Karlsruhe Institute of Technology

Application of Optimization Algorithms for Channel Decoding

21.06.2024

Andreas Tsouchlos

Overview

- Theoretical Background
- LP Decoding using ADMM
  - LP Decoding
  - Application of ADMM
  - Simulation Results
- Proximal Decoding
  - Decoding Algorithm
  - Simulation Results
  - Improved Algorithm

Overview

- Theoretical Background
- LP Decoding using ADMM
  - LP Decoding
  - Application of ADMM
  - Simulation Results
- Proximal Decoding
  - Decoding Algorithm
  - Simulation Results
  - Improved Algorithm

Motivation

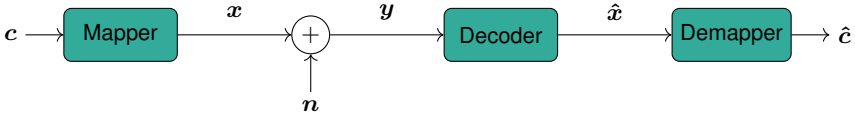
- General ML decoding problem NP-complete [BMT78]
- Iterative message–passing algorithms popular in practice do not guarantee optimality when the graph contains cycles [KTP19]
- Standard message-passing algorithms difficult to analyze [FEL03]

[BMT78] E. Berlekamp; R. McEliece; H. van Tilborg: *On the inherent intractability of certain coding problems (Corresp.).* IEEE Transactions on Information Theory 24.3 (May 1978), pp. 384–386.

[KTP19] Banu Kabakulak; Z. Caner Taşkın; Ali Emre Pusane: *Optimization–based decoding algorithms for LDPC convolutional codes in communication systems.* IISE Transactions 51.10 (2019), pp. 1061–1074.

[FEL03] Jon Feldman: *Decoding error-correcting codes via linear programming.* PhD thesis. MIT, 2003.

Presumptions: Channel & Modulation



- All simulations are performed with BPSK:

$$\boldsymbol{x} = (-1)^{\boldsymbol{c}}, \qquad \boldsymbol{c} \in \mathbb{F}_2^n, \;\; \boldsymbol{x} \in \{-1, 1\}^n$$

- The channel model is AWGN:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}, \qquad \boldsymbol{n} \sim \mathcal{N}\left(0, \frac{1}{2}\left(\frac{k}{n}\frac{E_b}{N_0}\right)^{-1}\right), \;\; \boldsymbol{y}, \boldsymbol{n} \in \mathbb{R}^n$$

- All-zeros assumption:

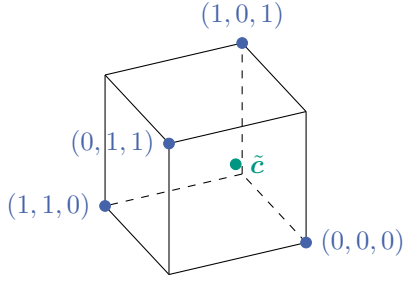
$$\boldsymbol{c} = \boldsymbol{0}$$

Optimization as a Decoding Method

- Reformulate decoding problem as optimization problem
  - Establish objective function
  - Establish constraints
- Use optimization method to solve the new problem

- Usage of "$\sim$" to denote change in domain, e.g.:

$$\boldsymbol{c} \in \{0,1\}^n \rightarrow \tilde{\boldsymbol{c}} \in [0,1]^n$$
$$\boldsymbol{x} \in \{-1,1\}^n \rightarrow \tilde{\boldsymbol{x}} \in \mathbb{R}^n$$



Overview

- Theoretical Background
- LP Decoding using ADMM
  - LP Decoding
  - Application of ADMM
  - Simulation Results
- Proximal Decoding
  - Decoding Algorithm
  - Simulation Results
  - Improved Algorithm

LP Decoding [FWK05]

- General reformulation of ML decoding as a linear program (LP)
- Cost function:

$$\boldsymbol{\gamma}^T \boldsymbol{c} = \sum_{i=1}^{n} \gamma_i c_i, \qquad \gamma_i = \ln\left(\frac{f_{Y_i|C_i}\left(y_i|c_i=0\right)}{f_{Y_i|C_i}\left(y_i|c_i=1\right)}\right)$$

- Exact *integer linear programming* (ILP) formulation of ML decoding:

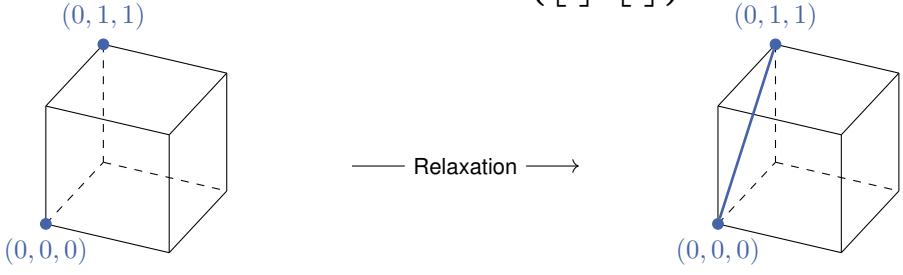
$$\begin{aligned} &\text{minimize } \boldsymbol{\gamma}^{\mathsf{T}} \boldsymbol{c} \\ &\text{subject to } \boldsymbol{c} \in \mathcal{C} \end{aligned}$$

- Goal: Relaxation of constraints to make a practical solution to the problem feasible

[FWK05] J. Feldman; M.J. Wainwright; D.R. Karger: *Using linear programming to Decode Binary linear codes.* IEEE Transactions on Information Theory 51.3 (2005), pp. 954–972.

LP Decoding: Relaxation

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad \mathcal{C} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right\}$$

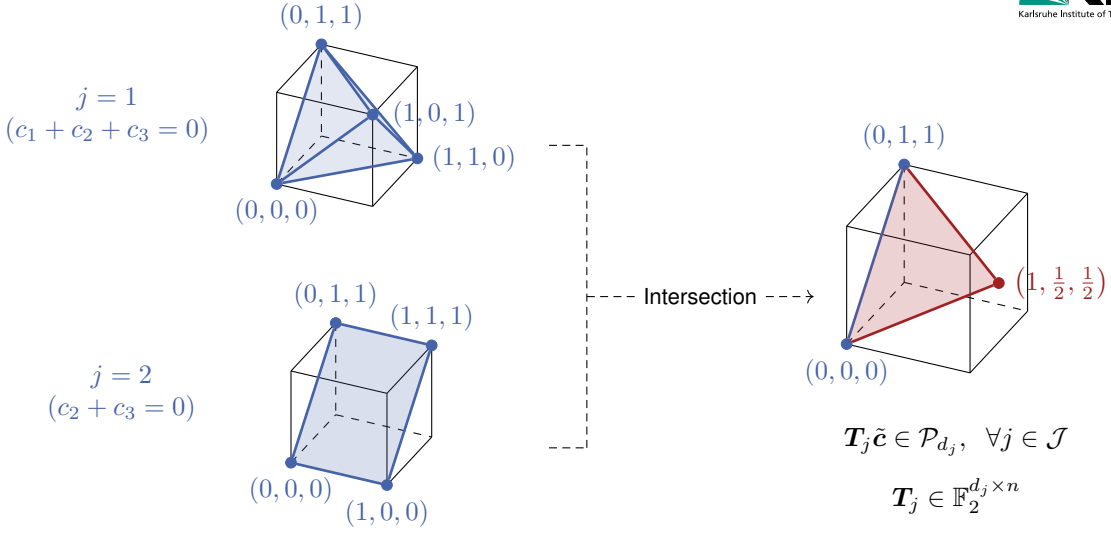


(a) Set of all codewords $\mathcal{C}$

(b) Codeword polytope poly $(\mathcal{C})$

$$\text{poly}\,(\mathcal{C}) = \left\{ \sum_{\boldsymbol{c} \in \mathcal{C}} \alpha_{\boldsymbol{c}} \boldsymbol{c} : \alpha_{\boldsymbol{c}} \geq 0, \sum_{\boldsymbol{c} \in \mathcal{C}} \alpha_{\boldsymbol{c}} = 1 \right\}, \qquad \alpha_{\boldsymbol{c}} \in \mathbb{R}_{\geq 0}$$

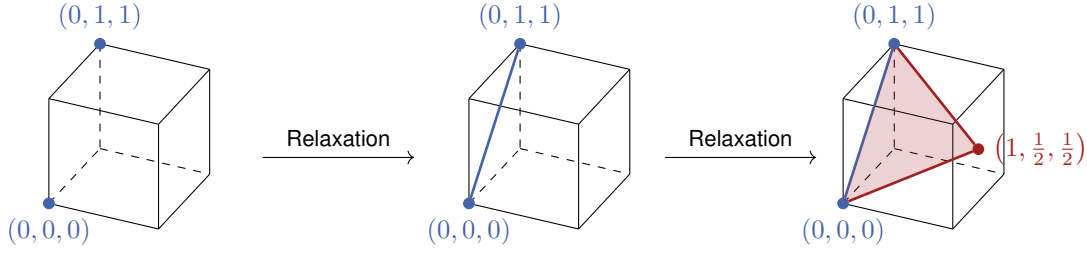
LP Decoding: Relaxation



(a) Local codeword polytopes $\mathcal{P}_{d_j}$ of the check nodes

(b) Relaxed codeword polytope $\overline{Q}$

LP Decoding: Relaxation



(a) *Integer linear program* (ILP)

$$\begin{aligned} &\text{minimize} && \boldsymbol{\gamma}^{\mathsf{T}}\boldsymbol{c} \\ &\text{subject to} && \boldsymbol{c} \in \mathcal{C} \end{aligned}$$

(b) Motivation

$$\begin{aligned} &\text{minimize} && \boldsymbol{\gamma}^{\mathsf{T}}\tilde{\boldsymbol{c}} \\ &\text{subject to} && \tilde{\boldsymbol{c}} \in \text{poly}\,(\mathcal{C}) \end{aligned}$$

(c) *Linear code linear program* (LCLP)

$$\begin{aligned} &\text{minimize} && \boldsymbol{\gamma}^{\mathsf{T}}\tilde{\boldsymbol{c}} \\ &\text{subject to} && \boldsymbol{T}_j\tilde{\boldsymbol{c}} \in \mathcal{P}_{d_j}, \quad \forall j \in \mathcal{J} \end{aligned}$$

LP Decoding using ADMM

- Solution using the *alternating direction method of multipliers* (ADMM) [Bar+13]
- Slight reformulation of the LCLP:

$$\begin{aligned} &\text{minimize} && \boldsymbol{\gamma}^{\mathsf{T}}\tilde{\boldsymbol{c}} + \sum_{j\in\mathcal{J}} g_j\left(\boldsymbol{z}_j\right) \\ &\text{subject to} && \boldsymbol{T}_j\tilde{\boldsymbol{c}} = \boldsymbol{z}_j, \quad \forall j \in \mathcal{J} \end{aligned}, \qquad g_j\left(\boldsymbol{t}\right) := \begin{cases} 0, & \boldsymbol{t} \in \mathcal{P}_{d_j} \\ +\infty, & \boldsymbol{t} \notin \mathcal{P}_{d_j} \end{cases}$$

- Iterative algorithm:

$$\begin{aligned} \tilde{\boldsymbol{c}} &\leftarrow \operatorname*{argmin}_{\tilde{\boldsymbol{c}}} \boldsymbol{\gamma}^{\mathsf{T}}\tilde{\boldsymbol{c}} + \frac{\mu}{2}\sum_{j\in\mathcal{J}} \left\|\boldsymbol{T}_j\tilde{\boldsymbol{c}} - \boldsymbol{z}_j + \boldsymbol{u}_j\right\| \\ \boldsymbol{z}_j &\leftarrow \operatorname*{argmin}_{\boldsymbol{z}_j} g\left(\boldsymbol{z}_j\right) + \frac{\mu}{2}\left\|\boldsymbol{T}_j\tilde{\boldsymbol{c}} - \boldsymbol{z}_j + \boldsymbol{u}_j\right\|, && \forall j \in \mathcal{J} \\ \boldsymbol{u}_j &\leftarrow \boldsymbol{u}_j + \boldsymbol{T}_j\tilde{\boldsymbol{c}} - \boldsymbol{z}_j, && \forall j \in \mathcal{J} \end{aligned}$$

[Bar+13] Siddharth Barman et al.: *Decomposition Methods for Large Scale LP Decoding*. IEEE Transactions on Information Theory 59.12 (2013), pp. 7870–7886.

LP Decoding using ADMM

- Convergence properties enhanced by over-relaxation with parameter ρ
- Simplified rules:

$$\tilde{c}_i \leftarrow \frac{1}{|N_v(i)|}\left(\sum_{j\in N_v(i)}\left(\left(\boldsymbol{z}_j\right)_i - \left(\boldsymbol{u}_j\right)_i\right) - \frac{\gamma_i}{\mu}\right) \qquad \forall i \in \mathcal{I}$$

$$\boldsymbol{z}_j \leftarrow \Pi_{\mathcal{P}_{d_j}}\left(\rho \boldsymbol{T}_j \tilde{\boldsymbol{c}} - \boldsymbol{z}_j\left(1-\rho\right) + \boldsymbol{u}_j\right) \qquad \forall j \in \mathcal{J}$$

$$\boldsymbol{u}_j \leftarrow \boldsymbol{u}_j + \rho \boldsymbol{T}_j \tilde{\boldsymbol{c}} - \left(1-\rho\right)\boldsymbol{z}_j - \boldsymbol{z}_j \qquad \forall j \in \mathcal{J}$$

- The projections $\Pi_{\mathcal{P}_{d_j}},\ j \in \mathcal{J}$ are the main computational effort
- Many approaches exist [Bar+13], [ZS13], [Gen+20]
- The approach chosen here is the one described in [Bar+13]

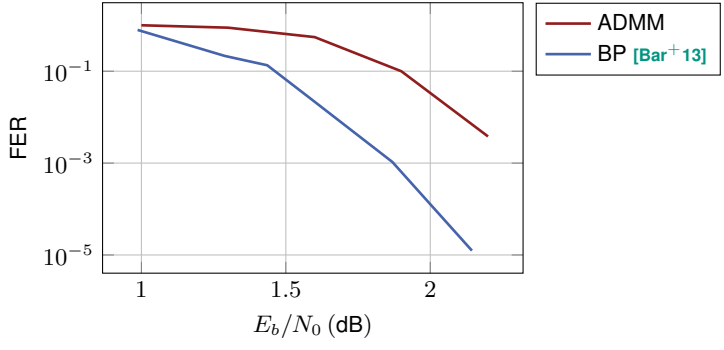
[Bar+13] Siddharth Barman et al.: *Decomposition Methods for Large Scale LP Decoding.* IEEE Transactions on Information Theory 59.12 (2013), pp. 7870–7886.

[ZS13] Xiaojie Zhang; Paul H. Siegel: *Efficient iterative LP decoding of LDPC codes with alternating direction method of multipliers.* 2013 IEEE International Symposium on Information Theory. 2013, pp. 1501–1505.

[Gen+20] Florian Gensheimer et al.: *A Reduced-Complexity Projection Algorithm for ADMM-Based LP Decoding.* IEEE Transactions on Information Theory 66.8 (2020), pp. 4819–4833.

LP Decoding using ADMM: Frame Error Rate

- "Margulis" LDPC code with $n = 2640$, $k = 1320$ [Mac23, Margulis2640.1320.3]
- Comparison of simulation with results from Barman et al. [Bar+13]



[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.

[Bar+13] Siddharth Barman et al.: *Decomposition Methods for Large Scale LP Decoding*. IEEE Transactions on Information Theory 59.12 (2013), pp. 7870–7886.

Overview

- Theoretical Background
- LP Decoding using ADMM
  - LP Decoding
  - Application of ADMM
  - Simulation Results
- Proximal Decoding
  - Decoding Algorithm
  - Simulation Results
  - Improved Algorithm

Proximal Decoding: General Approach [WT22]

- MAP rule as continuous maximization problem:

$$\begin{aligned}\hat{\boldsymbol{x}} &= \underset{\tilde{\boldsymbol{x}}\in\mathbb{R}^n}{\operatorname{argmax}} f_{\boldsymbol{Y}}\left(\boldsymbol{y}|\tilde{\boldsymbol{x}}\right) f_{\boldsymbol{X}}\left(\tilde{\boldsymbol{x}}\right) \\ &= \underset{\tilde{\boldsymbol{x}}\in\mathbb{R}^n}{\operatorname{argmax}} \mathrm{e}^{-L\left(\boldsymbol{y}|\tilde{\boldsymbol{x}}\right)} f_{\boldsymbol{X}}\left(\tilde{\boldsymbol{x}}\right), \qquad L\left(\boldsymbol{y}|\tilde{\boldsymbol{x}}\right) = -\ln\left(f_{\boldsymbol{Y}}\left(\boldsymbol{y}|\tilde{\boldsymbol{x}}\right)\right)\end{aligned}$$

- Approximation of prior PDF:

$$f_{\boldsymbol{X}}\left(\tilde{\boldsymbol{x}}\right) = \frac{1}{|\mathcal{C}|}\sum_{\boldsymbol{c}\in\mathcal{C}} \delta\left(\tilde{\boldsymbol{x}} - (-1)^{\boldsymbol{c}}\right) \approx \frac{1}{Z} e^{-\gamma h(\tilde{\boldsymbol{x}})}$$

- Code constraint polynomial:

$$h\left(\tilde{\boldsymbol{x}}\right) = \underbrace{\sum_{i=1}^{n}\left(\tilde{x}_i^2 - 1\right)^2}_{\text{Bipolar constraint}} + \underbrace{\sum_{j=1}^{m}\left[\left(\prod_{i\in N_v(j)} \tilde{x}_i\right) - 1\right]^2}_{\text{Parity constraint}},$$

$$\mathcal{I} := [1:n], \quad \mathcal{J} := [1:m]$$
$$N_v\left(j\right) := \left\{i | i \in \mathcal{I}, \boldsymbol{H}_{j,i} = 1\right\}, j \in \mathcal{J}$$

[WT22] Tadashi Wadayama; Satoshi Takabe: *Proximal Decoding for LDPC Codes.* IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences advpub (2022), 2022TAP0002.

Proximal Decoding: General Approach

- Objective function:

$$g(\tilde{\boldsymbol{x}}) = L(\boldsymbol{y}|\tilde{\boldsymbol{x}}) + \gamma h(\tilde{\boldsymbol{x}})$$

- Proximal operator [PB14]:

$$\begin{aligned}\text{prox}_{\gamma h}(\tilde{\boldsymbol{x}}) &\equiv \underset{\boldsymbol{t}\in\mathbb{R}^n}{\operatorname{argmin}}\, \gamma h(\boldsymbol{t}) + \frac{1}{2}\|\boldsymbol{t}-\tilde{\boldsymbol{x}}\|^2 \\ &\approx \tilde{\boldsymbol{x}} - \gamma \nabla h(\tilde{\boldsymbol{x}}), \qquad \gamma \text{ small}\end{aligned}$$

- Iterative decoding process:

$$\begin{aligned}\boldsymbol{r} &\leftarrow \boldsymbol{s} - \omega \nabla L(\boldsymbol{y}|\boldsymbol{s}), \qquad \omega > 0 \qquad \text{“Gradient descent step”} \\ \boldsymbol{s} &\leftarrow \boldsymbol{r} - \gamma \nabla h(\boldsymbol{r}), \qquad \gamma > 0 \qquad \text{“Code proximal step”}\end{aligned}$$

[PB14] Neal Parikh; Stephen Boyd: *Proximal Algorithms.* Found. Trends Optim. 1.3 (Jan. 2014), pp. 127–239.

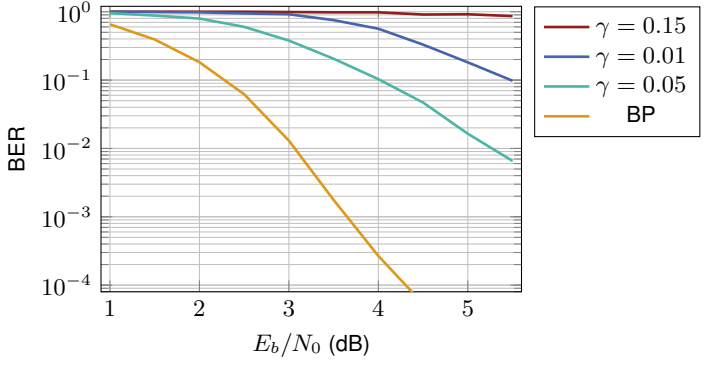
Proximal Decoding: Algorithm

- Iterative decoding algorithm:

$\boldsymbol{s} \leftarrow \mathbf{0}$
for K iterations **do**
    $\boldsymbol{r} \leftarrow \boldsymbol{s} - \omega \nabla L\left(\boldsymbol{y} \mid \boldsymbol{s}\right)$
    $\boldsymbol{s} \leftarrow \boldsymbol{r} - \gamma \nabla h\left(\boldsymbol{r}\right)$
    $\hat{\boldsymbol{x}} \leftarrow \text{sign}\left(\boldsymbol{s}\right)$
    if $\boldsymbol{H}\hat{\boldsymbol{c}} = \mathbf{0}$ **do**
        return $\hat{\boldsymbol{c}}$
    end if
end for
return $\hat{\boldsymbol{c}}$

Proximal Decoding: Bit Error Rate

- (3,6) regular LDPC code with $n = 204, k = 102$ [Mac23, 204.33.484]
- Comparison of simulation with results of Wadayama et al. [WT22]



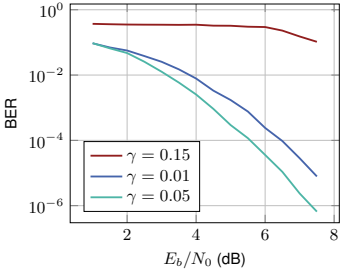
[WT22] Tadashi Wadayama; Satoshi Takabe: *Proximal Decoding for LDPC Codes.* IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences advpub (2022), 2022TAP0002.

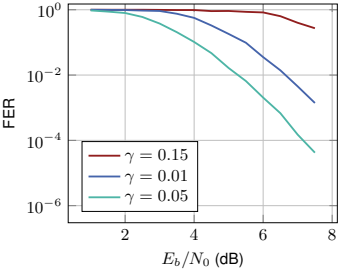
[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes.* Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`

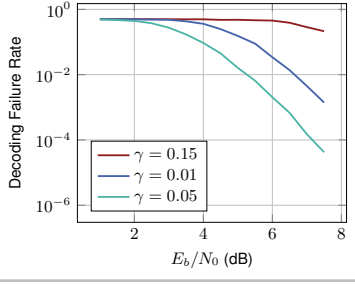
Proximal Decoding: Frame Error Rate

- (3,6) regular LDPC code with $n = 204$, $k = 102$ **[Mac23, 204.33.484]**

$\boldsymbol{s} \leftarrow \mathbf{0}$
for K iterations **do**
  $\boldsymbol{r} \leftarrow \boldsymbol{s} - \omega \nabla L(\boldsymbol{y} \mid \boldsymbol{s})$
  $\boldsymbol{s} \leftarrow \boldsymbol{r} - \gamma \nabla h(\boldsymbol{r})$
  $\hat{\boldsymbol{x}} \leftarrow \text{sign}(\boldsymbol{s})$
  if $H\hat{c} = 0$ **do**
    return $\hat{c}$
  end if
end for
return $\hat{c}$



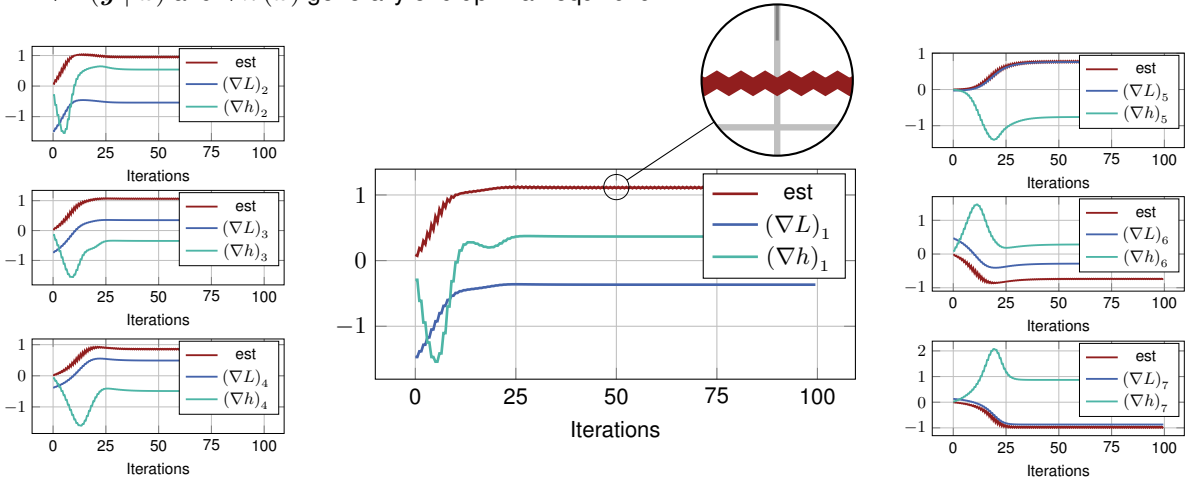




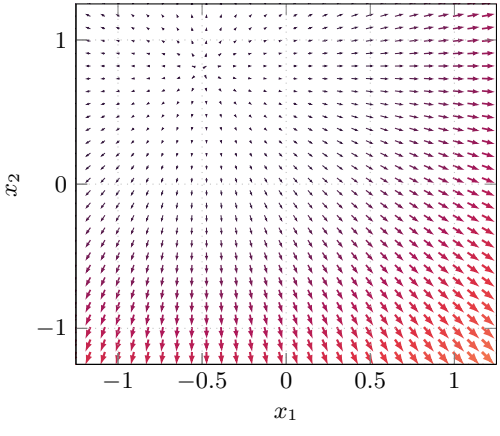
[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.

Proximal Decoding: Oscillation of Estimate

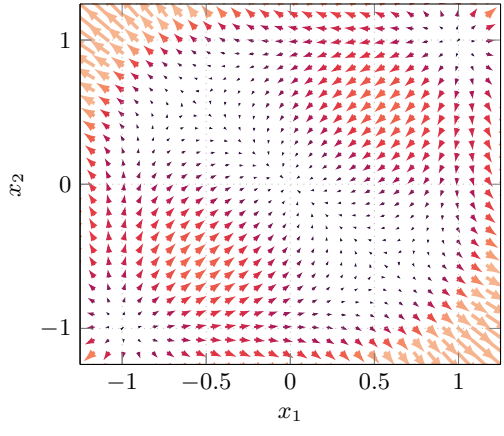
- Single decoding using the BCH$(7, 4)$ code; $E_b/N_0 = 5\,\mathrm{dB}$
- $\nabla L\left(\boldsymbol{y} \mid \tilde{\boldsymbol{x}}\right)$ and $\nabla h\left(\tilde{\boldsymbol{x}}\right)$ generally end up in an equilibrium



Proximal Decoding: Oscillation of Estimate



(a) $\nabla L\left(\boldsymbol{y} \mid \tilde{\boldsymbol{x}}\right)$[1] for a repetition code with $n = 2$

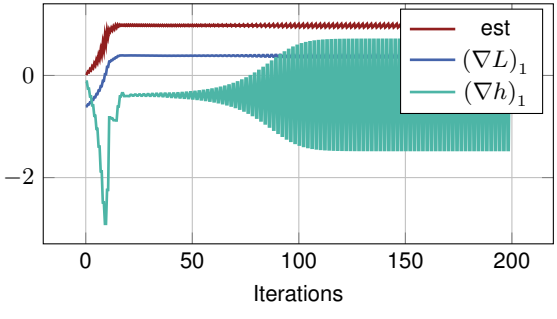


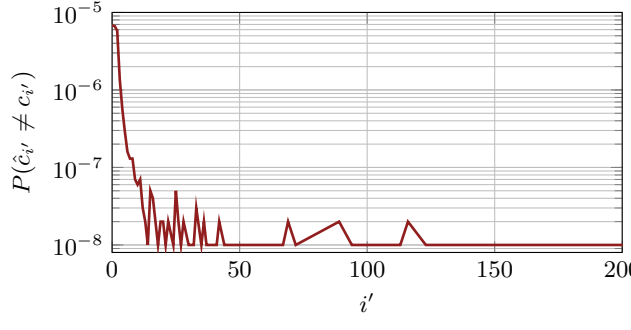
(b) $\nabla h\left(\tilde{\boldsymbol{x}}\right)$ for a repetition code with $n = 2$

[1] In an AWGN Channel $\nabla L\left(\boldsymbol{y} \mid \tilde{\boldsymbol{x}}\right) \propto \left(\tilde{\boldsymbol{x}} - \boldsymbol{y}\right)$

Proximal Decoding: Oscillation of Estimate

- Single decoding using a (3,6) regular LDPC code with $n = 204, k = 102$ **[Mac23, 204.33.484]**
- For larger n, the gradient itself starts to oscillate
- Amplitude of oscillation highly correlated to probability of bit error





[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: http://www.inference.org.uk/mackay/codes/data.html.

Improved Algorithm

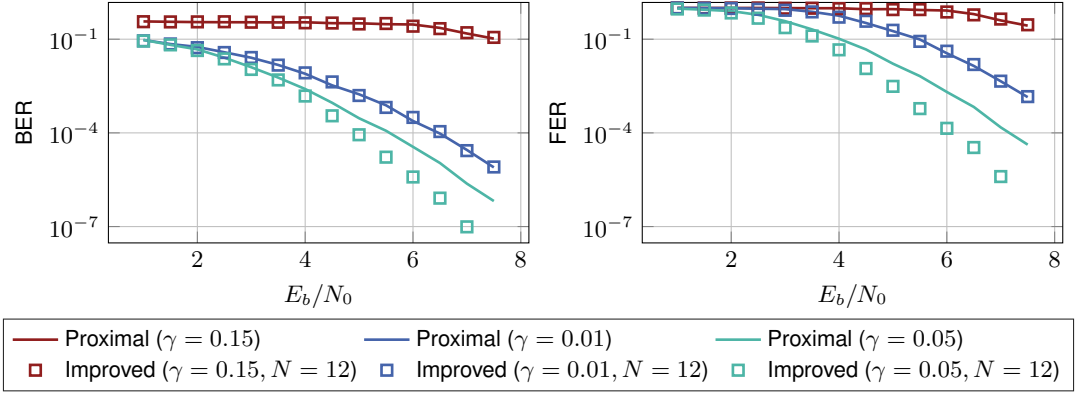
- Improvement of proximal decoding by addition of “ML-in-the-List” step after iterating

$\boldsymbol{s} \leftarrow \mathbf{0}$
for K iterations **do**
  $\boldsymbol{r} \leftarrow \boldsymbol{s} - \omega \nabla L\left(\boldsymbol{y} \mid \boldsymbol{s}\right)$
  $\boldsymbol{s} \leftarrow \boldsymbol{r} - \gamma \nabla h\left(\boldsymbol{r}\right)$
  $\hat{\boldsymbol{x}} \leftarrow \text{sign}\left(\boldsymbol{s}\right)$
  if $\boldsymbol{H}\hat{\boldsymbol{c}} = \mathbf{0}$ **do**
    return $\hat{\boldsymbol{c}}$
  end if
end for
return $\hat{\boldsymbol{c}}$

$\boldsymbol{s} \leftarrow \mathbf{0}$
for K iterations **do**
  $\boldsymbol{r} \leftarrow \boldsymbol{s} - \omega \nabla L\left(\boldsymbol{y} \mid \boldsymbol{s}\right)$
  $\boldsymbol{s} \leftarrow \boldsymbol{r} - \gamma \nabla h\left(\boldsymbol{r}\right)$
  $\hat{\boldsymbol{x}} \leftarrow \text{sign}\left(\boldsymbol{s}\right)$
  if $\boldsymbol{H}\hat{\boldsymbol{c}} = \mathbf{0}$
    return $\hat{\boldsymbol{c}}$
  end if
end for
Find N most probably wrong bits
Generate variations $\hat{\boldsymbol{c}}_l$ of $\hat{\boldsymbol{c}}$ with N bits modified
Compute $\boldsymbol{y}^{\mathsf{T}}\hat{\boldsymbol{c}}_l$ for all codewords $\hat{\boldsymbol{c}}_l$
return $\hat{\boldsymbol{c}}_l$ with highest $\boldsymbol{y}^{\mathsf{T}}\hat{\boldsymbol{c}}_l$, prioritizing valid codewords

Improved Algorithm

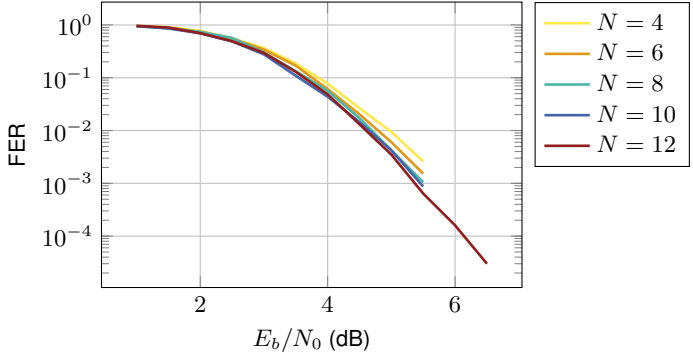
- (3,6) regular LDPC code with $n = 204, k = 102$ [Mac23, 204.33.484]
- Up to ~ 1 dB improvement



[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.

Improved Algorithm: Effect of Parameter N

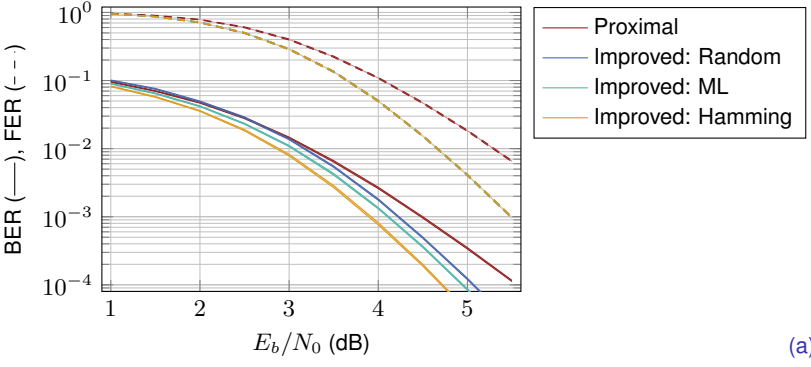
- (3,6) regular LDPC code with $n = 204, k = 102$ [Mac23, 204.33.484]
- Chosen parameters: $\gamma = 0.05$



[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.

Improved Algorithm: List Choice Policy

- (3,6) regular LDPC code with $n = 204, k = 102$ [Mac23, 204.33.484]
- Chosen parameters: $N = 8, \gamma = 0.05$



E_b/N_0	List size mean	List size var.
1.0	1.0	0.0
1.5	1.0	0.0
2.0	1.0	0.0
2.5	1.0	0.0
3.0	1.0	0.0
3.5	1.0	0.0
4.0	1.0	0.0
4.5	1.0	0.0
5.0	1.0	0.0
5.5	1.0	0.0

(a) List size statistics for successful decodings (no frame error).

[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: http://www.inference.org.uk/mackay/codes/data.html.

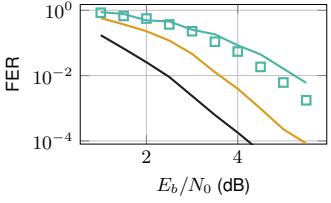
Questions

Thank you for your attention!
Any questions?

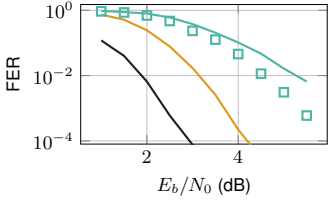
Overview

- Supplementary Slides

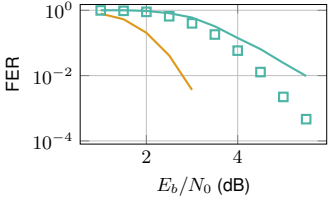
Comparison: Decoding Performance



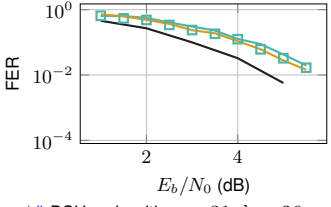
(a) $(3, 6)$-regular LDPC code with $n = 96, k = 48$ **[Mac23, 96.33.965]**



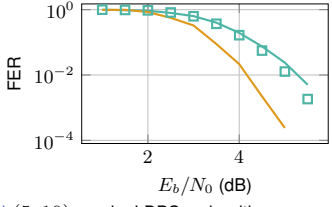
(b) $(3, 6)$-regular LDPC code with $n = 204, k = 102$ **[Mac23, 204.33.484]**



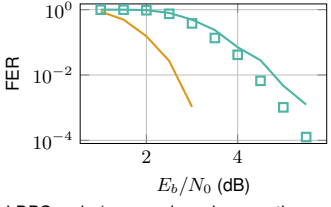
(c) $(3, 6)$-regular LDPC code with $n = 408, k = 204$ **[Mac23, 408.33.844]**



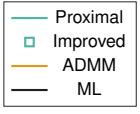
(d) BCH code with $n = 31, k = 26$



(e) $(5, 10)$-regular LDPC code with $n = 204, k = 102$ **[Mac23, 204.55.187]**



(f) LDPC code (progressive edge growth construction) with $n = 504, k = 252$ **[Mac23, PEGReg252x504]**



[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.

Comparison: Algorithm Structure

minimize $\underbrace{L\left(\boldsymbol{y} \mid \tilde{\boldsymbol{x}}\right)}_{\text{Likelihood}} + \underbrace{\gamma h\left(\tilde{\boldsymbol{x}}\right)}_{\text{Constraints}}$

subject to $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$

1 Initialize $\boldsymbol{r}, \boldsymbol{s}, \omega, \gamma$

2 **while** stopping critierion unfulfilled **do**

3 $\quad \boldsymbol{r} \leftarrow \boldsymbol{r} + \omega \nabla L\left(\boldsymbol{y} \mid \boldsymbol{s}\right)$

4 $\quad \boldsymbol{s} \leftarrow \mathbf{prox}_{\gamma h}\left(\boldsymbol{r}\right)$

5 **end while**

6 **return** $\hat{c}$

minimize $\underbrace{\boldsymbol{\gamma}^{\mathsf{T}} \tilde{\boldsymbol{c}}}_{\text{Likelihood}} + \underbrace{\sum_{j \in \mathcal{J}} g_j\left(\boldsymbol{T}_j \tilde{\boldsymbol{c}}\right)}_{\text{Constraints}}$

subject to $\tilde{\boldsymbol{c}} \in [0,1]^n$

1 Initialize $\tilde{\boldsymbol{c}}, \boldsymbol{z}, \boldsymbol{u}, \boldsymbol{\gamma}, \rho$

2 **while** stopping criterion unfulfilled **do**

3 $\quad \tilde{\boldsymbol{c}} \leftarrow \operatorname{argmin}_{\tilde{\boldsymbol{c}}} \left(\boldsymbol{\gamma}^{\mathsf{T}} \tilde{\boldsymbol{c}} + \frac{\rho}{2} \sum_{j \in \mathcal{J}} \left\| \boldsymbol{T}_j \tilde{\boldsymbol{c}} - \boldsymbol{z}_j + \boldsymbol{u}_j \right\| \right)$

4 $\quad \boldsymbol{z}_j \leftarrow \mathbf{prox}_{g_j}\left(\boldsymbol{T}_j \tilde{\boldsymbol{c}} + \boldsymbol{u}_j\right), \qquad \forall j \in \mathcal{J}$

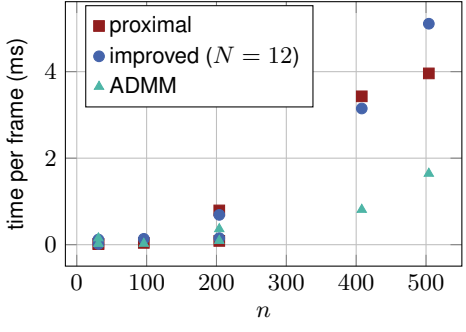
5 $\quad \boldsymbol{u}_j \leftarrow \boldsymbol{u}_j + \tilde{\boldsymbol{c}} - \boldsymbol{z}_j, \qquad \forall j \in \mathcal{J}$

6 **end while**

7 **return** $\hat{c}$

Comparison: Time Complexity

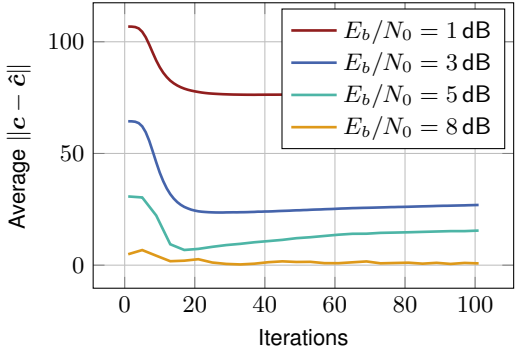
- Codes: BCH $(31, 11)$; BCH $(31, 26)$; **[Mac23, 96.3.965; 204.33.484; 204.55.187; 408.33.844; PEGReg252x504]**
- Measured performance: $\sim 10\,000$ frames/s on Intel Core i7-7700HQ @ 2.80GHz; $n = 204$
- Both algorithms are $\mathcal{O}(n)$ on average



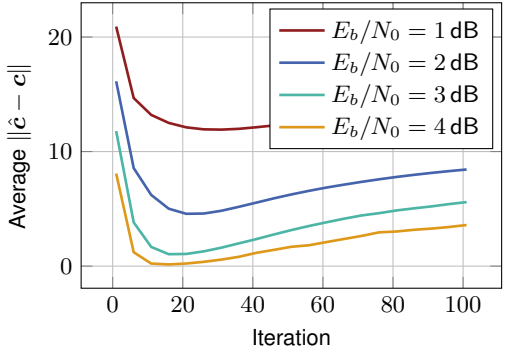
[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.

Comparison: Convergence Behavior

- (3,6) regular LDPC code with $n = 204, k = 102$ [Mac23, 204.33.484]
- Minimum number of iterations independent of SNR



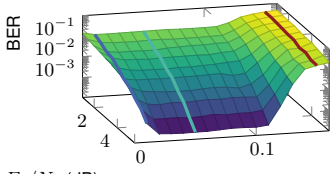
(a) Proximal Decoding



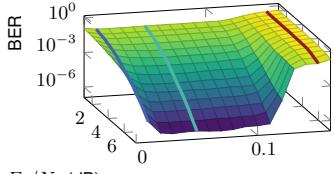
(b) LP Decoding using ADMM

[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.

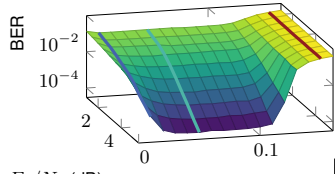
Proximal Decoding: Choice of γ



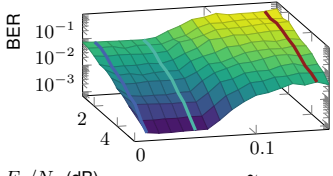
(a) $(3, 6)$-regular LDPC code with $n = 96, k = 48$ **[Mac23, 96.3.965]**



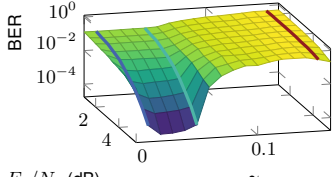
(b) $(3, 6)$-regular LDPC code with $n = 204, k = 102$ **[Mac23, 204.33.484]**



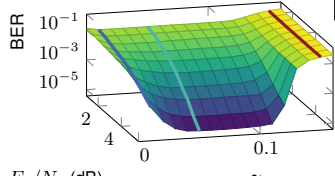
(c) $(3, 6)$-regular LDPC code with $n = 408, k = 204$ **[Mac23, 408.33.844]**



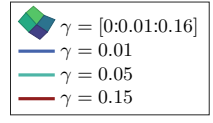
(d) BCH code with $n = 31, k = 26$



(e) $(5, 10)$-regular LDPC code with $n = 204, k = 102$ **[Mac23, 204.55.187]**



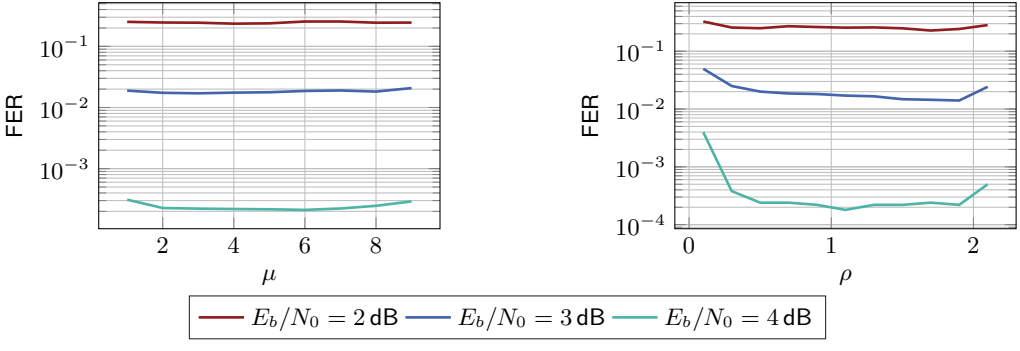
(f) LDPC code (progressive edge growth construction) with $n = 504, k = 252$ **[Mac23, PEGReg252x504]**



[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes.* Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.

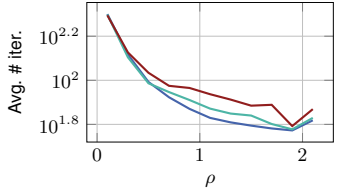
LP Decoding using ADMM: Choice of μ and ρ

- (3,6) regular LDPC code with $n = 204, k = 102$ **[Mac23, 204.33.484]**
- No clear optimum value

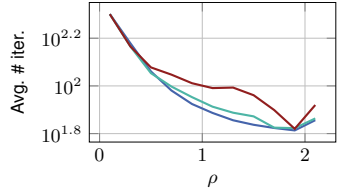


[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.

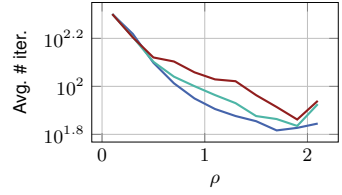
LP Decoding using ADMM: Choice of μ and ρ



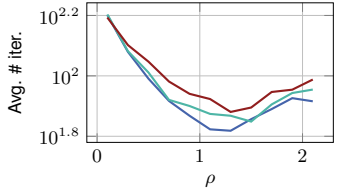
(a) $(3, 6)$-regular LDPC code with $n = 96, k = 48$ **[Mac23, 96.3.965]**



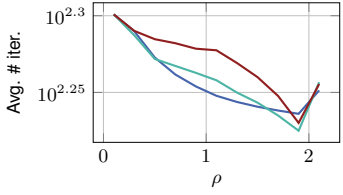
(b) $(3, 6)$-regular LDPC code with $n = 204, k = 102$ **[Mac23, 204.33.484]**



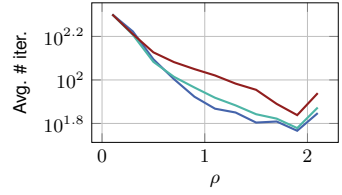
(c) $(3, 6)$-regular LDPC code with $n = 408, k = 204$ **[Mac23, 408.33.844]**



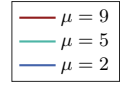
(d) BCH code with $n = 31, k = 26$



(e) $(5, 10)$-regular LDPC code with $n = 204, k = 102$ **[Mac23, 204.55.187]**



(f) LDPC code (progressive edge growth construction) with $n = 504, k = 252$ **[Mac23, PEGReg252x504]**



[Mac23] David J.C. MacKay: *Encyclopedia of Sparse Graph Codes*. Jan. 2023. URL: `http://www.inference.org.uk/mackay/codes/data.html`.