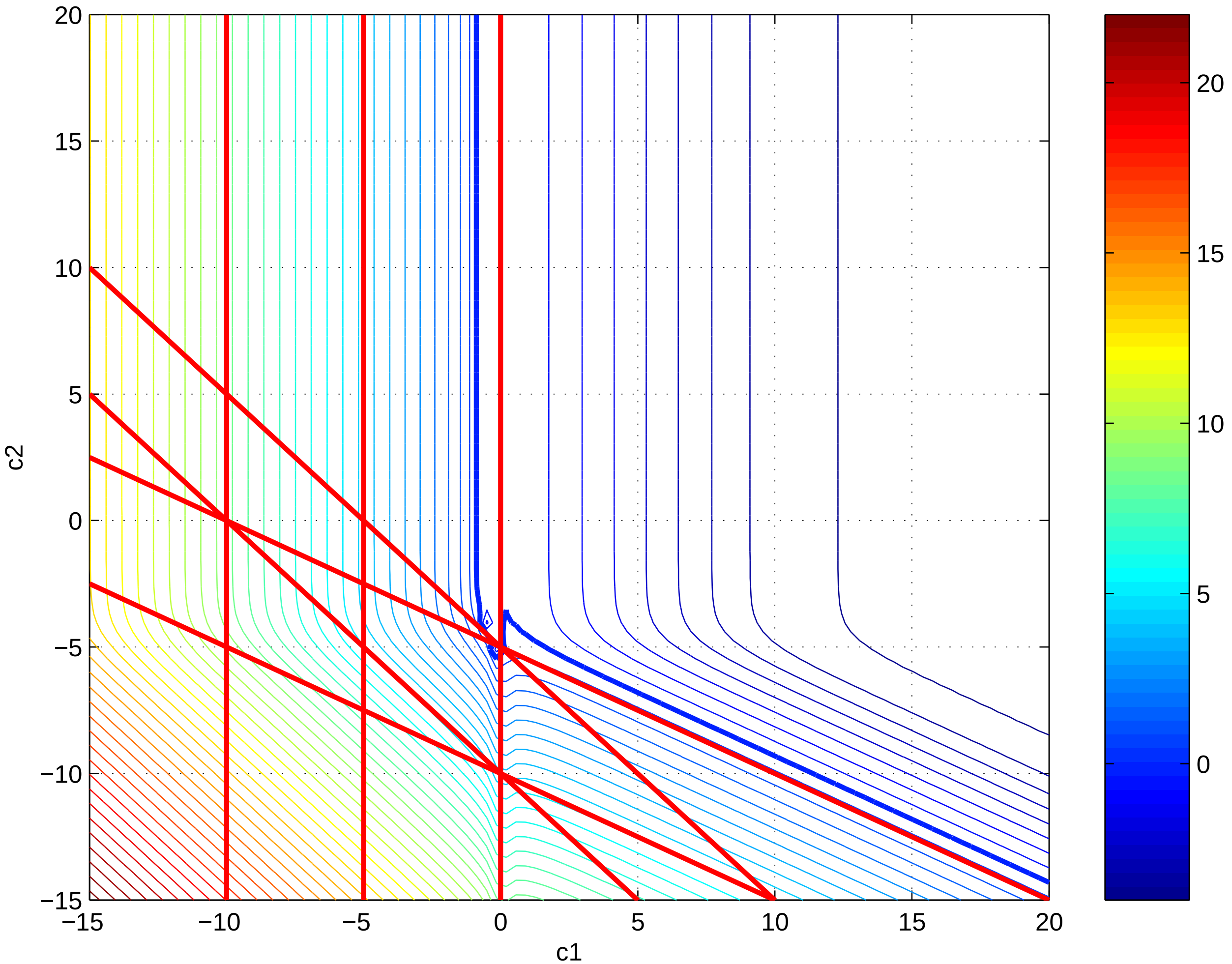
The Newton polyhedron of the zeta function of $N(\mathbf{H})$

equals

the fundamental cone $\mathcal{K}(\mathbf{H})$.

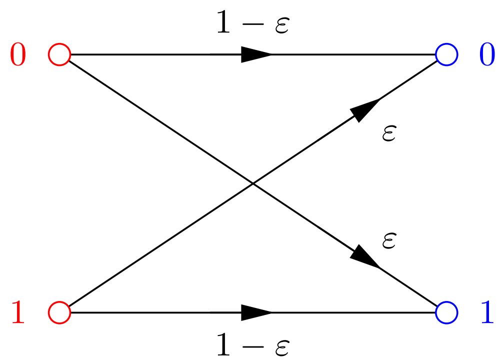
Characterizing the Fundamental Cone Through the Zeta Function

The inverse of the zeta function seems to give some valuable information about the dual cone of the fundamental cone.



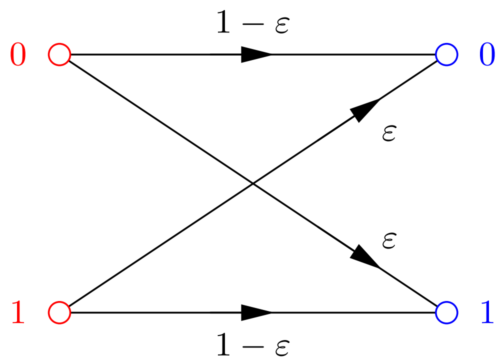
LP decoding thresholds for the BSC

The Binary Symmetric Channel (Part 1)



Let $\varepsilon \in [0,1]$. A simple model is e.g. the binary symmetric channel (BSC) with cross-over probability ε. It is a DMC

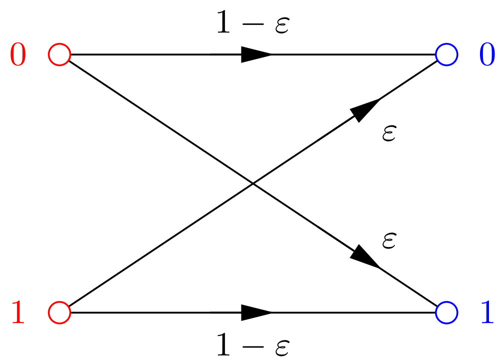
The Binary Symmetric Channel (Part 1)



Let $\varepsilon \in [0, 1]$. A simple model is e.g. the binary symmetric channel (BSC) with cross-over probability ε. It is a DMC

- with input alphabet $\mathcal{X} = \{0,\ 1\}$,

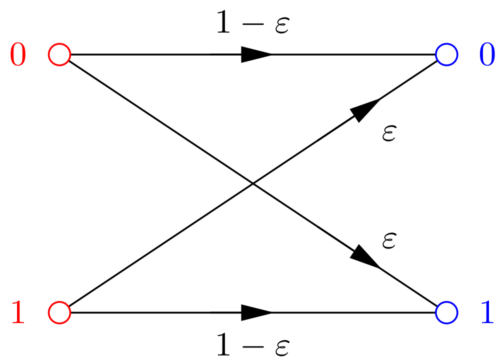
The Binary Symmetric Channel (Part 1)



Let $\varepsilon \in [0, 1]$. A simple model is e.g. the binary symmetric channel (BSC) with cross-over probability ε. It is a DMC

- with input alphabet $\mathcal{X} = \{0,\ 1\}$,
- with output alphabet $\mathcal{Y} = \{0,\ 1\}$,

The Binary Symmetric Channel (Part 1)

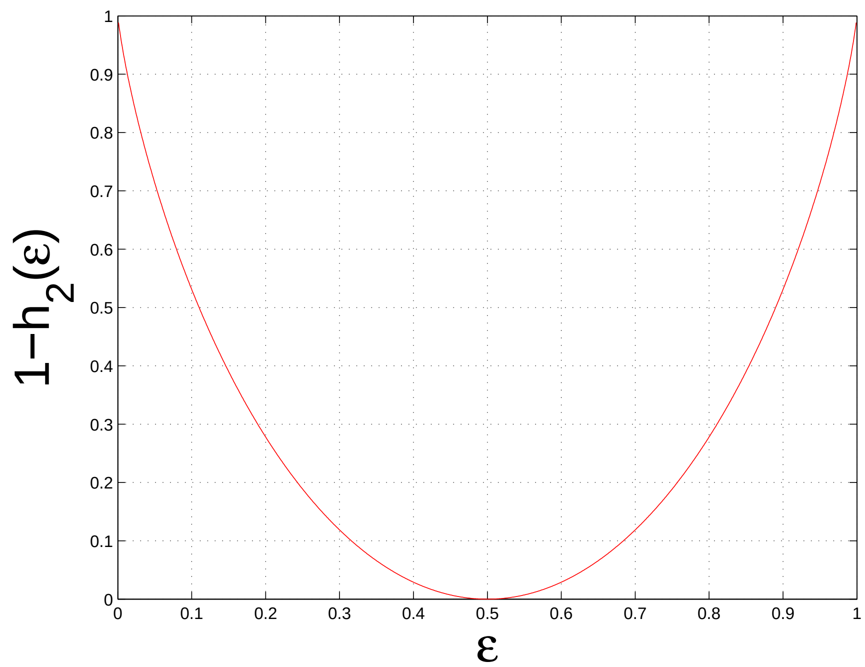


Let $\varepsilon \in [0, 1]$. A simple model is e.g. the binary symmetric channel (BSC) with cross-over probability ε. It is a DMC

- with input alphabet $\mathcal{X} = \{0,\ 1\}$,
- with output alphabet $\mathcal{Y} = \{0,\ 1\}$,
- and with conditional probability mass function

$$P_{Y_i|X_i}(y_i|x_i) = \begin{cases} 1 - \varepsilon & (y_i = x_i) \\ \varepsilon & (y_i \neq x_i) \end{cases}.$$

The Binary Symmetric Channel (Part 2)

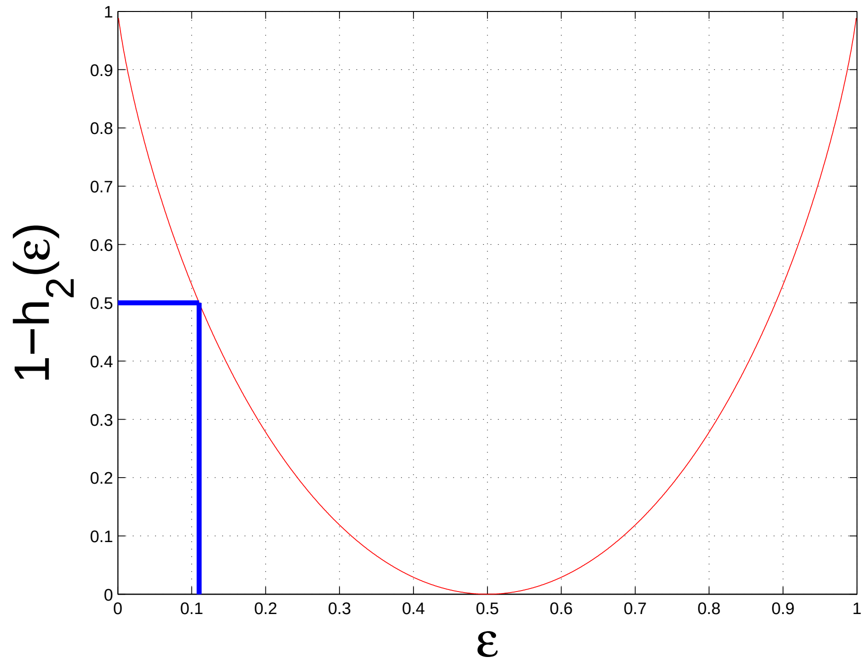


The capacity for the BSC as a function of the cross-over probability ε is

$$C_{\mathrm{BSC}} = 1 - h_2(\varepsilon),$$

where $h_2(\varepsilon) \triangleq -\varepsilon \log_2(\varepsilon) - (1-\varepsilon)\log_2(1-\varepsilon)$.

The Binary Symmetric Channel (Part 2)



The capacity for the BSC as a function of the cross-over probability ε is

$$C_{\mathrm{BSC}} = 1 - h_2(\varepsilon),$$

where $h_2(\varepsilon) \triangleq -\varepsilon \log_2(\varepsilon) - (1-\varepsilon)\log_2(1-\varepsilon)$.

The Binary Symmetric Channel (Part 3)

Assume that the channel is a BSC with cross-over probability ε.

The Binary Symmetric Channel (Part 3)

Assume that the channel is a BSC with cross-over probability ε.

Channel capacity:

- Channel coding theorem

- Converse to the channel coding theorem

The Binary Symmetric Channel (Part 3)

Assume that the channel is a BSC with cross-over probability ε.

Channel capacity:

- Channel coding theorem

  (Gallager's random coding error exponent, etc.)

- Converse to the channel coding theorem

  (Fano's inequality, etc.)

The Binary Symmetric Channel (Part 3)

Assume that the channel is a BSC with cross-over probability ε.

Channel capacity:

- Channel coding theorem
  (Gallager's random coding error exponent, etc.)
- Converse to the channel coding theorem
  (Fano's inequality, etc.)

Important: we are allowed to use the best available coding and decoding schemes for a given rate R.

The Binary Symmetric Channel (Part 4)

Assume that the channel is a BSC with cross-over probability ε.

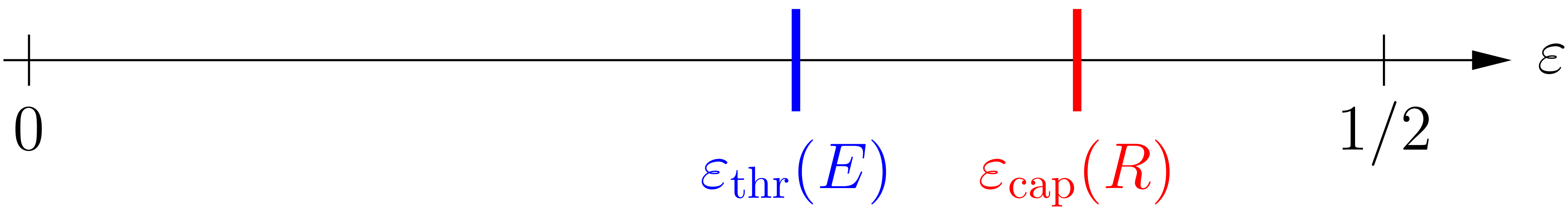
Additionally, assume that we put restrictions on the coding schemes and/or on the decoding schemes.

The Binary Symmetric Channel (Part 4)

Assume that the channel is a BSC with cross-over probability ε.

Additionally, assume that we put restrictions on the coding schemes and/or on the decoding schemes.

$\Rightarrow$ Thresholds.

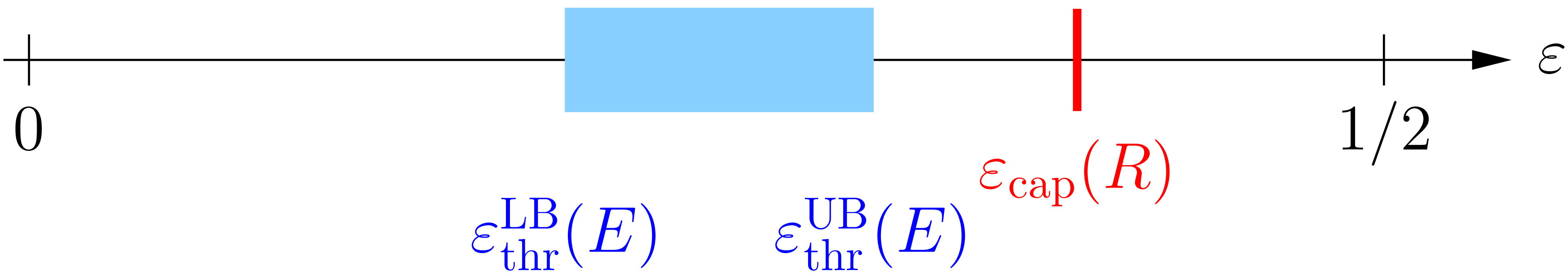


The Binary Symmetric Channel (Part 4)

Assume that the channel is a BSC with cross-over probability ε.

Additionally, assume that we put restrictions on the coding schemes and/or on the decoding schemes.

$\Rightarrow$ Thresholds.



Existence of LP Decoding Thresholds

- A priori it is not clear for what families/ensembles of codes there is an LP decoding threshold.

Existence of LP Decoding Thresholds

- A priori it is not clear for what families/ensembles of codes there is an LP decoding threshold.
- The tight connection between min-sum algorithm decoding and LP decoding suggests that families/ensembles that have a threshold under min-sum algorithm decoding also have a threshold under LP decoding.

Existence of LP Decoding Thresholds

- A priori it is not clear for what families/ensembles of codes there is an LP decoding threshold.
- The tight connection between min-sum algorithm decoding and LP decoding suggests that families/ensembles that have a threshold under min-sum algorithm decoding also have a threshold under LP decoding.
- [Koetter:Vontobel:06]: there is an LP decoding threshold for $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC codes where $2 < w_{\mathrm{col}} < w_{\mathrm{row}}$.

BSC: An Upper Bound on the Threshold (Part 1)

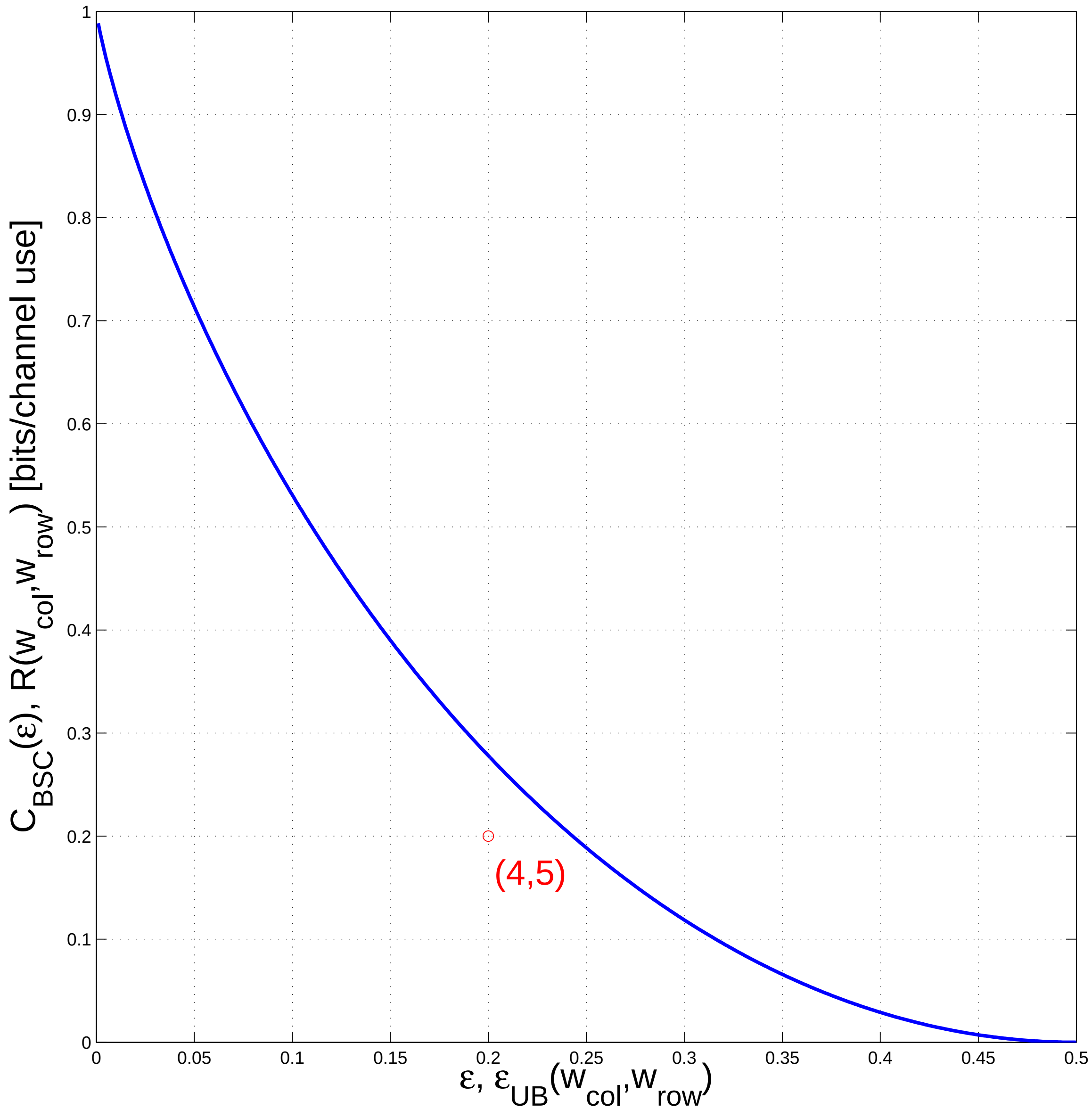
Theorem:

- Consider a family of $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular codes of increasing block length n.
- Consider a BSC with cross-over probability ε.
- In the limit $n \to \infty$, if

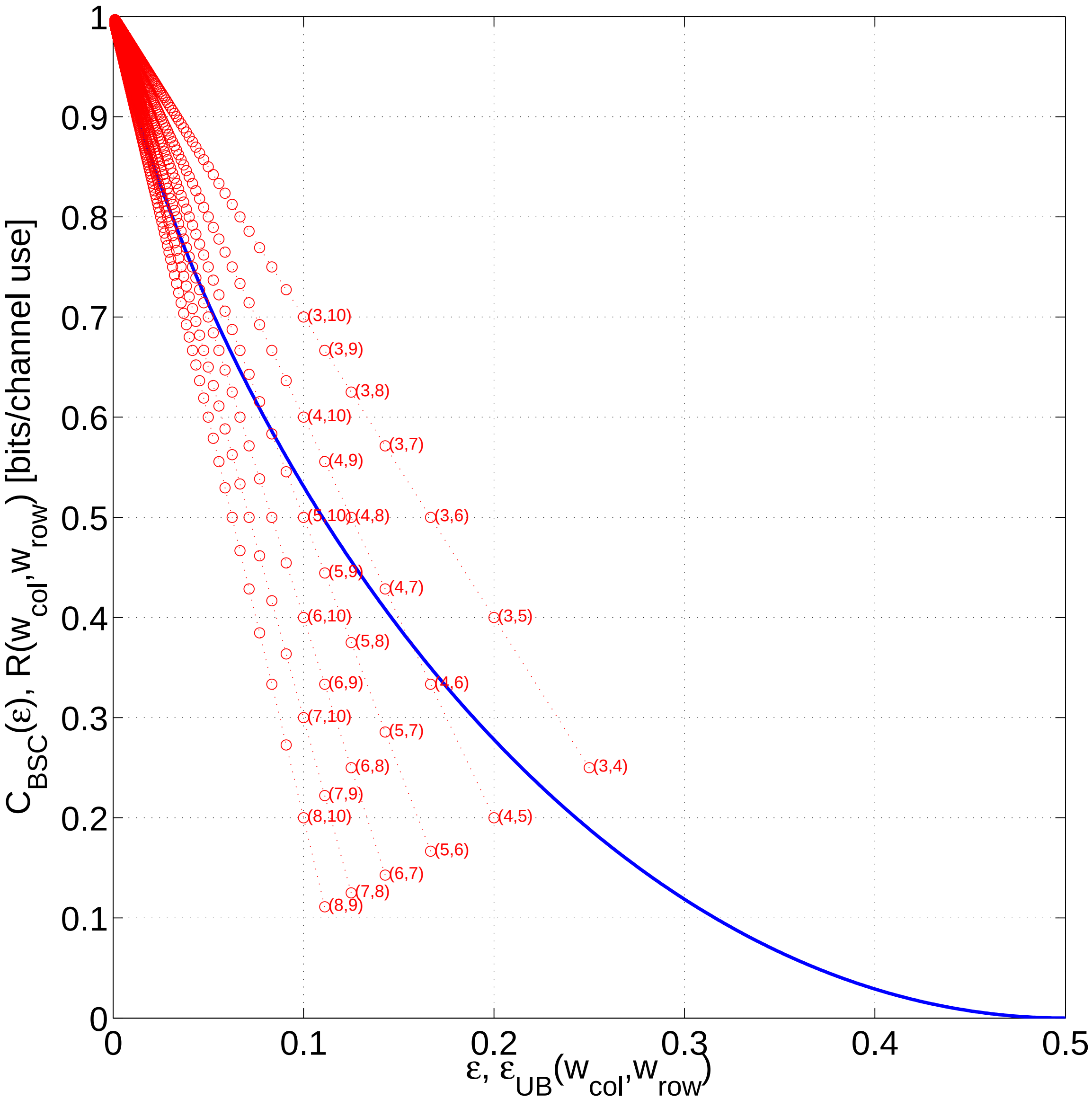
$$\varepsilon > \frac{1}{w_{\mathrm{row}}}$$

then with probability 1 the LP decoder will not decode to the transmitted codeword.

BSC: An Upper Bound on the Threshold (Part 2)



BSC: An Upper Bound on the Threshold (Part 2)



BSC: An Upper Bound on the Threshold (Part 3)

Theorem: Consider a family of codes where the minimal row-degree goes to $w_{\mathrm{row}}^{\mathrm{min}}(\infty)$ when $n \to \infty$ and a BSC with cross-over probability ε. In the limit $n \to \infty$, if

$$\varepsilon > \frac{1}{w_{\mathrm{row}}^{\mathrm{min}}(\infty)}$$

then with probability 1 the LP decoder will not decode to the transmitted codeword.

BSC: An Upper Bound on the Threshold (Part 3)

Theorem: Consider a family of codes where the minimal row-degree goes to $w_{\text{row}}^{\min}(\infty)$ when $n \to \infty$ and a BSC with cross-over probability ε. In the limit $n \to \infty$, if

$$\varepsilon > \frac{1}{w_{\text{row}}^{\min}(\infty)}$$

then with probability 1 the LP decoder will not decode to the transmitted codeword.

Corollary: For any family of codes where $w_{\text{row}}^{\min}(n)$ grows unboundedly, i.e. where

$$\lim_{n \to \infty} w_{\text{row}}^{\min}(n) = \infty,$$

the above right-hand side expression goes to 0.

Not Deciding for the All-Zeros Codeword (Part 1)

Linear programming (LP) decoding:

$$\hat{\boldsymbol{\omega}} = \arg \min_{\boldsymbol{\omega} \in \mathcal{P}(\mathbf{H})} \sum_{i=1}^{n} \lambda_i \omega_i .$$

Not Deciding for the All-Zeros Codeword (Part 1)

Linear programming (LP) decoding:

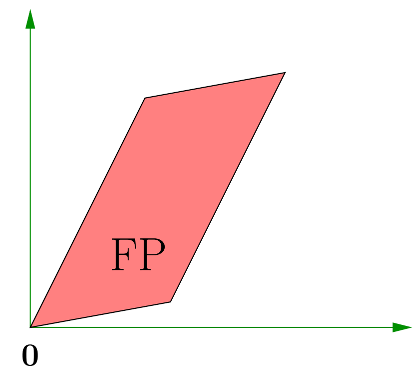
$$\hat{\boldsymbol{\omega}} = \arg \min_{\boldsymbol{\omega} \in \mathcal{P}(\mathbf{H})} \sum_{i=1}^{n} \lambda_i \omega_i .$$

Assume that the zero codeword has been sent. LP decoding does not decide for the all-zeros codeword if there is a vector

$$\boldsymbol{\omega} \in \mathcal{P}(\mathbf{H}) \setminus \{\mathbf{0}\}$$

such that

$$\sum_{i=1}^{n} \lambda_i \omega_i < 0.$$



Not Deciding for the All-Zeros Codeword (Part 2)

Linear programming (LP) decoding:

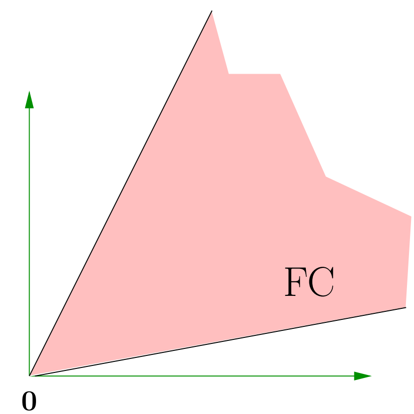
$$\hat{\boldsymbol{\omega}} = \arg \min_{\boldsymbol{\omega} \in \mathcal{P}(\mathbf{H})} \sum_{i=1}^{n} \lambda_i \omega_i .$$

Assume that the zero codeword has been sent. LP decoding does not decides for the all-zeros codeword if there is a vector

$$\boldsymbol{\omega} \in \mathcal{K}(\mathbf{H}) \setminus \{\mathbf{0}\}$$

such that

$$\sum_{i=1}^{n} \lambda_i \omega_i < 0.$$



Not Deciding for the All-Zeros Codeword (Part 3)

- Assume that we have a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC code.

Not Deciding for the All-Zeros Codeword (Part 3)

- Assume that we have a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC code.
- Moreover, let $\boldsymbol{\omega} \in \mathbb{R}^n$ be a vector with the following entries:

$$\omega_i \triangleq \begin{cases} \frac{1}{w_{\mathrm{row}}-1} & \text{if } \lambda_i \geq 0 \\ 1 & \text{if } \lambda_i < 0 \end{cases}.$$

Not Deciding for the All-Zeros Codeword (Part 3)

- Assume that we have a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC code.
- Moreover, let $\boldsymbol{\omega} \in \mathbb{R}^n$ be a vector with the following entries:

$$\omega_i \triangleq \begin{cases} \frac{1}{w_{\mathrm{row}}-1} & \text{if } \lambda_i \geq 0 \\ 1 & \text{if } \lambda_i < 0 \end{cases}.$$

One can easily verify that $\boldsymbol{\omega} \in \mathcal{K}(\mathbf{H})$.

Not Deciding for the All-Zeros Codeword (Part 3)

- Assume that we have a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC code.
- Moreover, let $\boldsymbol{\omega} \in \mathbb{R}^n$ be a vector with the following entries:

$$\omega_i \triangleq \begin{cases} \frac{1}{w_{\mathrm{row}}-1} & \text{if } \lambda_i \geq 0 \\ 1 & \text{if } \lambda_i < 0 \end{cases}.$$

  One can easily verify that $\boldsymbol{\omega} \in \mathcal{K}(\mathbf{H})$.

- So, if

$$0 > \sum_{i=1}^{n} \lambda_i \omega_i = \left(\sum_{\substack{i=1 \\ \lambda_i \geq 0}}^{n} \lambda_i \right) \cdot \frac{1}{w_{\mathrm{row}} - 1} + \left(\sum_{\substack{i=1 \\ \lambda_i < 0}}^{n} \lambda_i \right) \cdot 1$$

  then LP decoding does not decide for the all-zeros codeword.

Not Deciding for the All-Zeros Codeword: BSC (Part1)

- For simplicity, assume that we are transmitting over a BSC with crossover probability $0 \leq \varepsilon < 1/2$.

$$\Rightarrow \quad \lambda_i \in \{\pm L\} \quad \text{where} \quad L \triangleq \log\left(\frac{1-\varepsilon}{\varepsilon}\right) > 0.$$

Not Deciding for the All-Zeros Codeword: BSC (Part1)

- For simplicity, assume that we are transmitting over a BSC with crossover probability $0 \leq \varepsilon < 1/2$.

$$\Rightarrow \quad \lambda_i \in \{\pm L\} \quad \text{where} \quad L \triangleq \log\left(\frac{1-\varepsilon}{\varepsilon}\right) > 0.$$

- So, if

$$0 > \sum_{i=1}^{n} \lambda_i \omega_i = L \cdot \left((\#\text{not flipped}) \frac{1}{w_{\text{row}} - 1} - (\#\text{flipped}) \right).$$

then LP decoding does not decide for the all-zeros codeword.

Not Deciding for the All-Zeros Codeword: BSC (Part 2)

So, if

$$0 > \sum_{i=1}^{n} \lambda_i \omega_i = L \cdot \left((\#\text{not flipped}) \frac{1}{w_{\mathrm{row}} - 1} - (\#\text{flipped}) \right).$$

then LP decoding does not decide for the all-zeros codeword.

Not Deciding for the All-Zeros Codeword: BSC (Part 2)

So, if

$$0 > \sum_{i=1}^{n} \lambda_i \omega_i = L \cdot \left((\#\text{not flipped}) \frac{1}{w_{\text{row}} - 1} - (\#\text{flipped}) \right).$$

then LP decoding does not decide for the all-zeros codeword.

- Upon normalization, the above condition reads

$$0 > \frac{1}{n} \sum_{i=1}^{n} \lambda_i \omega_i = L \cdot \left(\frac{(\#\text{not flipped})}{n} \frac{1}{w_{\text{row}} - 1} - \frac{(\#\text{flipped})}{n} \right).$$

LABS hp

Not Deciding for the All-Zeros Codeword: BSC (Part 2)

So, if

$$0 > \sum_{i=1}^{n} \lambda_i \omega_i = L \cdot \left((\#\text{not flipped}) \frac{1}{w_{\text{row}} - 1} - (\#\text{flipped}) \right).$$

then LP decoding does not decide for the all-zeros codeword.

- Upon normalization, the above condition reads

$$0 > \frac{1}{n} \sum_{i=1}^{n} \lambda_i \omega_i = L \cdot \left(\frac{(\#\text{not flipped})}{n} \frac{1}{w_{\text{row}} - 1} - \frac{(\#\text{flipped})}{n} \right).$$

- In the limit $n \to \infty$, the above condition is with probability one equal to the condition

$$0 > \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \lambda_i \omega_i = L \cdot \left((1 - \varepsilon) \frac{1}{w_{\text{row}} - 1} - \varepsilon \right).$$

BSC: An Upper Bound on the Threshold (Part 1)

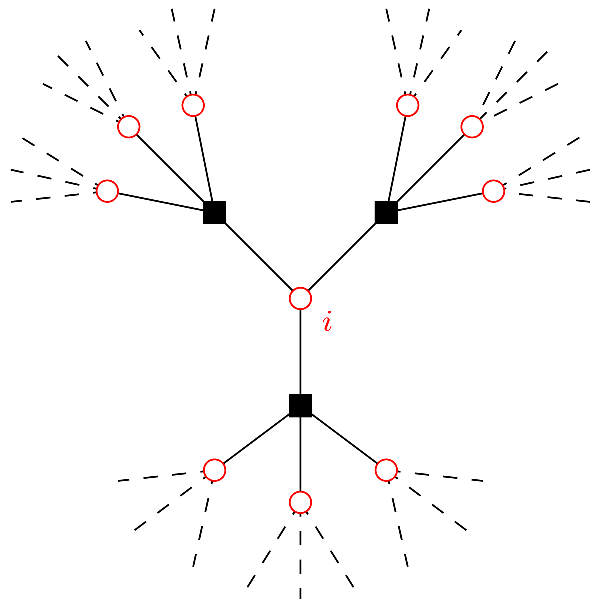
Theorem:

- Consider a family of $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular codes of increasing block length n.
- Consider a BSC with cross-over probability ε.
- In the limit $n \to \infty$, if

$$\varepsilon > \frac{1}{w_{\mathrm{row}}}$$

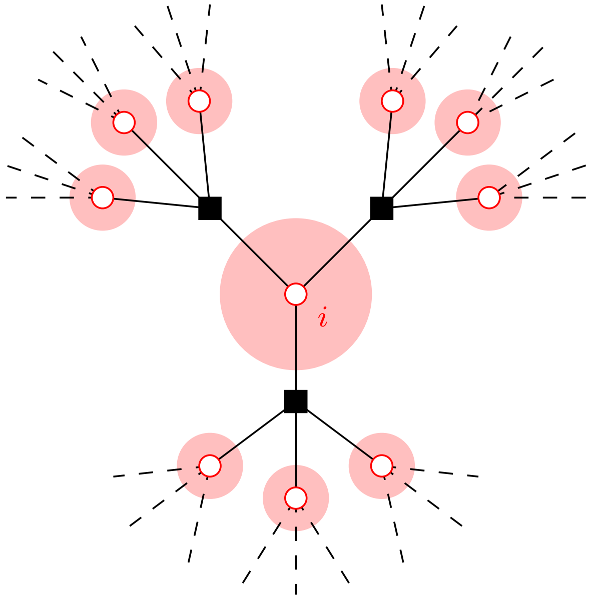
then with probability 1 the LP decoder will not decode to the transmitted codeword.

0-Neighborhood-Based Bounds (Part 1)



$\boldsymbol{\omega}$-vector that we constructed before: note that the the assignment of a value to ω_i was based only on the value of λ_i.

0-Neighborhood-Based Bounds (Part 2)



$\boldsymbol{\omega}$-vector that we constructed before: note that the the assignment of a value to $\boldsymbol{\omega}_i$ was based only on the value of λ_i:

$$\omega_i = f(\lambda_i) = f\left(\{\lambda_{i'}\}_{i' \in \mathcal{N}_i^{(0)}}\right).$$

0-Neighborhood-Based Bounds (Part 3)

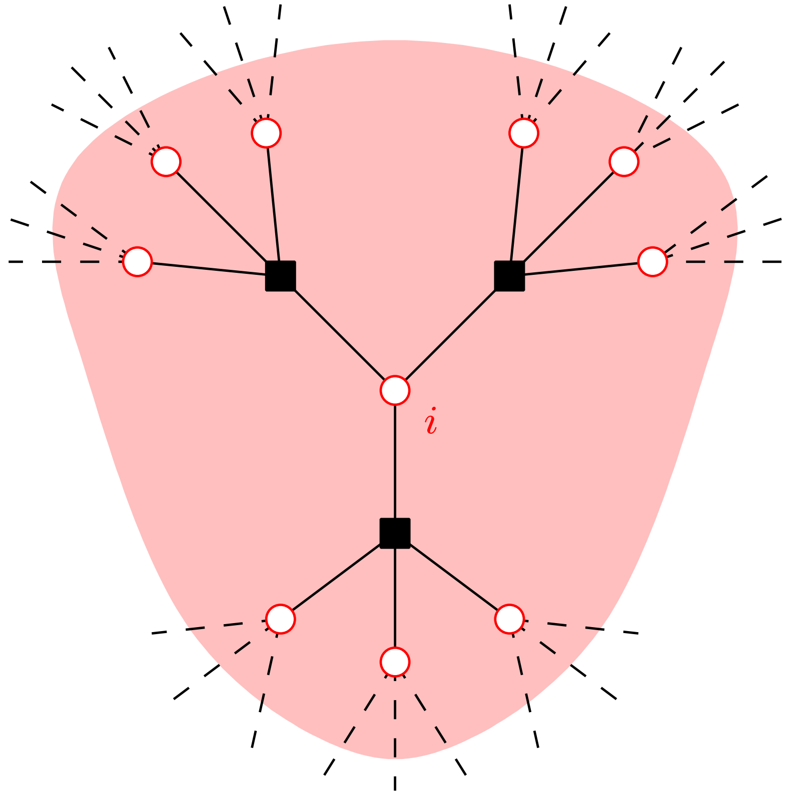
$\boldsymbol{\omega}$-vector that we constructed before: note that the the assignment of a value to $\boldsymbol{\omega}_i$ was based only on the value of λ_i:

$$\omega_i = f(\lambda_i) = f\left(\{\lambda_{i'}\}_{i' \in \mathcal{N}_i^{(0)}}\right).$$

$$\omega_i \triangleq \begin{cases} \frac{1}{w_{\mathrm{row}}-1} & \text{if } \lambda_i \geq 0 \\ 1 & \text{if } \lambda_i < 0 \end{cases}.$$

One can easily check that $\boldsymbol{\omega} \in \mathcal{K}(\mathbf{H})$.

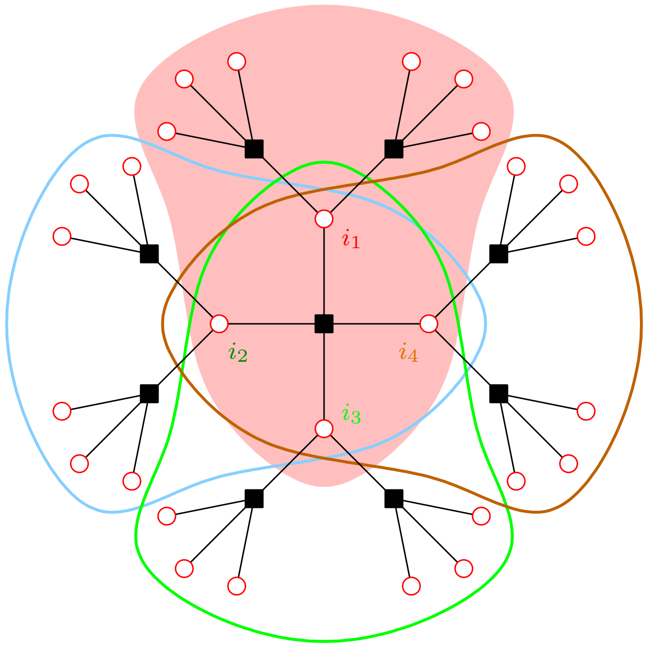
2-Neighborhood-Based Bounds on the Threshold



Generalization:

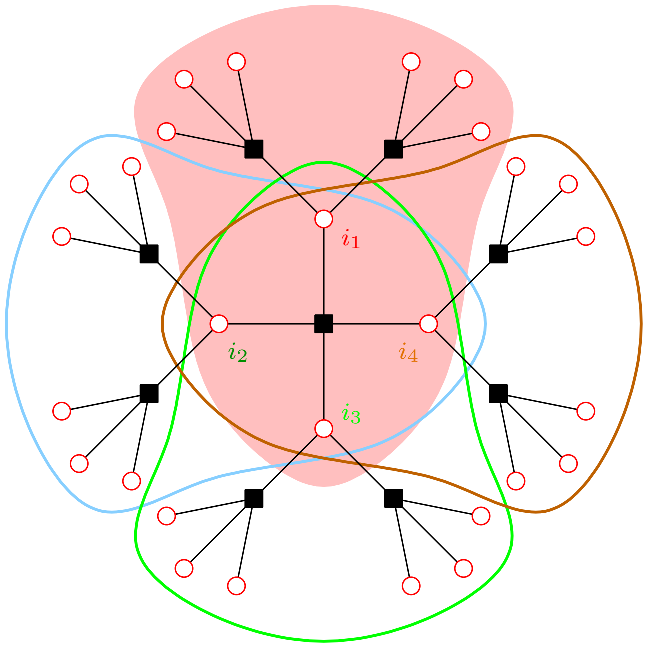
$$\omega_i = f\left(\{\lambda_{i'}\}_{i' \in \mathcal{N}_i^{(2)}}\right).$$

2-Neighborhood-Based Bounds on the Threshold



We must take care of constrains: the map $f\left(\{\lambda_{i'}\}_{i' \in \mathcal{N}_i^{(2)}}\right)$ has to yield a vector in $\mathcal{K}(\mathbf{H})$.

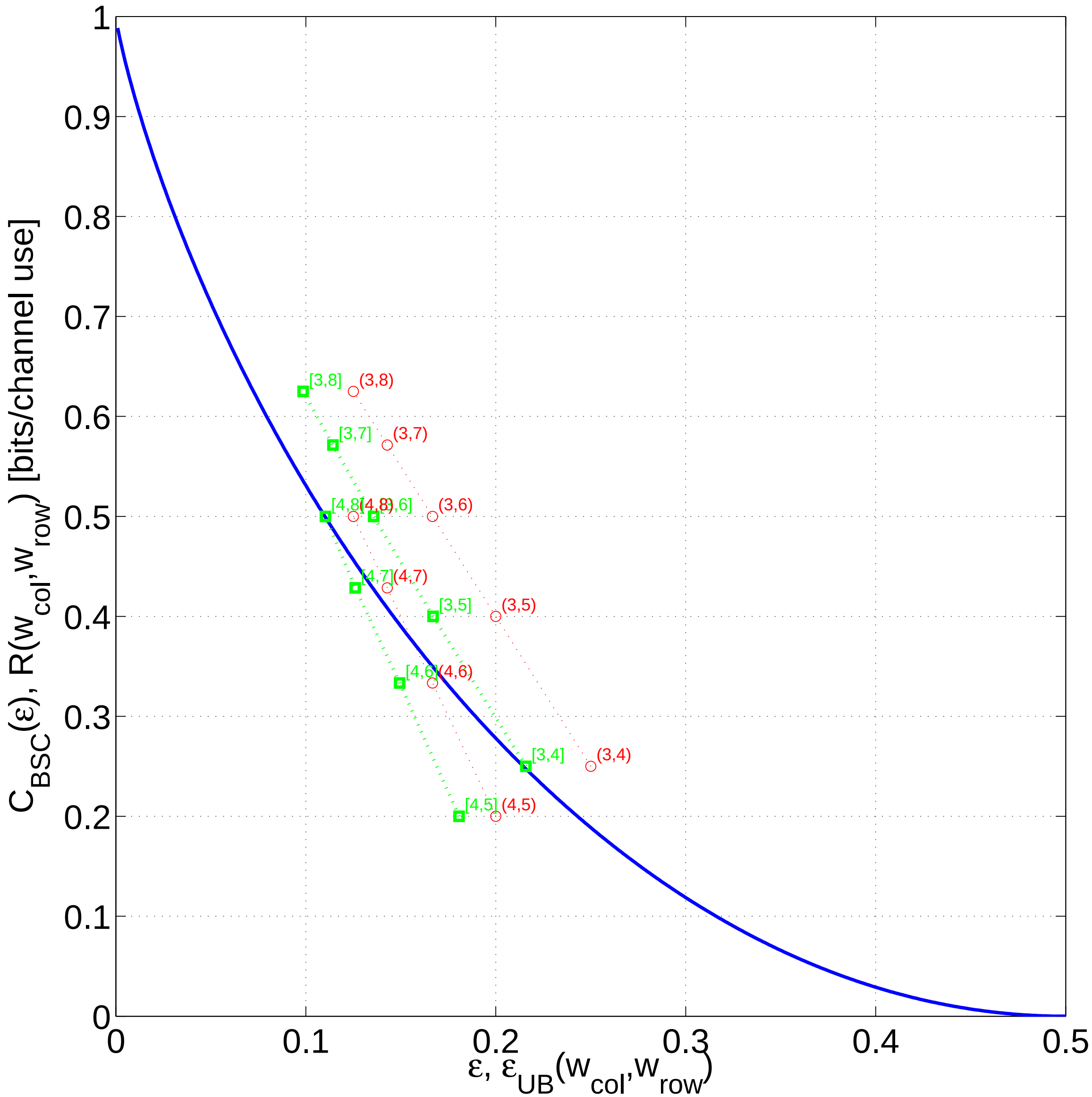
2-Neighborhood-Based Bounds on the Threshold



We must take care of constrains: the map $f\left(\{\lambda_{i'}\}_{i' \in \mathcal{N}_i^{(2)}}\right)$ has to yield a vector in $\mathcal{K}(\mathbf{H})$.

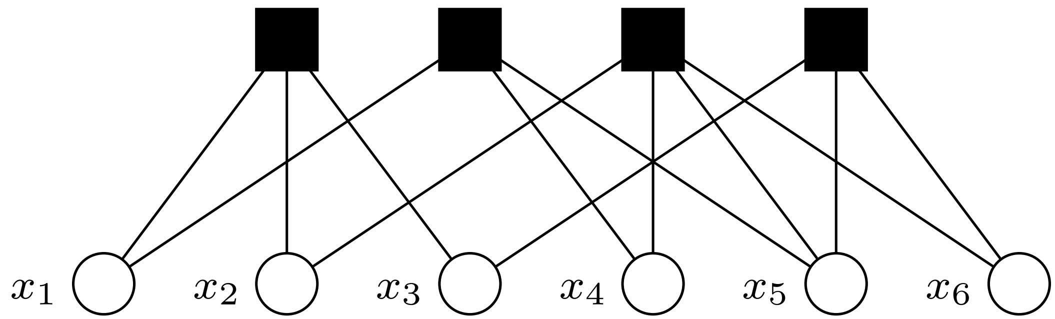
$\Rightarrow$ We can set up a linear program that yields the best possible threshold for a 2-neighborhood. (Graph automorphisms help in simplifying that LP.)

2-Neighborhood-Based Bounds on the Threshold



Stopping sets, near-codewords, . . .

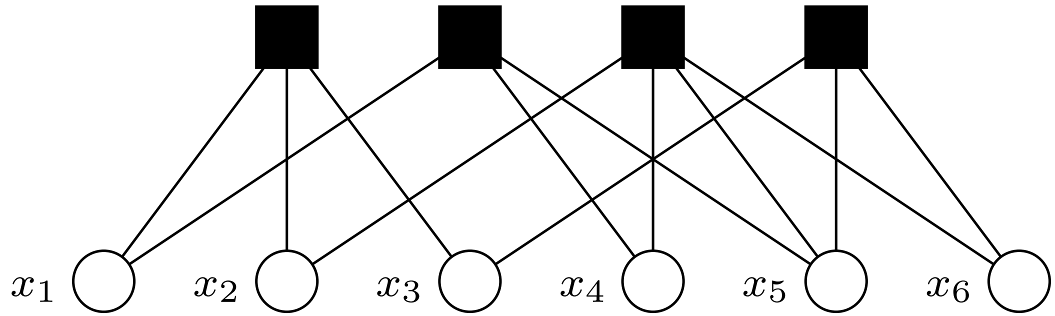
Stopping Sets (Part 1)



received values:

after first iteration:

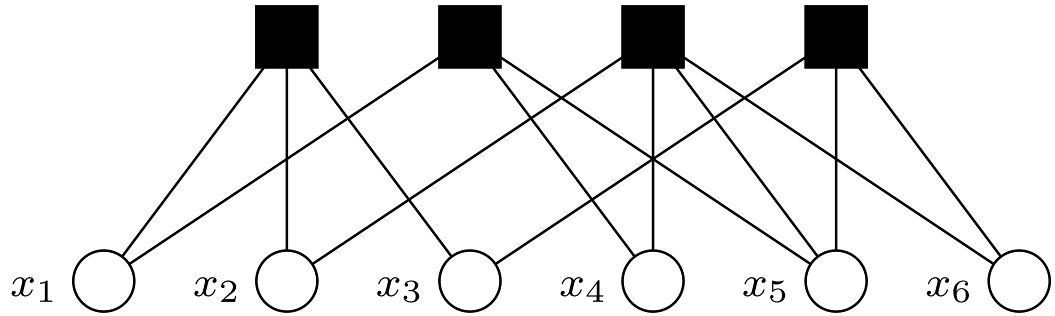
Stopping Sets (Part 1)



received values: **0** **?** **0** **?** **?** **?**

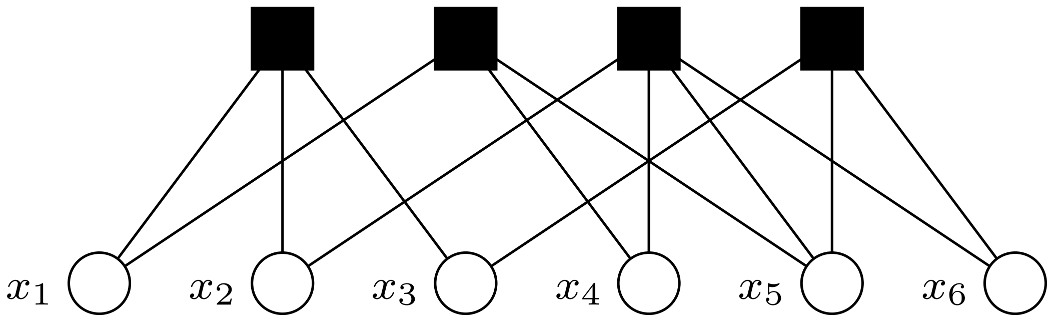
after first iteration:

Stopping Sets (Part 1)



received values:	**0**	**?**	**0**	**?**	**?**	**?**
after first iteration:	**0**	**0**	**0**	**?**	**?**	**?**

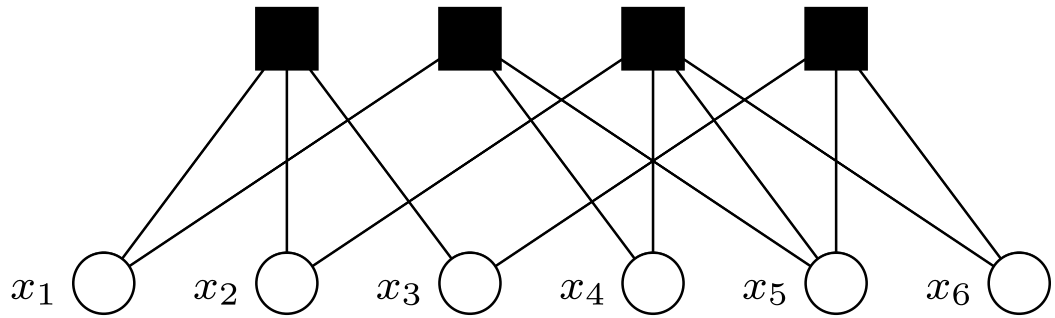
Stopping Sets (Part 1)



received values:	**0**	?	**0**	?	?	?
after first iteration:	**0**	**0**	**0**	?	?	?

stuck!

Stopping Sets (Part 1)

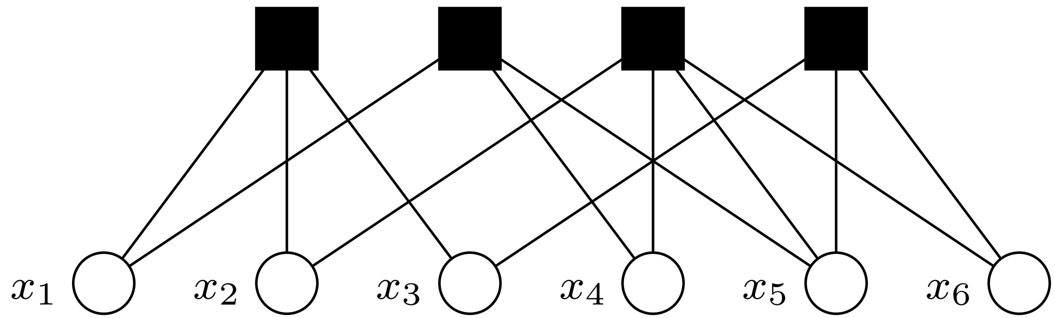


received values:	**0**	**?**	**0**	**?**	**?**	**?**
after first iteration:	**0**	**0**	**0**	**?**	**?**	**?**

stuck!

Stopping set: $\mathcal{S} = \{x_4, x_5, x_6\}$.

Stopping Sets (Part 1)



received values:	**0**	**?**	**0**	**?**	**?**	**?**
after first iteration:	**0**	**0**	**0**	**?**	**?**	**?**

stuck!

Stopping set: $\mathcal{S} = \{x_4, x_5, x_6\}$.

The log-likelihood ratio vector for the above example is $\boldsymbol{\lambda} = (+\infty, 0, +\infty, 0, 0, 0)$. Note that under LP decoding the vector $(0, 0, 0, 0, 0, 0)$ (which is a codeword) and the vector $(0, 0, 0, \frac{2}{3}, \frac{2}{3}, \frac{2}{3})$ (which is a pseudo-codeword) have equal cost, i.e. cost zero.

Stopping Sets (Part 2)

Theorem:

Stopping Sets (Part 2)

Theorem:

- The support of any pseudo-codeword is a stopping set.

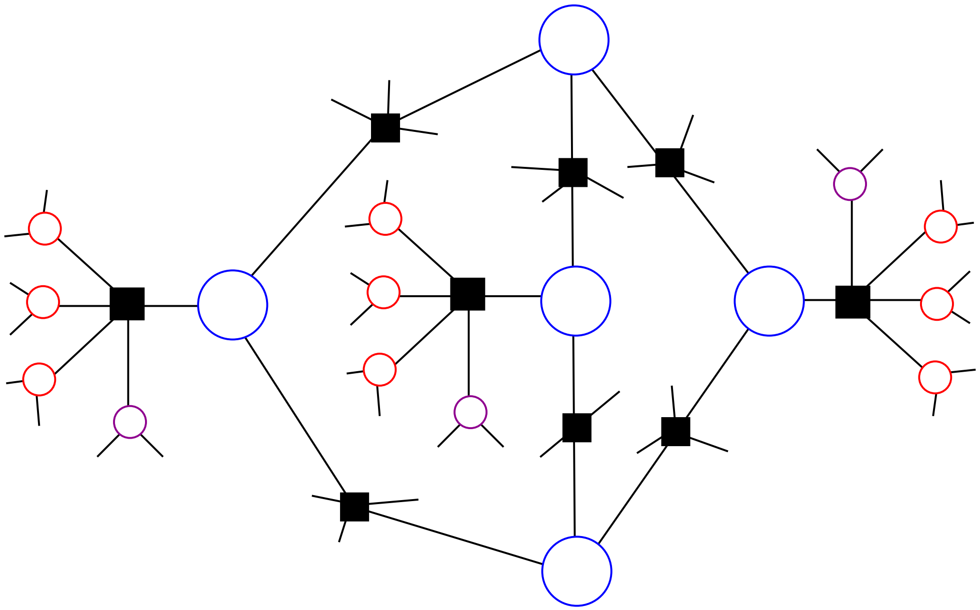
Stopping Sets (Part 2)

Theorem:

- The support of any pseudo-codeword is a stopping set.
- For any stopping set there exists at least one pseudo-codeword such that its support equals that stopping set.

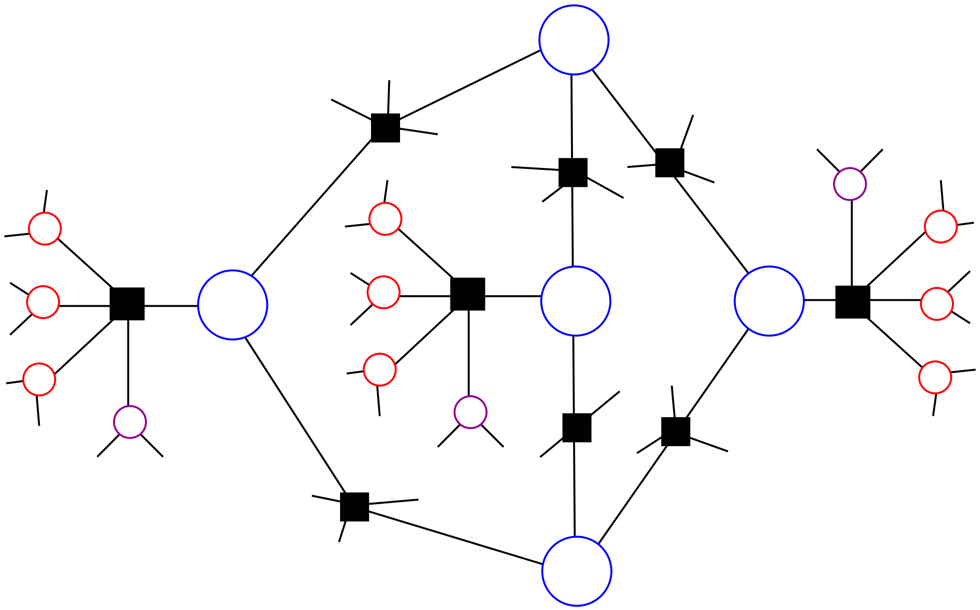
Near-Codewords (Part 1)

Example: a $[155, 64, 20]$ binary linear code by Tanner.



Near-Codewords (Part 1)

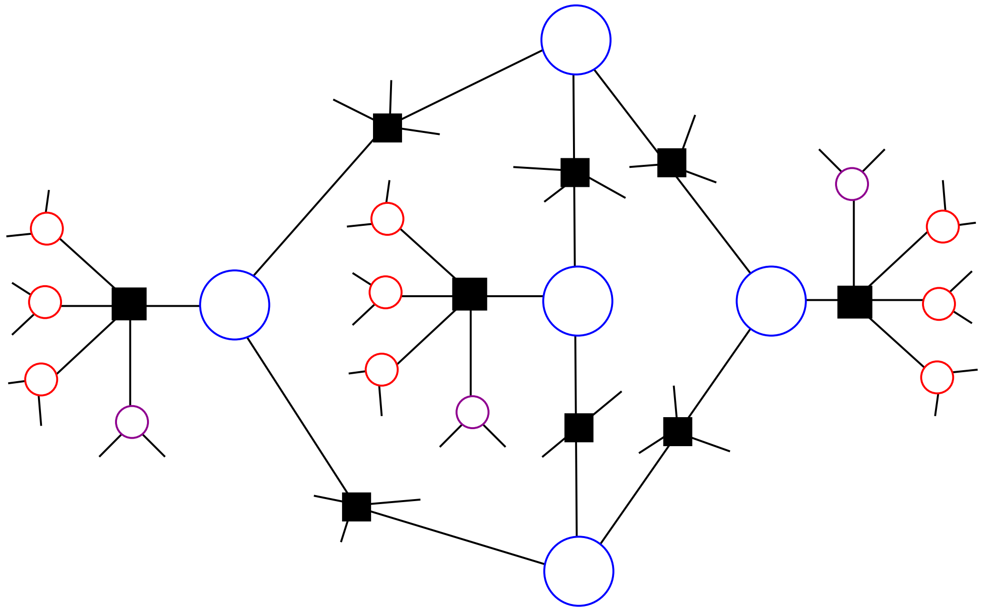
Example: a $[155, 64, 20]$ binary linear code by Tanner.



The blue vertices form a so-called $(5, 3)$ near-codeword.

Near-Codewords (Part 2)

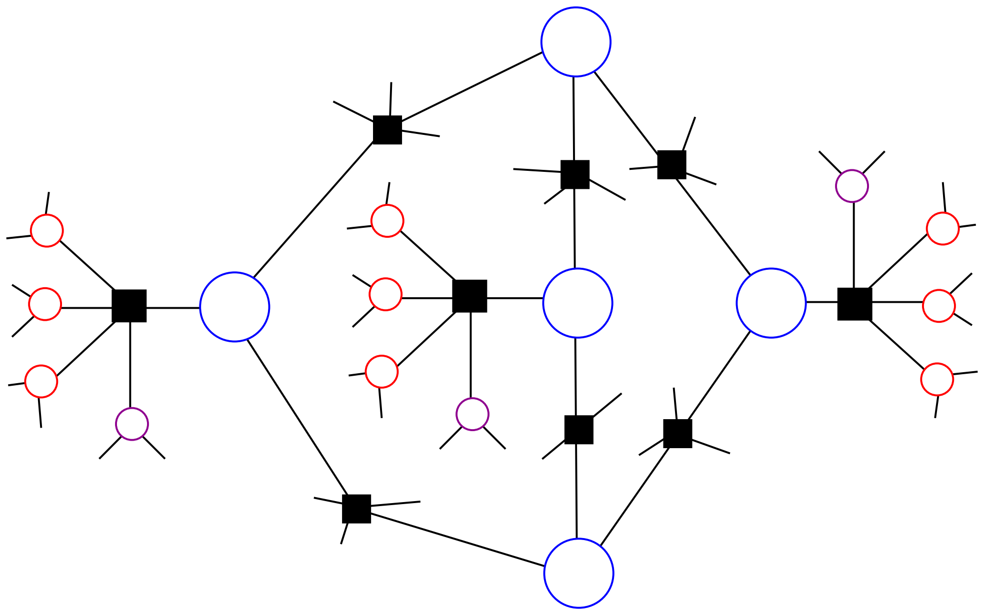
Example: a $[155, 64, 20]$ binary linear code by Tanner.



Heuristic why near-codewords are bad for MPI decoding: the canonical completion w.r.t. the set of blue vertices gives a pseudo-codeword which is "bad" itself or is a good starting point for searching "bad" pseudo-codewords in the fundamental cone.

Near-Codewords (Part 2)

Example: a $[155, 64, 20]$ binary linear code by Tanner.



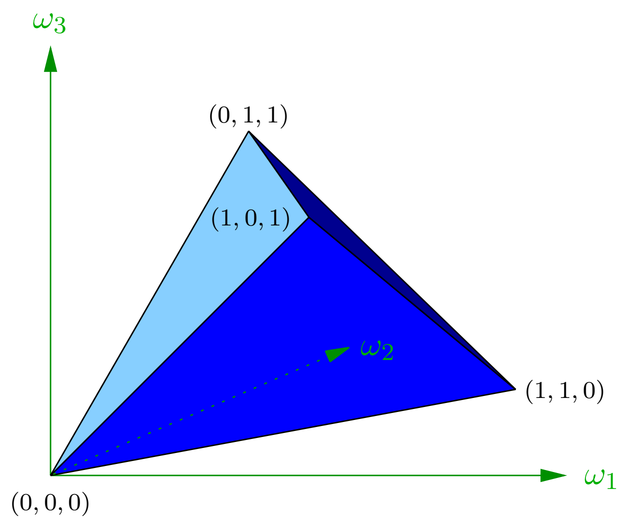
Heuristic why near-codewords are bad for MPI decoding: the canonical completion w.r.t. the set of blue vertices gives a pseudo-codeword which is "bad" itself or is a good starting point for searching "bad" pseudo-codewords in the fundamental cone.

Closely related notions: trapping sets, absorption sets.

The fundamental polytope

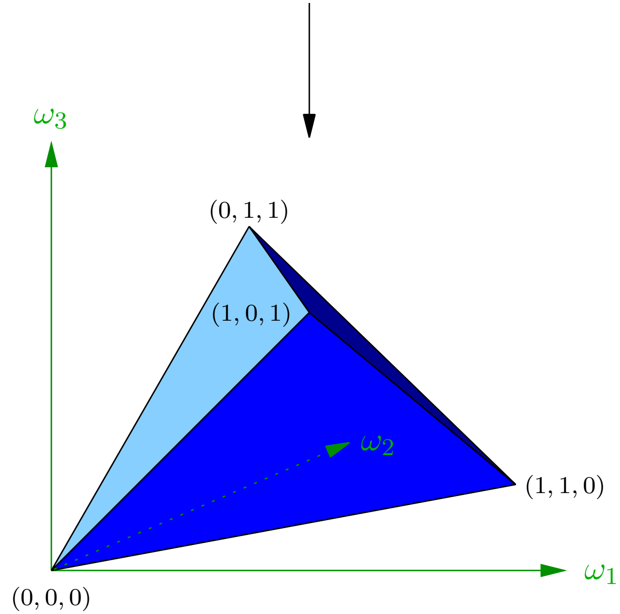
in various contexts

The Fundamental Polytope in Various Contexts



The Fundamental Polytope in Various Contexts

Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)

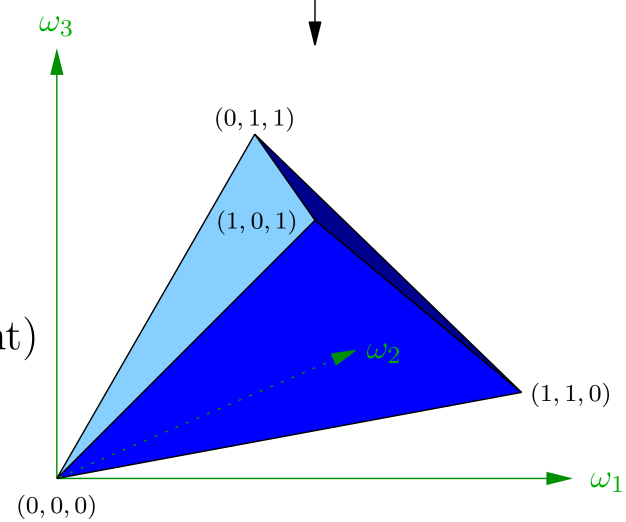


The Fundamental Polytope in Various Contexts

Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)

LP Decoding

(Feldman/Karger/Wainwright)
(CISS 2003)

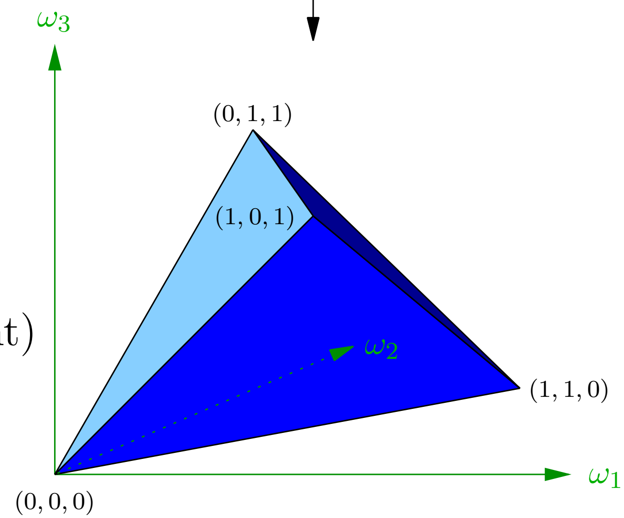


The Fundamental Polytope in Various Contexts

Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)

LP Decoding

(Feldman/Karger/Wainwright)
(CISS 2003)



Bethe free energy

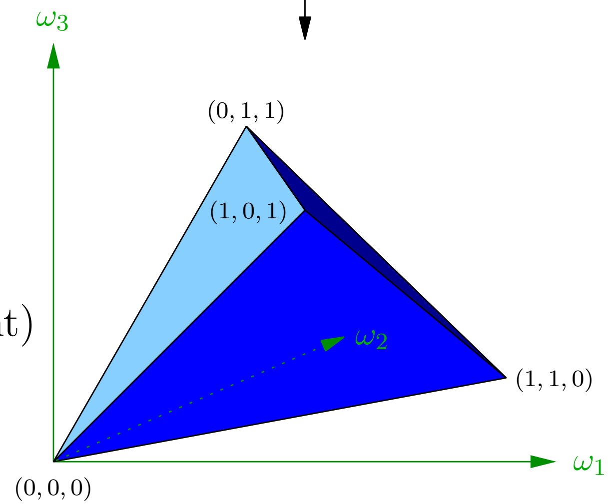
(Yedidia/Freeman/Weiss)
(MERL Rep. 2001)

The Fundamental Polytope in Various Contexts

Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)

LP Decoding

(Feldman/Karger/Wainwright)
(CISS 2003)



Bethe free energy

(Yedidia/Freeman/Weiss)
(MERL Rep. 2001)

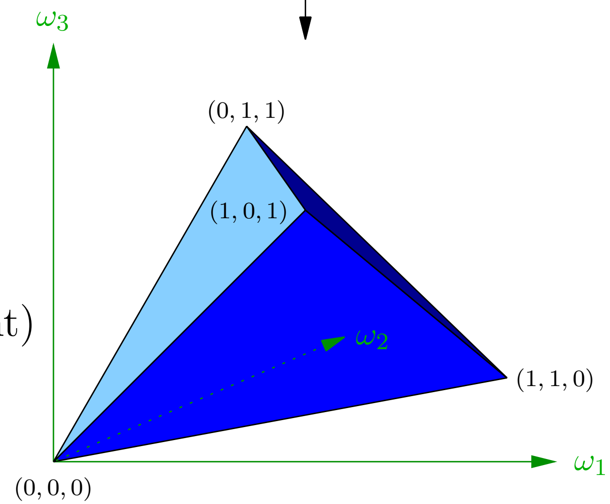
Fundamental cone of cycle codes is the Newton polyhedron
of the edge zeta function of normal factor graph
(Koetter/Li/Vontobel/Walker, ITW2004)

The Fundamental Polytope in Various Contexts

Finite-length analysis of iterative decoding based on graph covers
(Koetter/Vontobel, Turbo Conf. 2003)

LP Decoding

(Feldman/Karger/Wainwright)
(CISS 2003)



Bethe free energy

(Yedidia/Freeman/Weiss)
(MERL Rep. 2001)

"The Tropical Geometry of Statistical Models"
Sturmfels, plenary talk at Allerton 2004

Fundamental cone of cycle codes is the Newton polyhedron
of the edge zeta function of normal factor graph
(Koetter/Li/Vontobel/Walker, ITW2004)

LABS hp