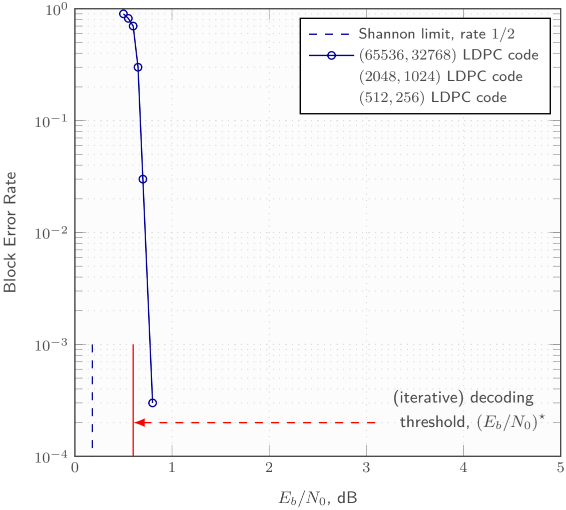
11th International ITG Conference on Systems, Communications and Coding
February 6, 2017

Channel Codes for Short Blocks: A Survey

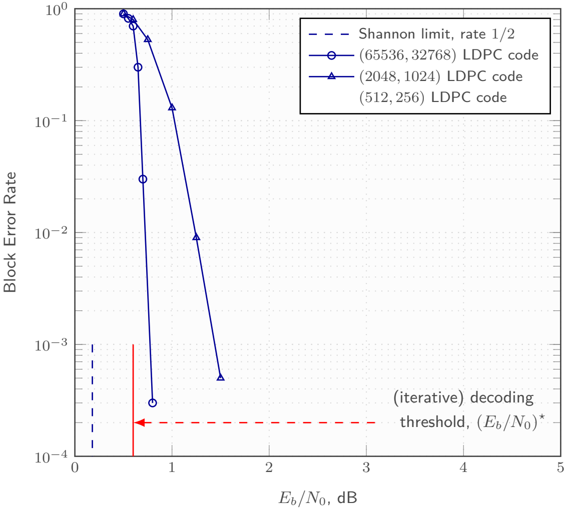
Gianluigi Liva, gianluigi.liva@dlr.de
Fabian Steiner, fabian.steiner@tum.de

Outline

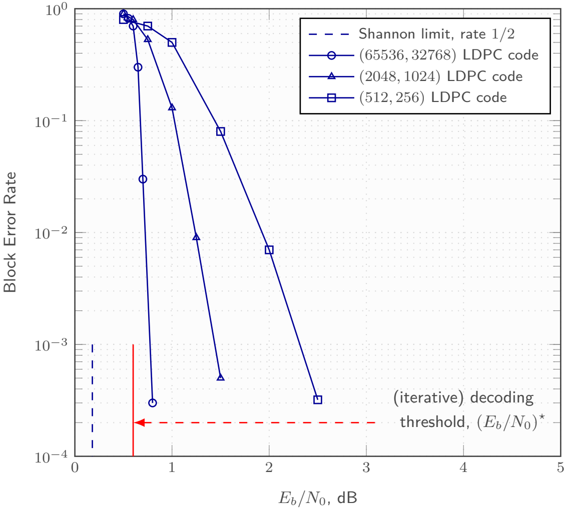
- Preliminaries
- Efficient Short Classical Codes
- Efficient Short Modern Codes
- Two Case Studies
- Beyond Error Correction
- Conclusions


Shannon limit, rate 1/2
(65536, 32768) LDPC code
(2048, 1024) LDPC code
(512, 256) LDPC code
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
E_b/N_0, dB
(iterative) decoding
threshold, $(E_b/N_0)^\star$


DLR



Shannon limit, rate 1/2
(65536, 32768) LDPC code
(2048, 1024) LDPC code
(512, 256) LDPC code
(iterative) decoding threshold, $(E_b/N_0)^\star$
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
E_b/N_0, dB



Shannon limit, rate 1/2
(65536, 32768) LDPC code
(2048, 1024) LDPC code
(512, 256) LDPC code
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
E_b/N_0, dB
(iterative) decoding
threshold, $(E_b/N_0)^\star$

Motivation

Transmission of small amounts of information[1]

- Wireless sensor networks
- Machine-type communications and IoT
- Ultra-reliable low-latency communications

[1] G. Durisi, T. Koch, and P. Popovski, "Towards Massive, Ultra-Reliable, and Low-Latency Wireless: The Art of Sending Short Packets," *ArXiv*, 2015

Motivation

Transmission of small amounts of information[1]

- Wireless sensor networks
- Machine-type communications and IoT
- Ultra-reliable low-latency communications
- *Small data*

[1] G. Durisi, T. Koch, and P. Popovski, "Towards Massive, Ultra-Reliable, and Low-Latency Wireless: The Art of Sending Short Packets," *ArXiv*, 2015

Motivation

Transmission of small amounts of information[1]

- Wireless sensor networks
- Machine-type communications and IoT
- Ultra-reliable low-latency communications
- *Small data*

Approaching capacity (with large block lengths): Problem practically solved, e.g., with iterative codes

[1] G. Durisi, T. Koch, and P. Popovski, "Towards Massive, Ultra-Reliable, and Low-Latency Wireless: The Art of Sending Short Packets," *ArXiv*, 2015

Motivation

Transmission of small amounts of information[1]

- Wireless sensor networks
- Machine-type communications and IoT
- Ultra-reliable low-latency communications
- *Small data*

Approaching capacity (with large block lengths): Problem practically solved, e.g., with iterative codes

At medium-short block lengths, iterative codes yield performance losses when compared to *classical* linear block codes, when the later are decoded with (near) maximum-likelihood algorithms

[1] G. Durisi, T. Koch, and P. Popovski, "Towards Massive, Ultra-Reliable, and Low-Latency Wireless: The Art of Sending Short Packets," *ArXiv*, 2015

Motivation

Transmission of small amounts of information[1]

- Wireless sensor networks
- Machine-type communications and IoT
- Ultra-reliable low-latency communications
- *Small data*

Approaching capacity (with large block lengths): Problem practically solved, e.g., with iterative codes

At medium-short block lengths, iterative codes yield performance losses when compared to *classical* linear block codes, when the later are decoded with (near) maximum-likelihood algorithms

Performance vs. Complexity

[1] G. Durisi, T. Koch, and P. Popovski, "Towards Massive, Ultra-Reliable, and Low-Latency Wireless: The Art of Sending Short Packets," *ArXiv*, 2015

Latency vs. Blocklength

Low latency $\neq$ Short blocks

Latency is more difficult to define[2][3][4]

- Encoding latency
- Transmission latency
- Decoding latency
- . . .

[2] T. Hehn and J.B.Huber, "LDPC codes and convolutional codes with equal structural delay: a comparison," *IEEE Trans. Commun.*, 2009

[3] S. V. Maiya, D. J. Costello, and T. E. Fuja, "Low latency coding: Convolutional codes vs. ldpc codes," *IEEE Trans. Commun.*, 2012

[4] C. Rachinger, J. B. Huber, and R. R. Müller, "Comparison of convolutional and block codes for low structural delay," *IEEE Trans. Commun.*, 2015

Latency vs. Blocklength

Low latency $\neq$ Short blocks

Latency is more difficult to define[2][3][4]

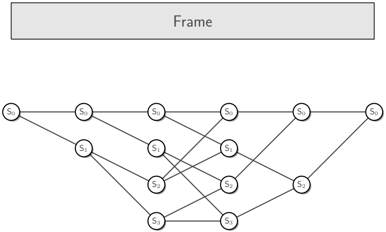
- Encoding latency
- Transmission latency
- Decoding latency
- ...

[2] T. Hehn and J.B.Huber, "LDPC codes and convolutional codes with equal structural delay: a comparison," *IEEE Trans. Commun.*, 2009

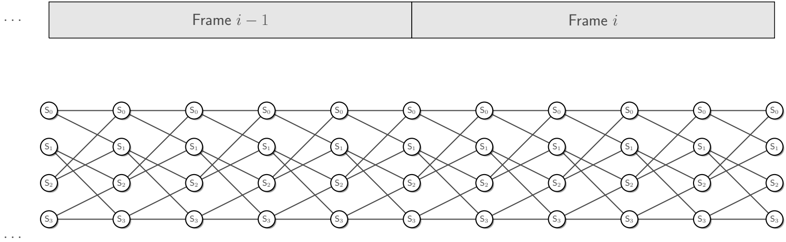
[3] S. V. Maiya, D. J. Costello, and T. E. Fuja, "Low latency coding: Convolutional codes vs. ldpc codes," *IEEE Trans. Commun.*, 2012

[4] C. Rachinger, J. B. Huber, and R. R. Müller, "Comparison of convolutional and block codes for low structural delay," *IEEE Trans. Commun.*, 2015

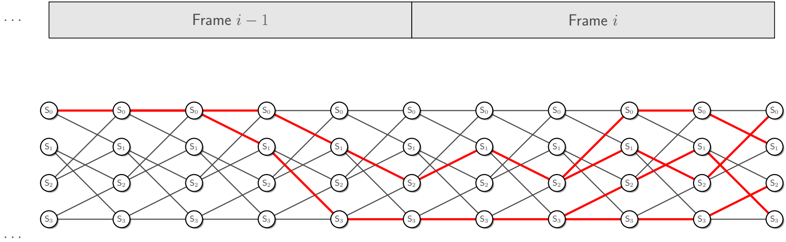
Latency vs. Blocklength
Decoding Latency



Latency vs. Blocklength
Decoding Latency



Latency vs. Blocklength
Decoding Latency



Example: Viterbi with back-tracking decouples
block length and decoding latency

What We Will Consider (and What Not)

We will consider

- Block length rather than latency

What We Will Consider (and What Not)

We will consider

- Block length rather than latency
  - Short blocks are not only required for low latency

What We Will Consider (and What Not)

We will consider

- Block length rather than latency
  - Short blocks are not only required for low latency
  - Wireless sensor networks, machine-type communications, etc.: The data units can be inherently small

What We Will Consider (and What Not)

We will consider

- Block length rather than latency
  - Short blocks are not only required for low latency
  - Wireless sensor networks, machine-type communications, etc.: The data units can be inherently small
- The binary-input additive white Gaussian noise (AWGN) channel

What We Will Consider (and What Not)

We will consider

- Block length rather than latency
    - Short blocks are not only required for low latency
    - Wireless sensor networks, machine-type communications, etc.: The data units can be inherently small
- The binary-input additive white Gaussian noise (AWGN) channel
- At some point, channel uncertainty

What We Will Consider (and What Not)

We will consider

- Block length rather than latency
  - Short blocks are not only required for low latency
  - Wireless sensor networks, machine-type communications, etc.: The data units can be inherently small
- The binary-input additive white Gaussian noise (AWGN) channel
- At some point, channel uncertainty

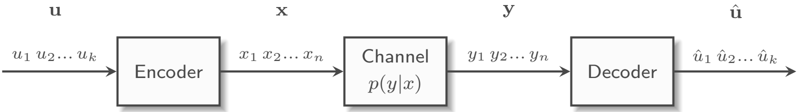
We will not consider

- Fading
- Feedback
- Synchronization
- High-order modulations (with an exception...)

Outline

- Preliminaries
- Efficient Short Classical Codes
- Efficient Short Modern Codes
- Two Case Studies
- Beyond Error Correction
- Conclusions

Preliminaries
Binary Input AWGN Channel



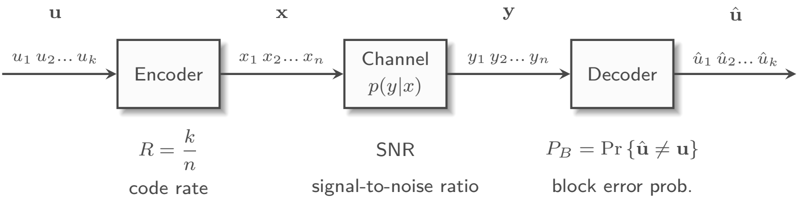
$R = \dfrac{k}{n}$ code rate

SNR signal-to-noise ratio

$P_B = \Pr\{\hat{\mathbf{u}} \neq \mathbf{u}\}$ block error prob.

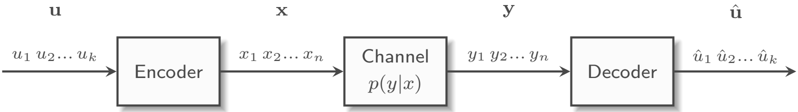
Input alphabet $\mathcal{X} = \{\pm 1\}$

Preliminaries
Binary Input AWGN Channel



Output alphabet $\mathcal{Y} \equiv \mathbb{R}$

Preliminaries
Binary Input AWGN Channel



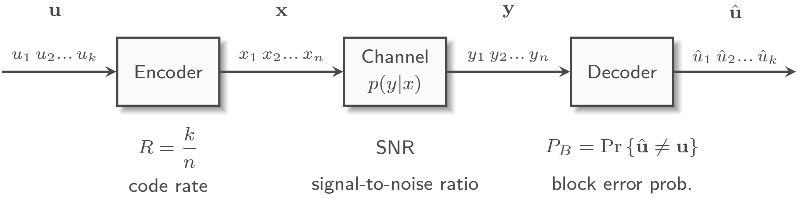
$R = \dfrac{k}{n}$ code rate

SNR signal-to-noise ratio

$P_B = \Pr\{\hat{\mathbf{u}} \neq \mathbf{u}\}$ block error prob.

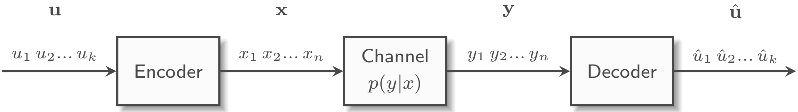
$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y-x)^2\right]$$

Preliminaries
Binary Input AWGN Channel



$$\frac{E_b}{N_0} = \frac{1}{2R\sigma^2}$$

Preliminaries
Binary Input AWGN Channel



$R = \frac{k}{n}$ code rate

SNR signal-to-noise ratio

$P_B = \Pr\{\hat{\mathbf{u}} \neq \mathbf{u}\}$ block error prob.

$$C = 1 - \mathbb{E}\left[\log_2\left(1 + \exp\left(-\frac{2}{\sigma^2}XY\right)\right)\right] \qquad \text{[bpcu]}$$

Binary Linear Block Codes

- (n, k) binary linear block code $\mathcal{C}$: Defined by $k \times n$ generator matrix $\mathbf{G}$
- Alternatively, it may be defined through its $(n-k) \times n$ parity-check matrix $\mathbf{H}$

$$\mathbf{c}\mathbf{H}^T = \mathbf{0}$$

  where $\mathbf{0}$ is the $n-k$-elements all-zero vector
- The parity-check matrix rows span the subspace orthogonal to $\mathcal{C}$. We denote such subspace as $\mathcal{C}^{\perp}$

$$\mathbf{c} \perp \mathbf{v} \qquad \forall \mathbf{c} \in \mathcal{C}, \mathbf{v} \in \mathcal{C}^{\perp}.$$

- The vectors in $\mathcal{C}^{\perp}$ form a linear block code of dimension $n-k$, referred to as dual code of $\mathcal{C}$

Finite-Length Benchmarks

Denote by $\mathcal{C}^\star$ the best (n, k) binary code

Finite-Length Benchmarks

Denote by $\mathcal{C}^\star$ the best (n,k) binary code

We are interested in

$$P_B(\mathcal{C}^\star)$$

Gallager's Random Coding Bound[5]

Upper bound on $P_B(\mathcal{C}^\star)$

$$P_B(\mathcal{C}^\star) \le \mathbb{E}\left[P_B(\mathcal{C})\right] \le P_U(n,k)$$

where

$$P_U(n,k) = 2^{-nE_G(R)}$$

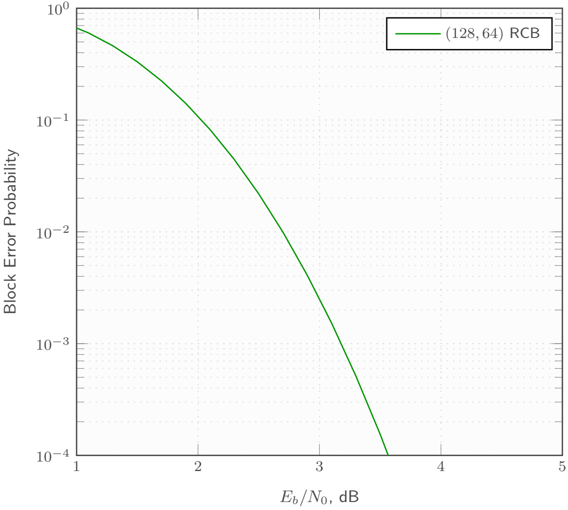
with

$$E_G(R) = \max_{0\le\rho\le 1}\left[E_0(\rho) - \rho R\right]$$

and

$$E_0(\rho) = -\log_2 \mathbb{E}\left[\left(\frac{\mathbb{E}\left[p(Y|\tilde{X})^{\frac{1}{1+\rho}}\middle|Y\right]}{p(Y|X)^{\frac{1}{1+\rho}}}\right)^{\rho}\right]$$

[5] R. Gallager, *Information theory and reliable communication*. Wiley, 1968



$(128, 64)$ RCB
Block Error Probability
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB

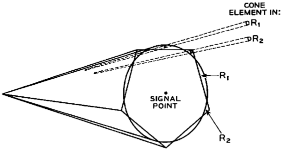
Sphere Packing Bound[6]

$$P_B(\mathcal{C}^\star) \geq P_L(n,k)$$

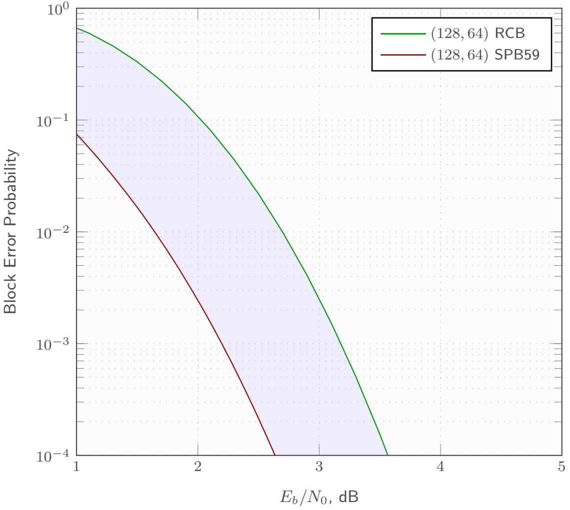
where

$$P_L(n,k)$$

is a lower bound on the error probability of n-dimensional spherical codes with 2^k codewords (cone packing)



[6] C. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell Sys. Tech. J.*, 1959



$(128, 64)$ RCB
$(128, 64)$ SPB59
Block Error Probability
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB

Normal Approximation[7,8]

Approximation of $P_B(\mathcal{C}^\star)$

$$P_B(\mathcal{C}^\star) \approx Q\left(\frac{n(C-R)+\frac{1}{2}\log_2 n}{\sqrt{nV}}\right)$$

where with $X \sim$ uniform

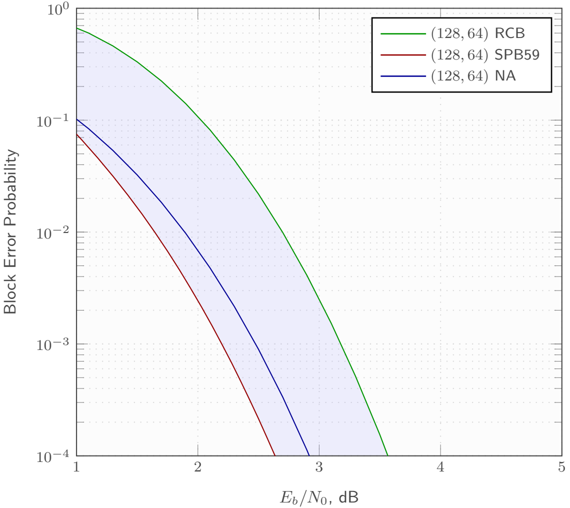
$$C = \mathbb{E}\left[\log_2 \frac{p(Y|X)}{p(Y)}\right] \qquad \text{(channel capacity)}$$

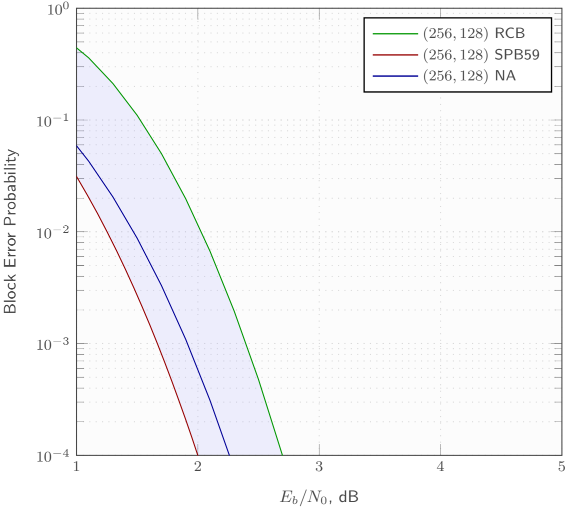
$$V = \mathrm{Var}\left[\log_2 \frac{p(Y|X)}{p(Y)}\right] \qquad \text{(channel dispersion)}$$

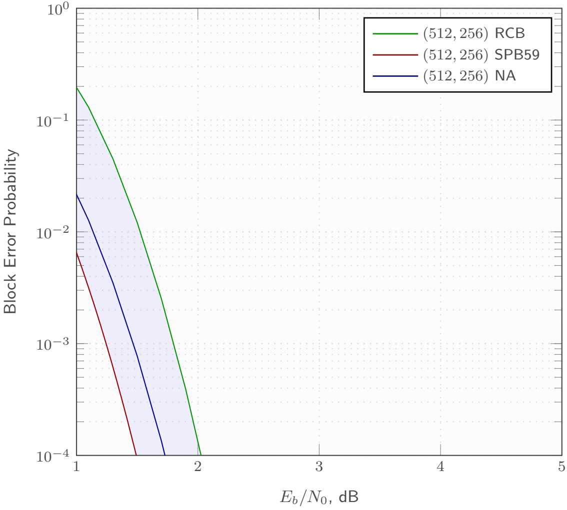
---

[7] V. Strassen, "Asymptotische Abschätzungen in Shannon's Informationstheorie," *Publ. Czech. Academy of Sciences*, 1962

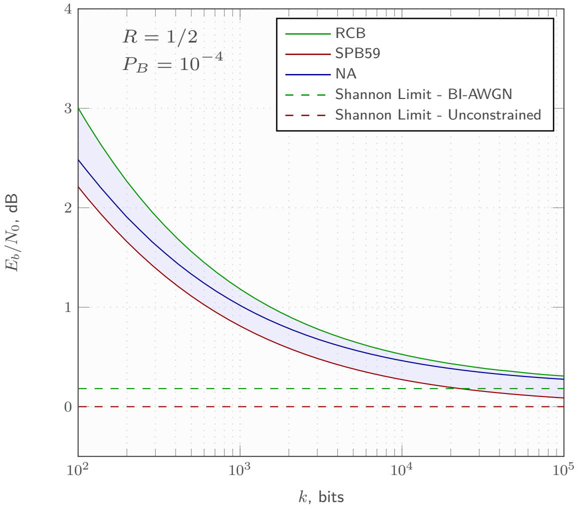
[8] Y. Polyanskiy, V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, 2010


$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
Block Error Probability
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB


(256, 128) RCB
(256, 128) SPB59
(256, 128) NA
Block Error Probability
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB



10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Probability
1
2
3
4
5
E_b/N_0, dB
$(512, 256)$ RCB
$(512, 256)$ SPB59
$(512, 256)$ NA


$R = 1/2$
$P_B = 10^{-4}$
RCB
SPB59
NA
Shannon Limit - BI-AWGN
Shannon Limit - Unconstrained
4
3
2
1
0
E_b/N_0, dB
10^2
10^3
10^4
10^5
k, bits

Spectre: short packet communication toolbox

https://sites.google.com/site/durisi/software

Efficient Short Channel Codes

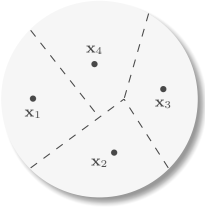
Classical

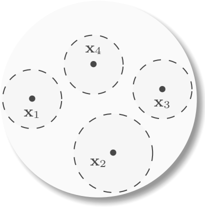
- Algebraic codes (BCH, Reed-Solomon, etc.)
- (Tail-biting) convolutional codes

Modern

- Turbo codes (parallel concatenation)
- Low-density parity-check (LDPC) codes, binary and non-binary
- Polar codes

Decoder Types
Complete vs. Incomplete[9]





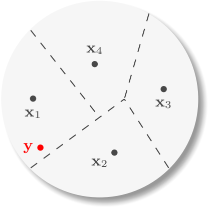
Complete:
- maximum-likelihood
- ordered statistics
- successive cancellation etc.

Incomplete:
- bounded distance*
- belief propagation* etc.

[9] G. Forney, "Exponential error bounds for erasure, list, and decision feedback schemes," *IEEE Trans. Inf. Theory*, 1968

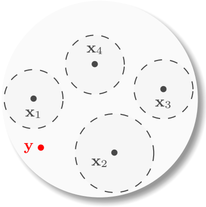
Decoder Types

Complete vs. Incomplete[9]



Complete:

- maximum-likelihood
- ordered statistics
- successive cancellation etc.
- all errors are undetected



Incomplete:

- bounded distance⋆
- belief propagation⋆ etc.
- error detection capability

[9] G. Forney, "Exponential error bounds for erasure, list, and decision feedback schemes," *IEEE Trans. Inf. Theory*, 1968

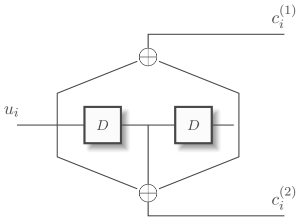
Outline

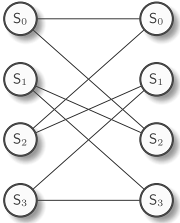
- Preliminaries
- Efficient Short Classical Codes
  - Tail-Biting Convolutional Codes
  - Near Maximum Likelihood Decoding of Linear Block Codes
- Efficient Short Modern Codes
- Two Case Studies
- Beyond Error Correction
- Conclusions

Outline

- Preliminaries
- Efficient Short Classical Codes
  - Tail-Biting Convolutional Codes
  - Near Maximum Likelihood Decoding of Linear Block Codes
- Efficient Short Modern Codes
- Two Case Studies
- Beyond Error Correction
- Conclusions

Convolutional Codes
Definitions by Example (Binary-Input Only)

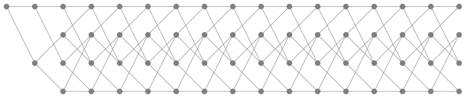




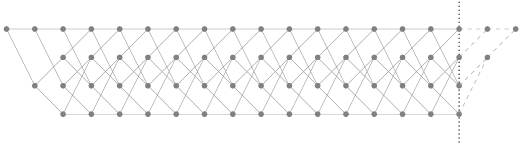
- $k_0 = 1$ inputs and $n_0 = 2$ outputs per clock
- Nominal rate $R_0 = k_0/n_0 = 1/2$
- Memory $m = 2$
- 2^m states per section
- $2^{k_0} = 2$ edges leaving each state

Trellises
Termination Strategies for Convolutional Codes

Convolutional codes to block codes: Run the encoder for k/k_0 clocks, then stop



Truncation: Block error probability rises to the last bits

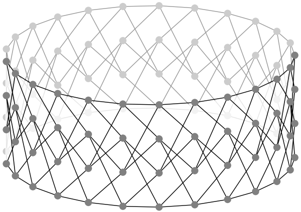


Zero-tail: Improved block error probability BUT rate loss

$$R = \frac{k}{k+m} R_0$$

Trellises

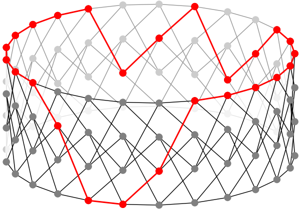
Termination Strategies for Convolutional Codes



Tail-biting:

- Force initial = final state

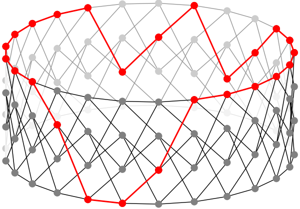
Trellises
Termination Strategies for Convolutional Codes



Tail-biting:

- Force initial $=$ final state
- Codewords $\equiv$ circular paths

Trellises
Termination Strategies for Convolutional Codes

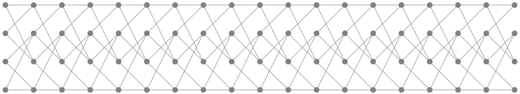


Tail-biting:

- Force initial = final state
- Codewords ≡ circular paths
- No rate loss, but decoding gets more complex...

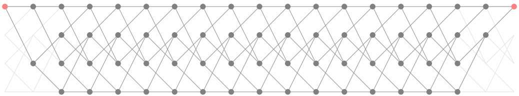
Tail-Biting Convolutional Codes
Maximum-Likelihood Decoding

- Unroll the tail-biting trellis



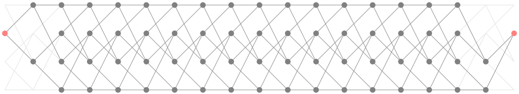
Tail-Biting Convolutional Codes
Maximum-Likelihood Decoding

- Unroll the tail-biting trellis
- Run 2^m instances of the Viterbi algorithm, one per initial/final state hypothesis



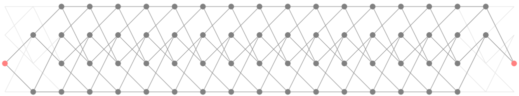
Tail-Biting Convolutional Codes
Maximum-Likelihood Decoding

- Unroll the tail-biting trellis
- Run 2^m instances of the Viterbi algorithm, one per initial/final state hypothesis



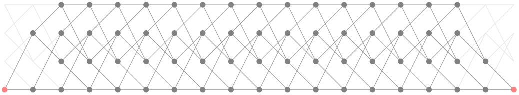
Tail-Biting Convolutional Codes
Maximum-Likelihood Decoding

- Unroll the tail-biting trellis
- Run 2^m instances of the Viterbi algorithm, one per initial/final state hypothesis



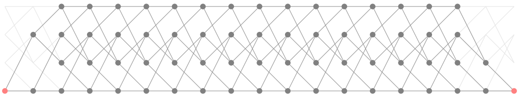
Tail-Biting Convolutional Codes
Maximum-Likelihood Decoding

- Unroll the tail-biting trellis
- Run 2^m instances of the Viterbi algorithm, one per initial/final state hypothesis



Tail-Biting Convolutional Codes
Maximum-Likelihood Decoding

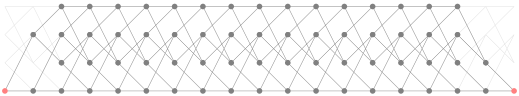
- Unroll the tail-biting trellis
- Run 2^m instances of the Viterbi algorithm, one per initial/final state hypothesis



- Each decoder produces a decision (path): List of 2^m codewords
- Select the most likely codeword in the list

Tail-Biting Convolutional Codes
Maximum-Likelihood Decoding

- Unroll the tail-biting trellis
- Run 2^m instances of the Viterbi algorithm, one per initial/final state hypothesis



- Each decoder produces a decision (path): List of 2^m codewords
- Select the most likely codeword in the list
- Complexity of (almost) 2^m Viterbi decoders, quadratic in 2^m

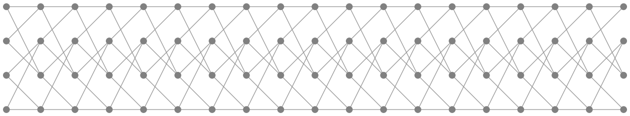
Tail-Biting Convolutional Codes
Wrap-Around Viterbi Algorithm (WAVA)[10]

- Runs the Viterbi algorithm successively for more iterations
- Improves the reliability of the decision at each iteration
- Achieves near-optimal performance

[10] R. Y.Shao, S. Lin, and M. P. Fossorier, "Two decoding algorithms for tailbiting codes," *IEEE Trans. Commun.*, 2003

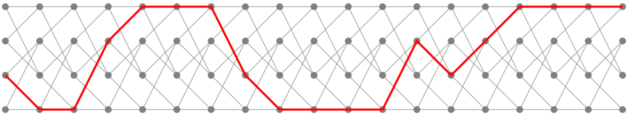
Tail-Biting Convolutional Codes
Wrap-Around Viterbi Algorithm (WAVA)

- Start decoding with equiprobable initial states



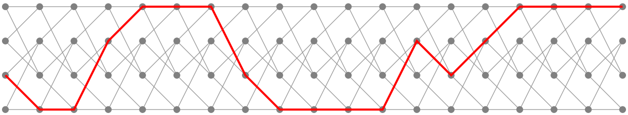
Tail-Biting Convolutional Codes
Wrap-Around Viterbi Algorithm (WAVA)

- Start decoding with equiprobable initial states
- Run a first Viterbi algorithm iteration, and output the most likely path $\mathcal{P}$



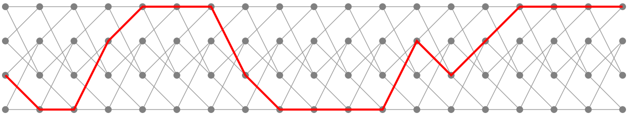
Tail-Biting Convolutional Codes
Wrap-Around Viterbi Algorithm (WAVA)

- Start decoding with equiprobable initial states
- Run a first Viterbi algorithm iteration, and output the most likely path $\mathcal{P}$
- Is $\mathcal{P}$ a tail-biting path?



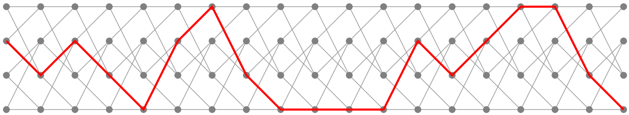
Tail-Biting Convolutional Codes
Wrap-Around Viterbi Algorithm (WAVA)

- Start decoding with equiprobable initial states
- Run a first Viterbi algorithm iteration, and output the most likely path $\mathcal{P}$
- Is $\mathcal{P}$ a tail-biting path?
  - YES: stop
  - NO: replace the initial state metrics with the computed final state metrics, and perform another Viterbi algorithm iteration



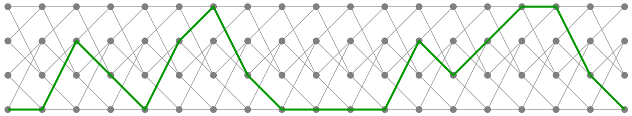
Tail-Biting Convolutional Codes
Wrap-Around Viterbi Algorithm (WAVA)

- Start decoding with equiprobable initial states
- Run a first Viterbi algorithm iteration, and output the most likely path $\mathcal{P}$
- Is $\mathcal{P}$ a tail-biting path?
  - YES: stop
  - NO: replace the initial state metrics with the computed final state metrics, and perform another Viterbi algorithm iteration



Tail-Biting Convolutional Codes
Wrap-Around Viterbi Algorithm (WAVA)

- Start decoding with equiprobable initial states
- Run a first Viterbi algorithm iteration, and output the most likely path $\mathcal{P}$
- Is $\mathcal{P}$ a tail-biting path?
  - YES: stop
  - NO: replace the initial state metrics with the computed final state metrics, and perform another Viterbi algorithm iteration
- A maximum number of iterations is allowed (e.g., 4)



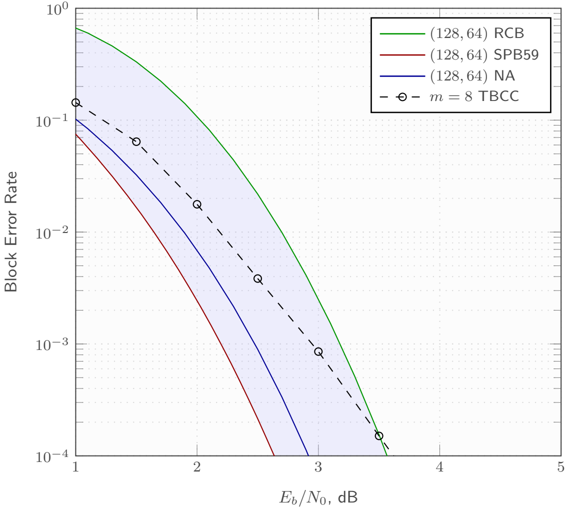
Tail-Biting Convolutional Codes

Examples of Good (Time-Invariant) Tail-Biting Codes[11][12]

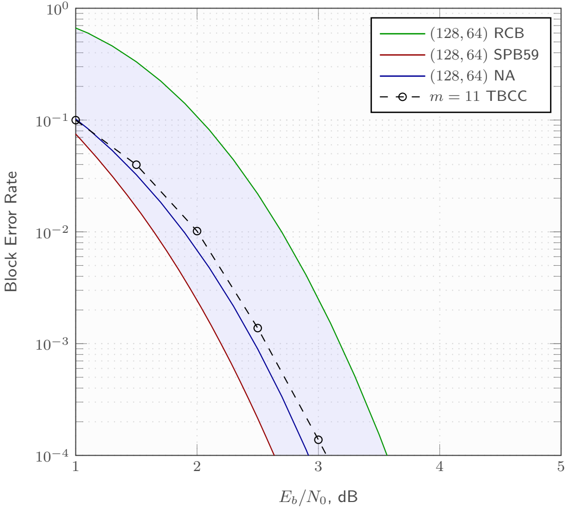
Generators (octal)	m	(n,k)	Minimum Distance
$[515, 677]$	8	$(128, 64)$	12
$[5537, 6131]$	11	$(128, 64)$	14
$[75063, 56711]$	14	$(128, 64)$	16
$[515, 677]$	8	$(256, 128)$	12
$[5537, 6131]$	11	$(256, 128)$	14
$[75063, 56711]$	14	$(256, 128)$	16

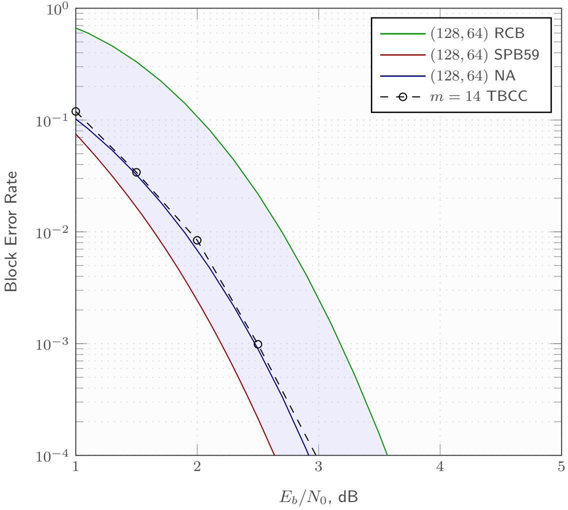
[11] P. Stahl, J. B. Anderson, and R. Johannesson, "Optimal and near-optimal encoders for short and moderate-length tail-biting trellises," *IEEE Trans. Inf. Theory*, 1999

[12] R. Johannesson and K. S. Zigangirov, *Fundamentals of convolutional coding*. John Wiley & Sons, 2015

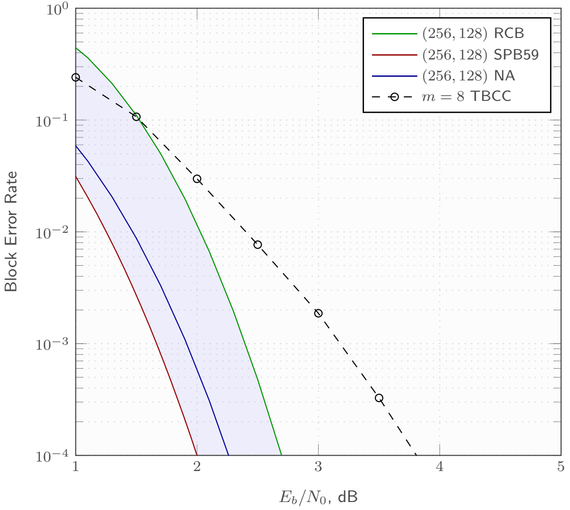


10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$m = 8$ TBCC

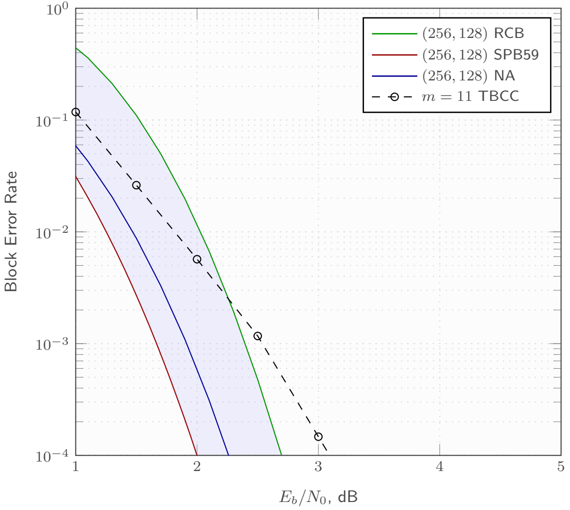

$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$m = 11$ TBCC
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB

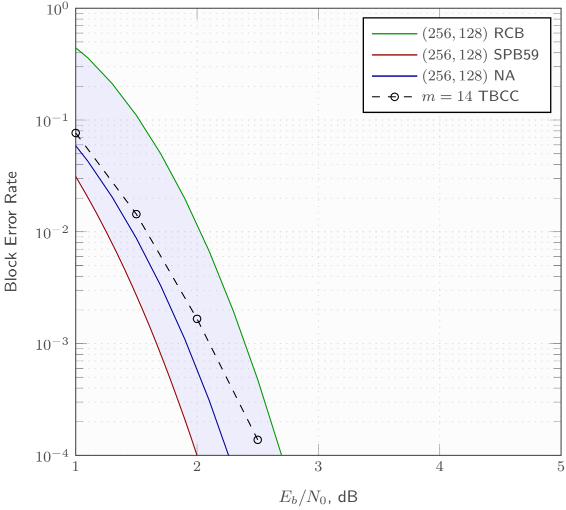


10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$m = 14$ TBCC

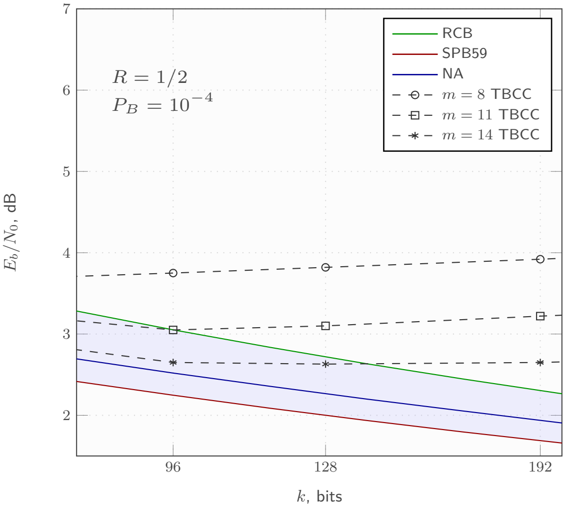


$(256, 128)$ RCB
$(256, 128)$ SPB59
$(256, 128)$ NA
$m = 8$ TBCC
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB


(256, 128) RCB
(256, 128) SPB59
(256, 128) NA
m = 11 TBCC
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB



10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(256, 128)$ RCB
$(256, 128)$ SPB59
$(256, 128)$ NA
$m = 14$ TBCC


$R = 1/2$
$P_B = 10^{-4}$
RCB
SPB59
NA
$m = 8$ TBCC
$m = 11$ TBCC
$m = 14$ TBCC
E_b/N_0, dB
k, bits
7
6
5
4
3
2
96
128
192

Tail-Biting Convolutional Codes
Observations

- Close to optimal at short block lengths ($k \leq 100$ bits)

Tail-Biting Convolutional Codes
Observations

- Close to optimal at short block lengths ($k \leq 100$ bits)
- Efficient decoding via wrap around Viterbi algorithm (incomplete decoding algorithm)

Tail-Biting Convolutional Codes
Observations

- Close to optimal at short block lengths ($k \leq 100$ bits)
- Efficient decoding via wrap around Viterbi algorithm (incomplete decoding algorithm)
- For a fixed memory, performance does not improve with the block length

Tail-Biting Convolutional Codes
Observations

- Close to optimal at short block lengths ($k \leq 100$ bits)
- Efficient decoding via wrap around Viterbi algorithm (incomplete decoding algorithm)
- For a fixed memory, performance does not improve with the block length
- Shall be employed only at the lowest part of the block length spectrum

DLR

Outline

Near Maximum Likelihood Decoding of Linear Block Codes

Ordered Statistics Decoding[15]

- Idea: Use channel reliability measures to build a good subset of $\mathcal{C}$ (list)
- Apply a maximum likelihood search within the list only

[13] A. Valembois and M. Fossorier, "Box and match techniques applied to soft-decision decoding," *IEEE Trans. Inf. Theory*, 2004

[14] Y. Wu and C. Hadjicostis, "Soft-decision decoding using ordered recodings on the most reliable basis," *IEEE Trans. Inf. Theory*, 2007

[15] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, 1995

Near Maximum Likelihood Decoding of Linear Block Codes
Ordered Statistics Decoding[15]

- Idea: Use channel reliability measures to build a good subset of $\mathcal{C}$ (list)
- Apply a maximum likelihood search within the list only
- Can be applied to any linear block code (no specific structure required)
  - No need to sacrifice minimum distance for structure

[13] A. Valembois and M. Fossorier, "Box and match techniques applied to soft-decision decoding," *IEEE Trans. Inf. Theory*, 2004

[14] Y. Wu and C. Hadjicostis, "Soft-decision decoding using ordered recodings on the most reliable basis," *IEEE Trans. Inf. Theory*, 2007

[15] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, 1995

Near Maximum Likelihood Decoding of Linear Block Codes

Ordered Statistics Decoding[15]

- Idea: Use channel reliability measures to build a good subset of $\mathcal{C}$ (list)
- Apply a maximum likelihood search within the list only
- Can be applied to any linear block code (no specific structure required)
  - No need to sacrifice minimum distance for structure
- Several enhancements during the past decade (see e.g. [13][14])

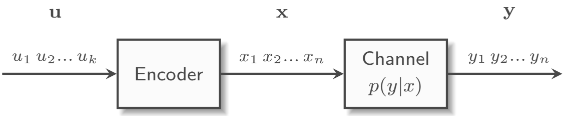
[13] A. Valembois and M. Fossorier, "Box and match techniques applied to soft-decision decoding," *IEEE Trans. Inf. Theory*, 2004

[14] Y. Wu and C. Hadjicostis, "Soft-decision decoding using ordered recodings on the most reliable basis," *IEEE Trans. Inf. Theory*, 2007

[15] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, 1995

Ordered Statistics Decoding

Consider a (n, k) binary linear block code with $k \times n$ generator matrix $\mathbf{G}$



Over the binary input AWGN channel

$$y_i = x_i + n_i$$

Ordered Statistics Decoding (1/3)

- Sort $\mathbf{y}$ in decreasing order of reliability

$$\mathbf{y}' = \pi(\mathbf{y}) \qquad \text{with} \qquad |y'_1| \geq |y'_2| \geq \ldots \geq |y'_n|$$

Ordered Statistics Decoding (1/3)

- Sort $\mathbf{y}$ in decreasing order of reliability

$$\mathbf{y}' = \pi(\mathbf{y}) \qquad \text{with} \qquad |y'_1| \geq |y'_2| \geq \ldots \geq |y'_n|$$

- Permute the columns of $\mathbf{G}$ accordingly: $\mathbf{G}' = \pi(\mathbf{G})$

Ordered Statistics Decoding (1/3)

- Sort $\mathbf{y}$ in decreasing order of reliability

$$\mathbf{y}' = \pi(\mathbf{y}) \qquad \text{with} \qquad |y_1'| \geq |y_2'| \geq \ldots \geq |y_n'|$$

- Permute the columns of $\mathbf{G}$ accordingly: $\mathbf{G}' = \pi(\mathbf{G})$
- Assume next that the first k columns of $\mathbf{G}'$ are linearly independent (information set)

  Put $\mathbf{G}'$ in systematic form by row operations only

$$\mathbf{G}_{\text{sys}} = (\mathbf{I}|\mathbf{P})$$

Ordered Statistics Decoding (1/3)

- Sort $\mathbf{y}$ in decreasing order of reliability

$$\mathbf{y}' = \pi(\mathbf{y}) \qquad \text{with} \qquad |y_1'| \geq |y_2'| \geq \ldots \geq |y_n'|$$

- Permute the columns of $\mathbf{G}$ accordingly: $\mathbf{G}' = \pi(\mathbf{G})$
- Assume next that the first k columns of $\mathbf{G}'$ are linearly independent (information set)

  Put $\mathbf{G}'$ in systematic form by row operations only

$$\mathbf{G}_{\mathsf{sys}} = (\mathbf{I}|\mathbf{P})$$

- Take a bit-by-bit hard decision $\mathbf{u}'$ on the first k elements of $\mathbf{y}'$

Ordered Statistics Decoding (2/3)

- Produce the all the k-bit error vectors with Hamming weight $\leq t$ (t is the *order* of the OSD)

$$
\begin{array}{ll}
\text{weight} - 0 & \begin{array}{lcl} \mathbf{e}_0 & = & (0,0,0,\ldots,0,0) \end{array} \\[1ex]
\text{weight} - 1 & \left\{ \begin{array}{lcl} \mathbf{e}_1 & = & (1,0,0,\ldots,0,0) \\ \mathbf{e}_2 & = & (0,1,0,\ldots,0,0) \\ & \ldots & \\ \mathbf{e}_k & = & (0,0,0,\ldots,0,1) \end{array} \right. \\[1ex]
\text{weight} - 2 & \left\{ \begin{array}{lcl} \mathbf{e}_{k+1} & = & (1,1,0,\ldots,0,0) \\ \mathbf{e}_{k+2} & = & (1,0,1,\ldots,0,0) \\ & \ldots & \\ \mathbf{e}_{k+k(k-1)/2} & = & (0,0,0,\ldots,1,1) \end{array} \right. \\[1ex]
\ldots &
\end{array}
$$

Ordered Statistics Decoding (3/3)

- Test error patterns list $\mathcal{E}$, size $|\mathcal{E}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$
- Build a list of $\mathcal{L} = \left\{ \mathbf{x}'_0\, \mathbf{x}'_1 \ldots \mathbf{x}'_{|\mathcal{E}|-1} \right\}$ of $|\mathcal{E}|$ codewords by

$$
\begin{aligned}
\mathbf{u}'_0 &= \mathbf{u}' + \mathbf{e}_0 \\
\mathbf{u}'_1 &= \mathbf{u}' + \mathbf{e}_1 \\
\mathbf{u}'_2 &= \mathbf{u}' + \mathbf{e}_2 \\
&\ldots \\
\mathbf{u}'_{|\mathcal{E}|-1} &= \mathbf{u}' + \mathbf{e}_{|\mathcal{E}|-1}
\end{aligned}
$$

Ordered Statistics Decoding (3/3)

- Test error patterns list $\mathcal{E}$, size $|\mathcal{E}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$
- Build a list of $\mathcal{L} = \left\{ \mathbf{x}'_0\, \mathbf{x}'_1 \ldots \mathbf{x}'_{|\mathcal{E}|-1} \right\}$ of $|\mathcal{E}|$ codewords by

$$
\begin{aligned}
\mathbf{c}'_0 &= \mathbf{u}'_0 \mathbf{G}_{\mathsf{sys}} \\
\mathbf{c}'_1 &= \mathbf{u}'_1 \mathbf{G}_{\mathsf{sys}} \\
\mathbf{c}'_2 &= \mathbf{u}'_2 \mathbf{G}_{\mathsf{sys}} \\
&\ldots \\
\mathbf{c}'_{|\mathcal{E}|-1} &= \mathbf{u}'_{|\mathcal{E}|-1} \mathbf{G}_{\mathsf{sys}}
\end{aligned}
$$

Ordered Statistics Decoding (3/3)

- Test error patterns list $\mathcal{E}$, size $|\mathcal{E}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$
- Build a list of $\mathcal{L} = \left\{\mathbf{x}'_0\, \mathbf{x}'_1 \ldots \mathbf{x}'_{|\mathcal{E}|-1}\right\}$ of $|\mathcal{E}|$ codewords by

$$
\begin{aligned}
\mathbf{x}'_0 &= 1 - 2\mathbf{c}'_0 \\
\mathbf{x}'_1 &= 1 - 2\mathbf{c}'_1 \\
\mathbf{x}'_2 &= 1 - 2\mathbf{c}'_2 \\
&\ldots \\
\mathbf{x}'_{|\mathcal{E}|-1} &= 1 - 2\mathbf{c}'_{|\mathcal{E}|-1}
\end{aligned}
$$

Ordered Statistics Decoding (3/3)

- Test error patterns list $\mathcal{E}$, size $|\mathcal{E}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$
- Build a list of $\mathcal{L} = \left\{\mathbf{x}'_0\, \mathbf{x}'_1 \ldots \mathbf{x}'_{|\mathcal{E}|-1}\right\}$ of $|\mathcal{E}|$ codewords by

$$
\begin{aligned}
\mathbf{x}'_0 &= 1 - 2\mathbf{c}'_0 \\
\mathbf{x}'_1 &= 1 - 2\mathbf{c}'_1 \\
\mathbf{x}'_2 &= 1 - 2\mathbf{c}'_2 \\
&\ldots \\
\mathbf{x}'_{|\mathcal{E}|-1} &= 1 - 2\mathbf{c}'_{|\mathcal{E}|-1}
\end{aligned}
$$

- Perform a maximum likelihood search in the list $\mathcal{L}$

$$\hat{\mathbf{x}}' = \arg\max_{\mathbf{x}' \in \mathcal{L}} p(\mathbf{y}'|\mathbf{x}') \qquad \rightarrow \qquad \hat{\mathbf{x}} = \pi^{-1}\left(\hat{\mathbf{x}}'\right)$$

Ordered Statistics Decoding
Example

- $(6, 3)$ shortened Hamming code

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

- Channel output $\mathbf{y} = (+0.2,\ +1.2,\ -0.2,\ +0.6,\ -0.1,\ +1.0)$
- Ordered observation $\mathbf{y}' = (y_2, y_6, y_4, y_1, y_3, y_5)$
- Permuted generator matrix

$$\mathbf{G}' = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \rightarrow \mathbf{G}_{\text{sys}} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Ordered Statistics Decoding
Example

- We select the OSD parameter $t = 1$

$$\mathbf{e}_0 = (0, 0, 0) \qquad \mathbf{e}_2 = (0, 1, 0)$$
$$\mathbf{e}_1 = (1, 0, 0) \qquad \mathbf{e}_3 = (0, 0, 1)$$

- Given the hard decision $\mathbf{u}' = (0, 0, 0)$ on $\mathbf{y}'$

$$\begin{aligned}
\mathbf{u}'_0 &= (0, 0, 0) &&\rightarrow \mathbf{c}'_0 = (0, 0, 0, 0, 0, 0) &&\rightarrow \mathbf{x}'_0 = (+1, +1, +1, +1, +1, +1)\\
\mathbf{u}'_1 &= (1, 0, 0) &&\rightarrow \mathbf{c}'_1 = (1, 0, 0, 1, 1, 0) &&\rightarrow \mathbf{x}'_1 = (-1, +1, +1, -1, -1, +1)\\
\mathbf{u}'_2 &= (0, 1, 0) &&\rightarrow \mathbf{c}'_2 = (0, 1, 0, 0, 1, 1) &&\rightarrow \mathbf{x}'_2 = (+1, -1, +1, +1, -1, -1)\\
\mathbf{u}'_3 &= (0, 0, 1) &&\rightarrow \mathbf{c}'_3 = (0, 0, 1, 1, 0, 1) &&\rightarrow \mathbf{x}'_3 = (+1, +1, -1, -1, +1, -1)
\end{aligned}$$

- Check that $\hat{\mathbf{x}}' = \mathbf{x}'_0$ which leads to $\hat{\mathbf{c}} = (0, 0, 0, 0, 0, 0)$

Ordered Statistics Decoding
Complexity

- Is proportional to the list size

$$|\mathcal{L}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$$

- The list size explodes quickly with t
- Example: $(128, 64)$ code

$$\begin{aligned}
t = 1 &: |\mathcal{L}| = 65 \\
t = 2 &: |\mathcal{L}| = 2081 \\
t = 3 &: |\mathcal{L}| = 43745 \\
t = 4 &: |\mathcal{L}| = 679121
\end{aligned}$$

Ordered Statistics Decoding
Complexity

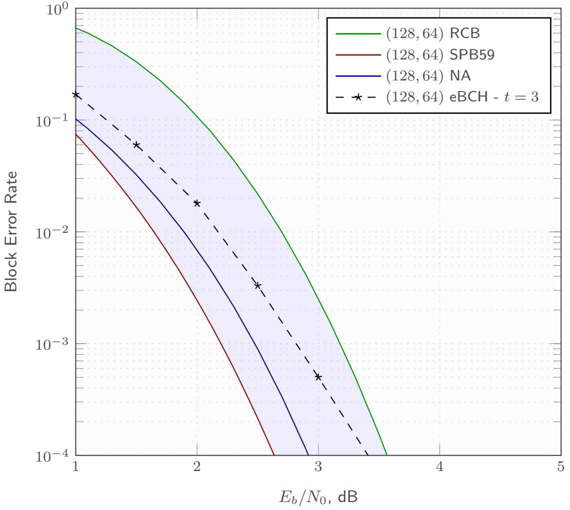
- The performance approaches the one of a maximum likelihood decoder with sufficiently large t

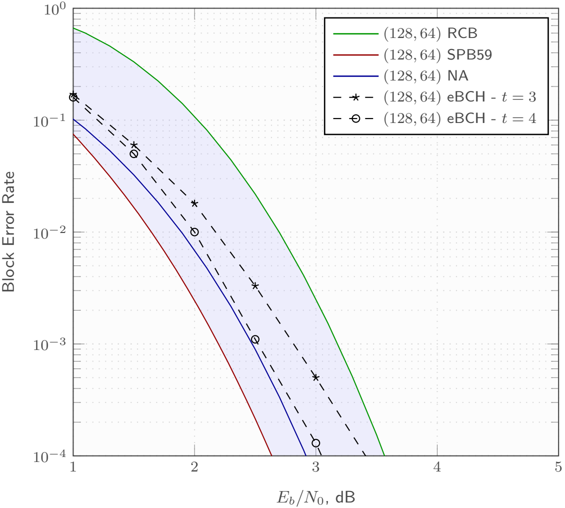
Ordered Statistics Decoding
Complexity

- The performance approaches the one of a maximum likelihood decoder with sufficiently large t
- The longer the code, the higher t needs to be to get close to maximum likelihood decoding

Ordered Statistics Decoding
Complexity

- The performance approaches the one of a maximum likelihood decoder with sufficiently large t
- The longer the code, the higher t needs to be to get close to maximum likelihood decoding
  - $(24, 12)$ Golay code: $t = 2$
  - $(128, 64)$ extended BCH code: $t = 4$


(128, 64) RCB
(128, 64) SPB59
(128, 64) NA
(128, 64) eBCH - $t = 3$
Block Error Rate
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB



10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$(128, 64)$ eBCH - $t = 3$
$(128, 64)$ eBCH - $t = 4$

Ordered Statistics Decoding
Complexity

- For a given t, early stopping can be used to avoid testing all the

$$|\mathcal{L}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$$

candidates

Ordered Statistics Decoding
Complexity

- For a given t, early stopping can be used to avoid testing all the

$$|\mathcal{L}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$$

candidates

- Define a threshold $d > 0$

Ordered Statistics Decoding
Complexity

- For a given t, early stopping can be used to avoid testing all the

$$|\mathcal{L}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$$

  candidates
- Define a threshold $d > 0$
- While building the list, if one finds a codeword $\mathbf{x}'$ s.t. $\|\mathbf{y} - \mathbf{x}'\| < d$ then decoding stops and $\hat{\mathbf{x}}' = \mathbf{x}'$

Ordered Statistics Decoding
Complexity

- For a given t, early stopping can be used to avoid testing all the

$$|\mathcal{L}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$$

  candidates
- Define a threshold $d > 0$
- While building the list, if one finds a codeword $\mathbf{x}'$ s.t. $\|\mathbf{y} - \mathbf{x}'\| < d$ then decoding stops and $\hat{\mathbf{x}}' = \mathbf{x}'$
- The threshold is typically related to the code minimum distance

Ordered Statistics Decoding

Complexity

- For a given t, early stopping can be used to avoid testing all the

$$|\mathcal{L}| = \sum_{\ell=0}^{t} \binom{k}{\ell}$$

  candidates
- Define a threshold $d > 0$
- While building the list, if one finds a codeword $\mathbf{x}'$ s.t. $\|\mathbf{y} - \mathbf{x}'\| < d$ then decoding stops and $\hat{\mathbf{x}}' = \mathbf{x}'$
- The threshold is typically related to the code minimum distance
- At high signal-to-noise ratios, the number of tested codewords drops by (several) orders of magnitude

Ordered Statistics Decoding

- OSD can be applied to any code: Also to turbo-like codes!

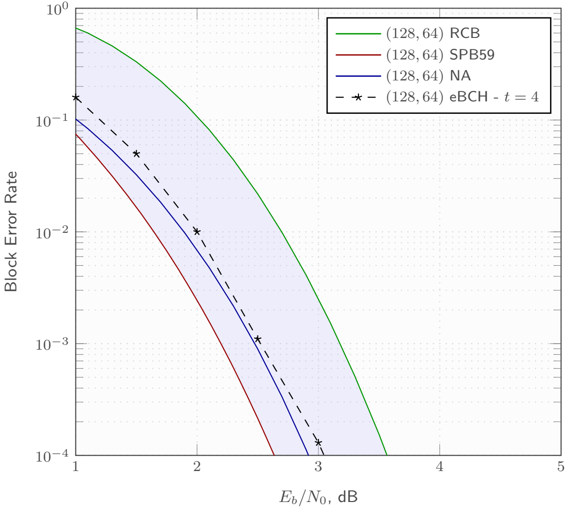
[16] M. P. C. Fossorier, “Iterative reliability-based decoding of low-density parity check codes,” *IEEE J. Sel. Areas Commun.*, 2001

Ordered Statistics Decoding

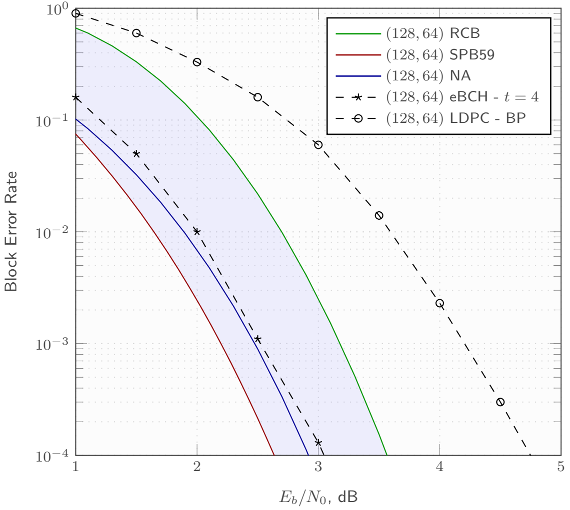
- OSD can be applied to any code: Also to turbo-like codes!
- First decoding attempt: Iterative
- Then, OSD only if iterative decoding fails

Hybrid iterative-OSD[16]

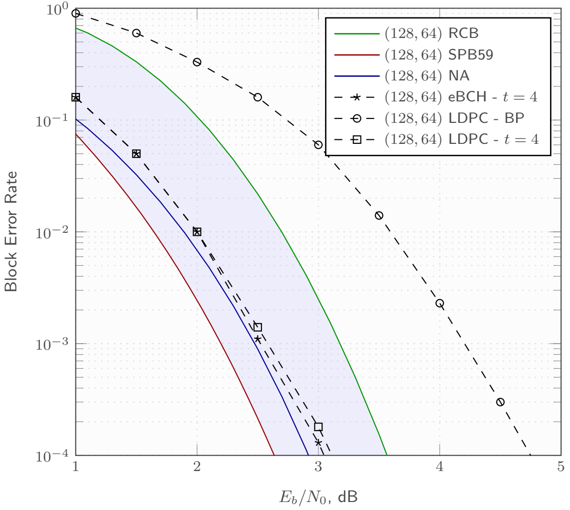
[16] M. P. C. Fossorier, "Iterative reliability-based decoding of low-density parity check codes," *IEEE J. Sel. Areas Commun.*, 2001



10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$(128, 64)$ eBCH - $t = 4$



(128, 64) RCB
(128, 64) SPB59
(128, 64) NA
(128, 64) eBCH - t = 4
(128, 64) LDPC - BP
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB


10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
(128, 64) RCB
(128, 64) SPB59
(128, 64) NA
(128, 64) eBCH - $t = 4$
(128, 64) LDPC - BP
(128, 64) LDPC - $t = 4$

Ordered Statistics Decoding
Observations

- Very promising in the short block length regime

Ordered Statistics Decoding
Observations

- Very promising in the short block length regime
- Complexity does not scale yet favorably with n (especially due to the need of increasing the OSD order), but

Ordered Statistics Decoding
Observations

- Very promising in the short block length regime
- Complexity does not scale yet favorably with n (especially due to the need of increasing the OSD order), but

  …research is on-going on the topic, and several enhancements can be used to extend the (block length) range of interest

Outline

- Preliminaries
- Efficient Short Classical Codes
- Efficient Short Modern Codes
  - Turbo Codes
  - Binary Low-Density Parity-Check Codes
  - Non-Binary Low-Density Parity-Check Codes
  - Polar Codes
- Two Case Studies
- Beyond Error Correction
- Conclusions

Outline

- Preliminaries
- Efficient Short Classical Codes
- Efficient Short Modern Codes
  - Turbo Codes
  - Binary Low-Density Parity-Check Codes
  - Non-Binary Low-Density Parity-Check Codes
  - Polar Codes
- Two Case Studies
- Beyond Error Correction
- Conclusions

Parallel Concatenated Convolutional Codes

- Turbo codes with 16-states component codes provide the excellent trade-off between minimum distance and decoding threshold[17][18]
- Tail-biting component codes reduce termination overhead[19][20]
- Interleaver design is crucial

FB/FFW Polynomial (Octal)	$(E_b/N_0)^\star$, $R = 1/2$	Notes
27/37	0.56 dB	16-states
23/35	0.62 dB	16-states
15/13	0.70 dB	8-states

[17] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. ICC*, 1993

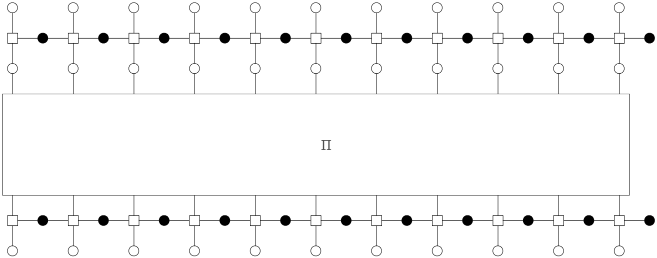
[18] H. El-Gamal and J. Hammons, AR., "Analyzing the turbo decoder using the gaussian approximation," *IEEE Trans. Inf. Theory*, 2001

[19] C. Weiss, C. Bettstetter, and S. Riedel, "Code construction and decoding of parallel concatenated tail-biting codes," *IEEE Trans. Inf. Theory*, 2001

[20] T. Jerkovits and B. Matuz, "Turbo code design for short blocks," in *Proc. 7th Advanced Satellite Mobile Systems Conference*, 2016

Parallel Concatenated Convolutional Codes
Factor Graph

Turbo codes factor graphs[21] are characterized by large girth



[21] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University, 1996

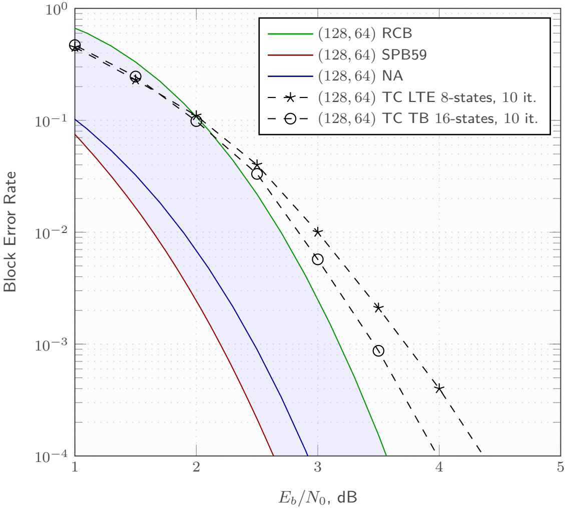
Parallel Concatenated Convolutional Codes
Interleavers

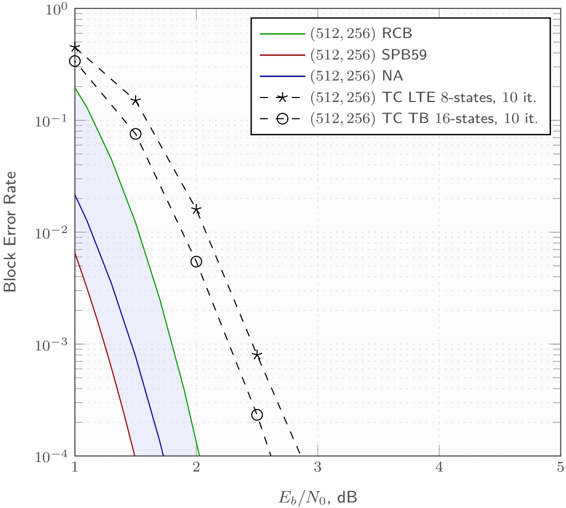
- The interleaver is the main responsible for large girth and spread (essential for large d_{min})
- Yet, $d_{\text{min}} = \mathcal{O}(\log n)$
- Among the best-known constructions
  - Dithered-Relative-Prime (DRP)[22]
  - Quadratic permutation polynomial (QPP) - LTE [23]

---

[22] S. Crozier and P. Guinand, "High-performance low-memory interleaver banks for turbo-codes," in *Proc. IEEE VTC*, 2001

[23] O. Takeshita, "On maximum contention-free interleavers and permutation polynomials over integer rings," *IEEE Trans. Inf. Theory*, 2006


(128, 64) RCB
(128, 64) SPB59
(128, 64) NA
(128, 64) TC LTE 8-states, 10 it.
(128, 64) TC TB 16-states, 10 it.
Block Error Rate
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB


$(512, 256)$ RCB
$(512, 256)$ SPB59
$(512, 256)$ NA
$(512, 256)$ TC LTE 8-states, 10 it.
$(512, 256)$ TC TB 16-states, 10 it.
Block Error Rate
E_b/N_0, dB
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5

Turbo Codes
Observations

- Performance within 0.5 dB from RCB at moderate error rates

Turbo Codes
Observations

- Performance within 0.5 dB from RCB at moderate error rates
- Decoding can be partially parallelized

Turbo Codes
Observations

- Performance within 0.5 dB from RCB at moderate error rates
- Decoding can be partially parallelized
- 16-states tail-biting component codes: Good compromise between decoding complexity and performance

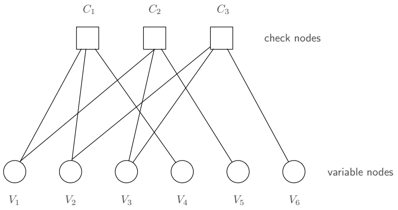
Outline

Low-Density Parity-Check Codes

Graphical Representation of the Parity-Check Matrix

- Low-density[24] $\mathbf{H}$ matrix imposing a set of $n-k$ constraints
- Graphical representation via Tanner graphs[25]
  - Codeword bits $\equiv$ variable nodes (VNs)
  - Check equations $\equiv$ check nodes (CNs)

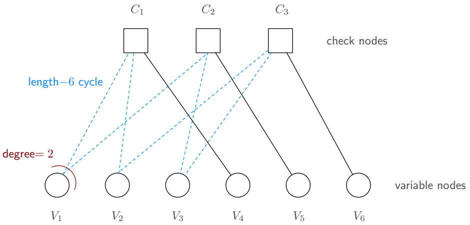
$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$



[24] R. Gallager, *Low-density parity-check codes*, 1963
[25] M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, 1981

Low-Density Parity-Check Codes
Graphical Representation of the Parity-Check Matrix

- Graphical representation via Tanner graphs (cont'd)

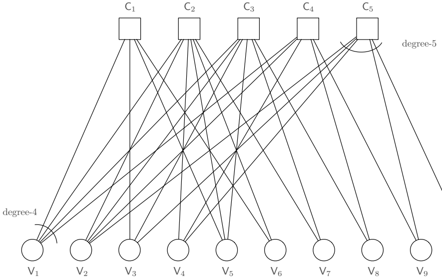


Low-Density Parity-Check Codes
Regular LDPC Codes Ensemble

- The $\mathscr{C}^{n}_{(d_v,d_c)}$ regular unstructured LDPC ensemble is the set of binary linear block codes defined by a Tanner graph with
  - n variable nodes, having degree d_v
  - nd_v/d_c check nodes with degree d_c
  - Edge permutation picked uniformly at random

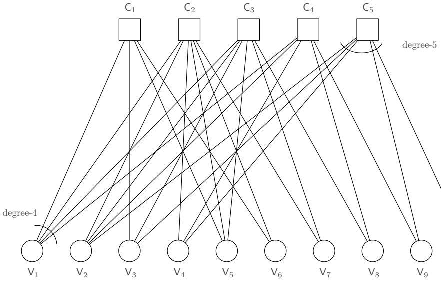
Low-Density Parity-Check Codes
Irregular LDPC Codes Ensemble

- An irregular LDPC code graph is often characterized by the distributions of degrees of variable and check nodes



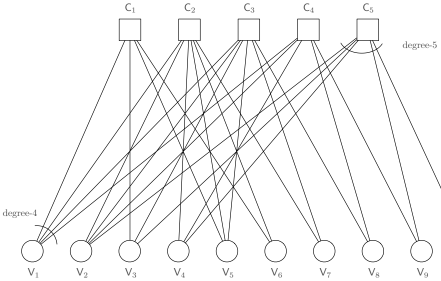
Low-Density Parity-Check Codes
Irregular LDPC Codes Ensemble

- An irregular LDPC code graph is often characterized by the distributions of degrees of variable and check nodes
- The node-oriented variable node degree distribution is denoted by $\mathbf{\Lambda} = \{\Lambda_i\}$, where Λ_i is the fraction of VNs with degree i



Low-Density Parity-Check Codes
Irregular LDPC Codes Ensemble

- An irregular LDPC code graph is often characterized by the distributions of degrees of variable and check nodes
- The node-oriented check node degree distribution is denoted by $\mathbf{P} = \{P_j\}$, where P_j is the fraction of CNs with degree j

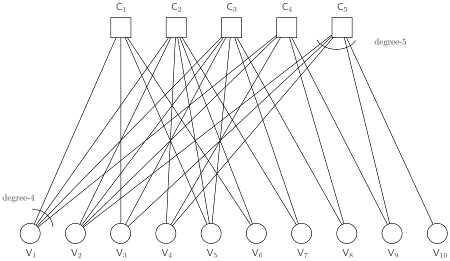


Low-Density Parity-Check Codes
Irregular LDPC Codes Ensemble

- The edge-oriented variable node degree distribution is denoted by $\boldsymbol{\lambda} = \{\lambda_i\}$, where λ_i is the fraction of edges connected to VNs with degree i
- The edge-oriented check node degree distribution is denoted by $\boldsymbol{\rho} = \{\rho_j\}$, where ρ_j is the fraction of edges connected to CNs with degree j

$$\lambda_i = \frac{i\Lambda_i}{\sum_l l\Lambda_l}$$

$$\rho_i = \frac{i\mathsf{P}_i}{\sum_l l\mathsf{P}_l}$$



Low-Density Parity-Check Codes
Irregular LDPC Codes Ensemble

- Degree distributions are often given in polynomial form

$$\Lambda(x) = \sum_i \Lambda_i x^i \qquad \text{and} \qquad \mathsf{P}(x) = \sum_j \mathsf{P}_j x^j$$

$$\lambda(x) = \sum_i \lambda_i x^{i-1} \qquad \text{and} \qquad \rho(x) = \sum_j \rho_j x^{j-1}$$

- We have that

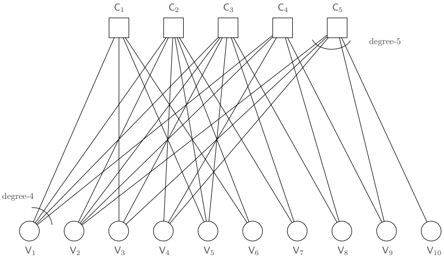
$$\lambda(x) = \frac{\Lambda'(x)}{\Lambda'(1)} \qquad \text{and} \qquad \rho(x) = \frac{\mathsf{P}'(x)}{\mathsf{P}'(1)}$$

with

$$\Lambda'(x) = \frac{\mathrm{d}\Lambda(x)}{\mathrm{d}x} \qquad \text{and} \qquad \mathsf{P}'(x) = \frac{\mathrm{d}\mathsf{P}(x)}{\mathrm{d}x}.$$

Low-Density Parity-Check Codes
Irregular LDPC Codes Ensemble

- The $\mathscr{C}^n_{(\mathbf{\Lambda},\mathbf{P})}$ irregular unstructured LDPC ensemble is the set of binary linear block codes defined by a Tanner graph with
  - n variable nodes whose degrees are distributed according to $\mathbf{\Lambda}$
  - m check nodes with degrees are according to $\mathbf{P}$
  - Edge permutation picked uniformly at random



LDPC Codes: Unstructured vs. Structured Ensembles
Basics

- Unstructured graphs are easy to analyze, but
  - Are difficult to implement in actual (hardware) decoders
  - Allow limited control on edge connectivity properties
- A refined definition leads to structured LDPC ensembles
  - Repeat-Accumulate-like Codes[26]
  - Protograph Codes[27]
  - Multi-Edge-Type Codes[28]

---

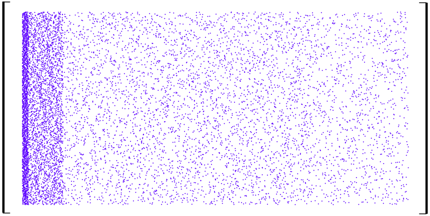
[26] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proc. IEEE Int. Symp. Turbo Codes and Related Topics*, 2000

[27] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," IPN Progress Report, 2003

[28] T. Richardson and R. Urbanke, "Multi-edge type LDPC codes," 2004, unpublished

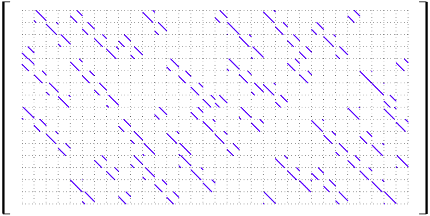
LDPC Codes: Structured Ensembles

$\mathbf{H} =$



Unstructured LDPC Code

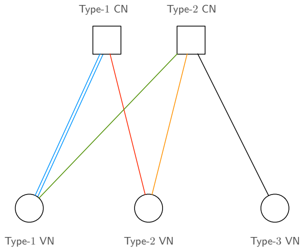
$\mathbf{H} =$



Structured LDPC Code

LDPC Codes: Structured Ensembles
Protograph Codes

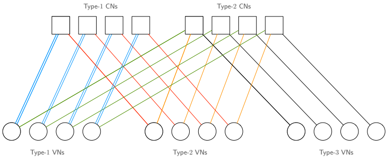
- Protograph: small Tanner graph used as template to build the code graph
- Equivalent representation: base matrix



$$\mathbf{B} = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

LDPC Codes: Structured Ensembles
Protograph Codes

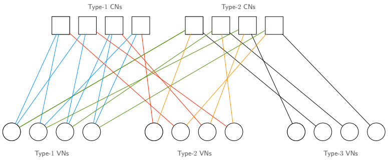
- A protograph can be used to construct a larger Tanner graph by a copy & permute procedure
- The larger Tanner graph defines the code
- First step: Protograph is copied Q times



$$\mathbf{B}' = \begin{pmatrix} 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

LDPC Codes: Structured Ensembles
Protograph Codes

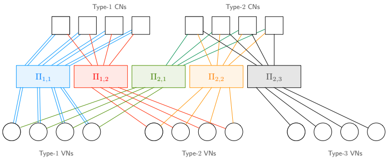
- Second step: Permute edges among the replicas
- Permutations shall avoid parallel edges



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

LDPC Codes: Structured Ensembles
Protograph Codes

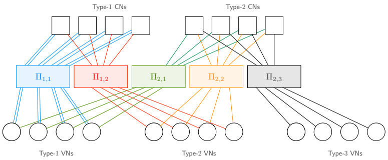
- Second step: Permute edges among the replicas
- Permutations shall avoid parallel edges



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

LDPC Codes: Structured Ensembles
Protograph Codes

- Second step: Permute edges among the replicas
- Permutations shall avoid parallel edges



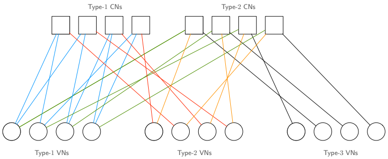
$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0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

- A protograph defines structured LDPC code ensemble: The iterative decoding threshold and distance properties follow from the protograph

LDPC Codes: Structured Ensembles

Protograph Codes

- Depending on code length, the expansion can be done in more steps
- In each step, girth optimization techniques[29] are used
- The final expansion is usually performed by means of circulant permutation matrices (quasi-cyclic code)[30]



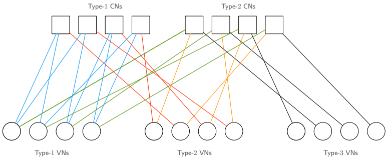
$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0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

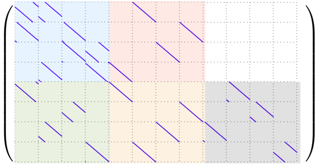
[29] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, 2005

[30] W. Ryan and S. Lin, *Channel codes – Classical and modern*. Cambridge Univ. Press, 2009

LDPC Codes: Structured Ensembles
Protograph Codes

- Depending on code length, the expansion can be done in more steps
- In each step, girth optimization techniques[29] are used
- The final expansion is usually performed by means of circulant permutation matrices (quasi-cyclic code)[30]



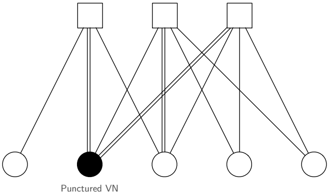
$\mathbf{H} =$ 

[29] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, 2005

[30] W. Ryan and S. Lin, *Channel codes – Classical and modern*. Cambridge Univ. Press, 2009

LDPC Codes: Structured Ensembles
Protograph Codes

- Punctured (state) and degree-1 variable nodes are allowed
- Near-capacity thresholds can be achieved with lower average degrees than unstructured LDPC codes $\rightarrow$ larger girth
- Example: Accumulate-Repeat-3-Accumlate (AR3A), $R = 1/2$, $(E_b/N_0)^\star = 0.475$ dB, only 0.3 dB from Shannon limit



LDPC Codes: Structured Ensembles
Protograph Codes

- Allow controlling in finer way the edge connectivity[31]
- Example: consider the based matrices

$$\mathbf{B}_1 = \begin{pmatrix} 1 & 2 & 1 & 1 & 0 \\ 2 & 1 & 1 & 1 & 0 \\ 1 & 2 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{B}_2 = \begin{pmatrix} 1 & 3 & 1 & 0 & 0 \\ 2 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

They describe sub-ensembles of the unstructured ensemble defined by

$$\Lambda(x) = 0.2x + 0.4x^2 + 0.2x^4 + 0.2x^5, \qquad P(x) = 0.333x^4 + 0.667x^5$$

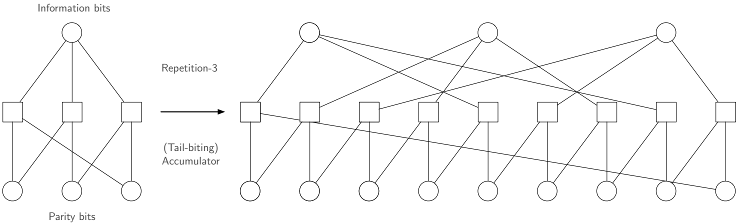
with degree-5 VNs punctured

However, $(E_b/N_0)^\star_1 = 0.475$ dB while $(E_b/N_0)^\star_2 = +\infty$ dB

[31] G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," in *Proc. IEEE Global Telecommun. Conf.*, 2007

Protograph Ensembles: Accumulator-based
Repeat-Accumulate

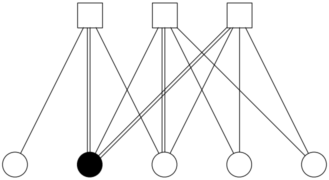
- Accumulator-based LDPC codes admit simple protograph representation
- Very low decoding thresholds with low average degrees → large girth
- Repeat-accumulate (RA) codes[32]



[32] D. Divsalar, S. Dolinar, J. Thorpe, and C. Jones, "Constructing LDPC codes from simple loop-free encoding modules," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2005

Protograph Ensembles: Accumulator-based
Example: AR3A Protograph

Accumulate-Repeat-3-Accumulate (AR3A) Protograph[33]



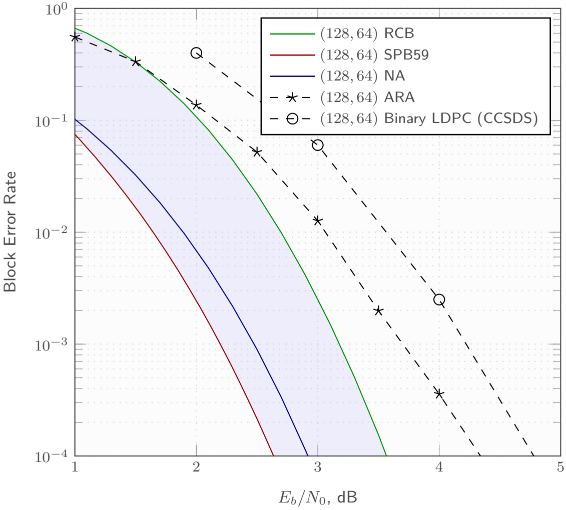
d_{min}	$(E_b/N_0)^{\star}$	Encoding
$\mathcal{O}(\log n)$	0.475 dB	$\mathcal{O}(n)$

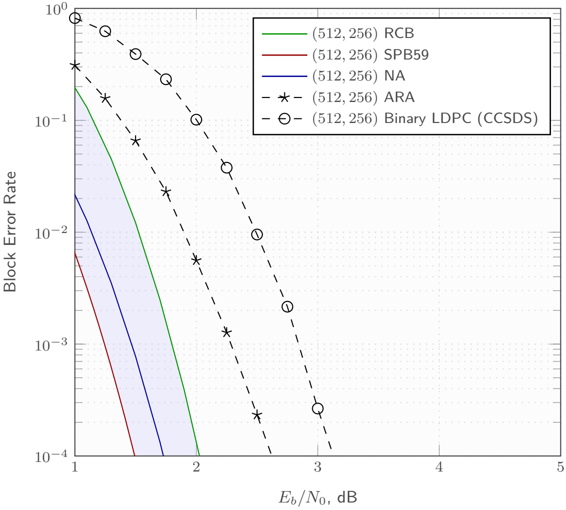
[33] D. Divsalar, S. Dolinar, C. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE JSAC*, 2009

Protograph Ensembles: Accumulator-based

AR3A Protograph Thresholds

R	$(E_b/N_0)^\star$	Shannon Limit
$7/8$	3.139 dB	2.845 dB
$4/5$	2.283 dB	2.040 dB
$3/4$	1.845 dB	1.626 dB
$2/3$	1.295 dB	1.059 dB
$1/2$	0.475 dB	0.187 dB
$1/3$	-0.050 dB	-0.495 dB
$1/4$	-0.554 dB	-0.794 dB
$1/5$	-0.755 dB	-0.964 dB
$1/6$	-0.791 dB	-1.073 dB
$1/8$	-0.962 dB	-1.204 dB


$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$(128, 64)$ ARA
$(128, 64)$ Binary LDPC (CCSDS)
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB



10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(512, 256)$ RCB
$(512, 256)$ SPB59
$(512, 256)$ NA
$(512, 256)$ ARA
$(512, 256)$ Binary LDPC (CCSDS)

Protograph Ensembles: Raptor-like

- Serial concatenation of a high-rate protograph-based outer LDPC code, and a protograph-based LT code[34]

$$\mathbf{B} = \left(\begin{array}{c|c} \mathbf{B}_{\mathsf{o}} & \mathbf{0} \\ \hline \multicolumn{2}{c}{\mathbf{B}_{\mathsf{LT}}} \end{array} \right)$$

[34] T.-Y. Chen, K. Vakilinia, D. Divsalar, and R. D. Wesel, "Protograph-based raptor-like LDPC codes," 2014. [Online]. Available: http://arxiv.org/abs/1403.2111

Protograph Ensembles: Raptor-like

- Serial concatenation of a high-rate protograph-based outer LDPC code, and a protograph-based LT code[34]

$$\mathbf{B} = \left(\begin{array}{c|c} \mathbf{B}_{\mathsf{o}} & \mathbf{0} \\ \hline \multicolumn{2}{c}{\mathbf{B}_{\mathsf{LT}}} \end{array} \right)$$

- Although the construction targets short block lengths, the outer code parity-check matrix density prevents from obtaining large girths at very short block lengths

[34] T.-Y. Chen, K. Vakilinia, D. Divsalar, and R. D. Wesel, "Protograph-based raptor-like LDPC codes," 2014. [Online]. Available: http://arxiv.org/abs/1403.2111

Protograph Ensembles: Raptor-like

Large flexibility of rates, with thresholds within 0.5 dB from the Shannon limit

$$\mathbf{B} = \begin{pmatrix}
2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

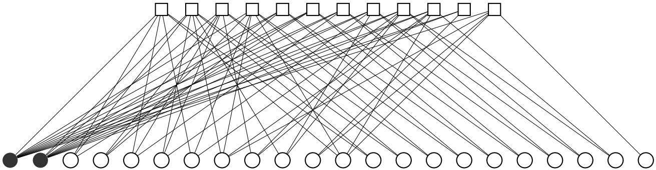
Protograph Ensembles: Raptor-like

Large flexibility of rates, with thresholds within 0.5 dB from the Shannon limit

R	$(E_b/N_0)^\star$	Shannon Limit
$6/7$	3.077 dB	2.625 dB
$6/8$	1.956 dB	1.626 dB
$6/9$	1.392 dB	1.059 dB
$6/10$	1.078 dB	0.679 dB
$6/11$	0.798 dB	0.401 dB
$6/12$	0.484 dB	0.187 dB
$6/13$	0.338 dB	0.018 dB
$6/14$	0.144 dB	-0.122 dB
$6/15$	0.072 dB	-0.238 dB
$6/16$	0.030 dB	-0.337 dB
$6/17$	-0.024 dB	-0.422 dB
$6/18$	-0.150 dB	-0.495 dB

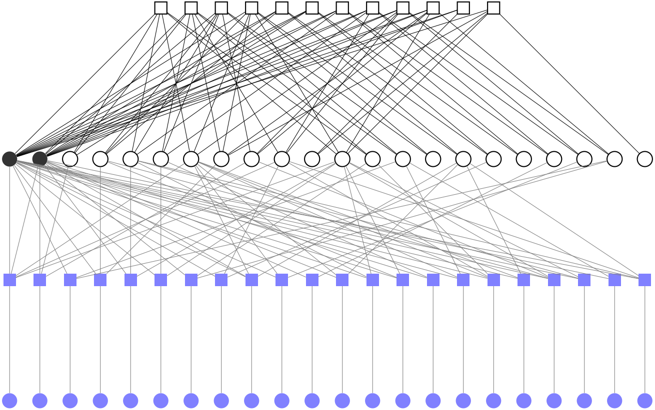
Protograph Ensembles: Raptor-like

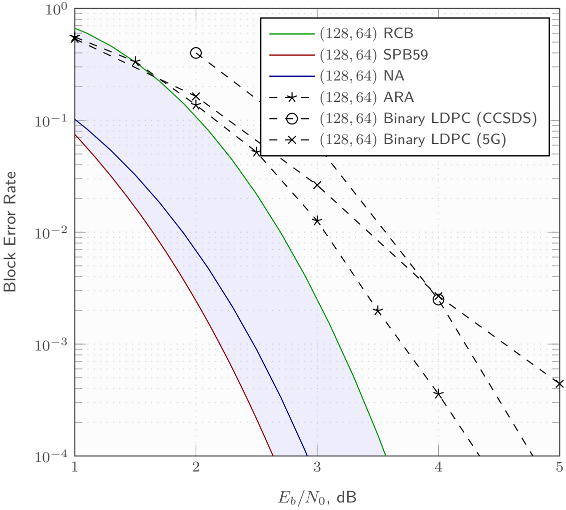
- 5G proposal (enhanced mobile broadband) by Qualcomm: Raptor-like



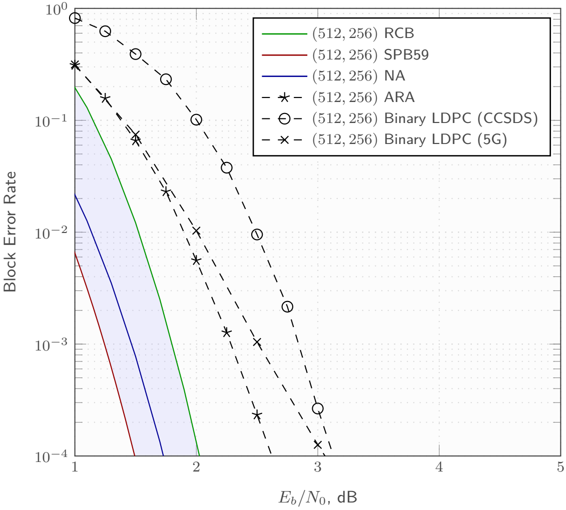
Protograph Ensembles: Raptor-like

- 5G proposal (enhanced mobile broadband) by Qualcomm: Raptor-like





10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$(128, 64)$ ARA
$(128, 64)$ Binary LDPC (CCSDS)
$(128, 64)$ Binary LDPC (5G)



$(512, 256)$ RCB
$(512, 256)$ SPB59
$(512, 256)$ NA
$(512, 256)$ ARA
$(512, 256)$ Binary LDPC (CCSDS)
$(512, 256)$ Binary LDPC (5G)
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB

Binary Low-Density Parity-Check Codes
Observations

- Performance within 1 dB from RCB at short block lengths

Binary Low-Density Parity-Check Codes
Observations

- Performance within 1 dB from RCB at short block lengths
- Protograph construction fundamental to achieve good performance with practical decoders

Binary Low-Density Parity-Check Codes
Observations

- Performance within 1 dB from RCB at short block lengths
- Protograph construction fundamental to achieve good performance with practical decoders
- Depending on the code design, strong error detection capability

Outline

- Preliminaries
- Efficient Short Classical Codes
- Efficient Short Modern Codes
  - Turbo Codes
  - Binary Low-Density Parity-Check Codes
  - Non-Binary Low-Density Parity-Check Codes
  - Polar Codes
- Two Case Studies
- Beyond Error Correction
- Conclusions

Non-Binary LDPC Codes
Basics

- Defined[35] by $M \times N$ sparse parity-check matrix $\mathbf{H}$ via the equation

$$\mathbf{c}\mathbf{H}^T = \mathbf{0}$$

  where $\mathbf{c} \in \mathbb{F}_q^n$ and the elements of $\mathbf{H}$ belong to $\mathbb{F}_q$, $q > 2$
- We are mainly interested in $q = 2^p$
- Example: parity-check matrix over $\mathbb{F}_{16}$

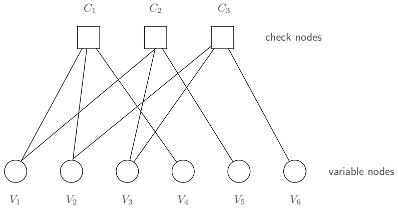
$$\mathbf{H} = \begin{bmatrix} \alpha^3 & \alpha^7 & \alpha^3 & 0 \\ \alpha^0 & 0 & \alpha^{12} & \alpha^{11} \end{bmatrix}$$

[35] M. Davey and D. MacKay, "Low density parity check codes over GF(q)," *IEEE Commun. Lett.*, 1998

Non-Binary LDPC Codes
Basics

- Codeword symbols $\equiv$ variable nodes (VNs)
- Non-Binary Check equations $\equiv$ check nodes (CNs)

$$\mathbf{H} = \begin{pmatrix} \alpha^2 & \alpha^0 & 0 & \alpha^7 & 0 & 0 \\ \alpha^5 & 0 & \alpha^{11} & 0 & \alpha^0 & 0 \\ 0 & \alpha^2 & \alpha^3 & 0 & 0 & \alpha^{10} \end{pmatrix}$$



Non-Binary LDPC Codes
Basics

- Binary image of the code: in the codeword, replace each symbol in $\mathbb{F}_q$ with its length-p, $p = \log_2 q$, binary representation
- Next, we will always refer to the binary block length $n = Np$ bits

Non-Binary LDPC Codes
Belief Propagation Decoding

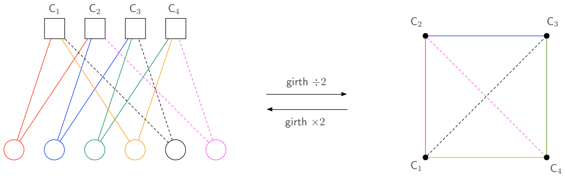
- Messages are vectors of q probabilities
- Convolution of messages at the check nodes
- Decoding complexity scales as quadratically in q
- It can be reduced to $\mathcal{O}(q \log_2 q)$ by performing the CN operation via fast Fourier transform[36]

[36] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2003

Non-Binary LDPC Codes

Ultra-Sparse Ensembles: Cycle LDPC Codes

- Non-Binary LDPC codes constructed on relatively-large fields ($q \geq 64$) perform particularly well when the graph is ultra-sparse[37]
- Constant variable node degree $d_v = 2$, uniform check node degree
- Ultra-sparse LDPC Codes are actually instances of cycle codes[38]



Tanner graph perspective

Cycle (circuit) code perspective

[37] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular $(2, d_c)$-LDPC codes over GF(q) using their binary images," *IEEE Trans. Commun.*, 2008

[38] S. Hakimi and J. Bredeson, "Graph theoretic error-correcting codes," *IEEE Trans. Inf. Theory*, 1968

Cycle LDPC Codes
Distance Properties

- Cycle code ensemble: $d_{\mathsf{min}} = \mathcal{O}(\log n)$
- Beyond girth optimization, the minimum distance of the binary image of the code can be kept sufficiently large by a proper choice of the parity-check matrix coefficients
- Choosing the coefficients in a parity-check matrix which optimizes the minimum distance of the binary image of the code is a formidable task...

Cycle LDPC Codes
Coefficient Optimization

- Less complex (sub-optimum): optimize the coefficients row-wise[39]
- Example: Parity-check matrix over $\mathbb{F}_{256}$, $p(x) = x^8 + x^4 + x^3 + x^2 + 1$

$$\mathbf{H} = \begin{bmatrix} \alpha^8 & \alpha^{80} & \alpha^0 & 0 \\ \alpha^0 & \alpha^{41} & \alpha^{122} & \alpha^{113} \end{bmatrix}$$

The first row imposes the equation $c_1\alpha^8 + c_2\alpha^{80} + c_3\alpha^0 = 0$, i.e.

$$[c_1, c_2, c_3]\,\mathbf{h}_1^T = 0 \qquad \mathbf{h}_1 = \left[\alpha^8, \alpha^{80}, \alpha^0\right]$$

(non-binary SPC code)

[39] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular $(2, d_c)$-LDPC codes over GF(q) using their binary images," *IEEE Trans. Commun.*, 2008

Cycle LDPC Codes
Coefficient Optimization

- Let us analyze the binary code associated with the non-binary SPC code with parity-check matrix $\mathbf{h}_1^T = \left[\alpha^8, \alpha^{80}, \alpha^0\right]$
- Companion matrix of $p(x) = \sum_{i=0}^{8} p_i x^i = x^8 + x^4 + x^3 + x^2 + 1$

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ p_0 & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \end{pmatrix}$$

- Denote by $\mathbf{c}_1$ the 8-bits representation of the symbol c_1, and by $\mathbf{s}_1$ the 8-bits binary image of $s_1 = c_1\alpha^i$. Then

$$\mathbf{s}_1 = \mathbf{c}_1 \mathbf{M}^i$$

DLR

Cycle LDPC Codes
Coefficient Optimization

- The binary code associated with the non-binary SPC code having $\mathbf{h}_1^T = \left[\alpha^8, \alpha^{80}, \alpha^0\right]$ has parity-check matrix given by

$$\mathbf{H}_r^{\mathsf{b}} = \left[\mathbf{M}^8 | \mathbf{M}^{80} | \mathbf{I}\right]$$

- This is the parity-check matrix of a $(24, 16)$ binary linear block code

Cycle LDPC Codes
Coefficient Optimization

- Distance spectrum of the code with parity-check matrix given by

$$\mathbf{H}_r^{\mathsf{b}} = \left[\mathbf{M}^{8}|\mathbf{M}^{80}|\mathbf{I}\right]$$

via MacWilliams identity[40]

$$A(x) = \frac{1}{2^{n-k}}\,(1+x)^n\,B\left(\frac{1-x}{1+x}\right)$$

where $A(x)$ is the WEF we are interested in, and $B(x)$ is the one of the dual code (having $\mathbf{H}_r^{\mathsf{b}}$ as generator matrix)

[40] J. MacWilliams and N. Sloane, *The theory of error-correcting codes.* North Holland Mathematical Libray, 1977

Cycle LDPC Codes
Coefficient Optimization

- Distance spectrum of the code with parity-check matrix given by

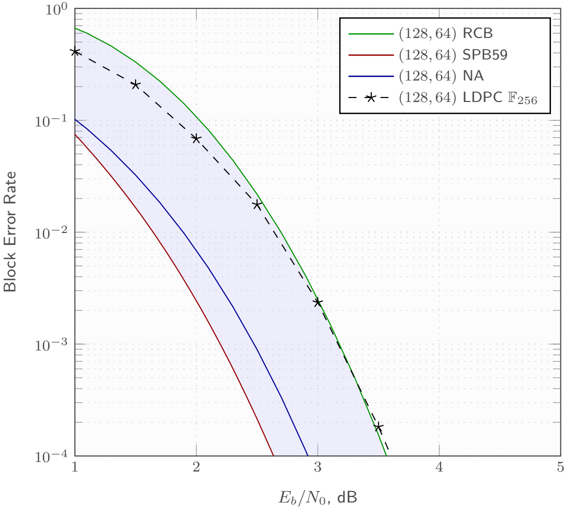
$$\mathbf{H}_r^{\mathsf{b}} = \left[\mathbf{M}^{8}|\mathbf{M}^{80}|\mathbf{I}\right]$$

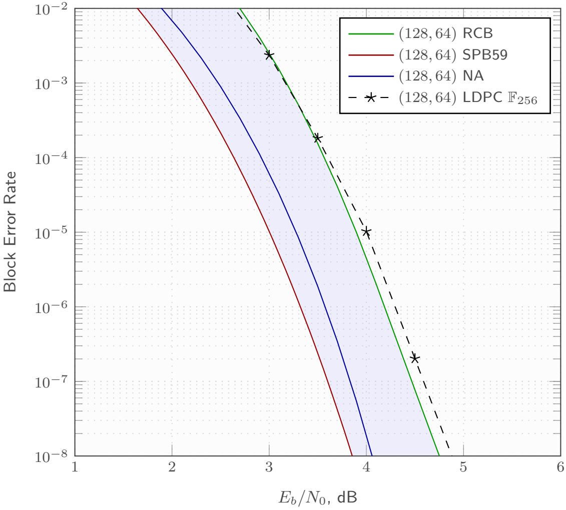
$$\begin{aligned} A(x) = 1 &+ 40x^4 + 202x^5 + 502x^6 + 1307x^7 + 2980x^8 + 5082x^9 + 7486x^{10} + \\ &+ 9854x^{11} + 10698x^{12} + 9686x^{13} + 7618x^{14} + 5076x^{15} + 2875x^{16} + \\ &+ 1406x^{17} + 522x^{18} + 146x^{19} + 46x^{20} + 8x^{21} + x^{23} \end{aligned}$$

Cycle LDPC Codes
Coefficient Optimization

List of good row coefficients, $\mathbb{F}_{256}$, check node degree 3

Coefficients, $\left[\alpha^{i}\,\alpha^{j}\,\alpha^{0}\right]$	Minimum Distance	Multiplicity
$\left[\alpha^{8}\,\alpha^{183}\,\alpha^{0}\right]$	4	36
$\left[\alpha^{10}\,\alpha^{82}\,\alpha^{0}\right]$	4	37
$\left[\alpha^{16}\,\alpha^{167}\,\alpha^{0}\right]$	4	37
$\left[\alpha^{41}\,\alpha^{127}\,\alpha^{0}\right]$	4	37
$\left[\alpha^{41}\,\alpha^{128}\,\alpha^{0}\right]$	4	37
$\left[\alpha^{42}\,\alpha^{128}\,\alpha^{0}\right]$	4	37
$\left[\alpha^{86}\,\alpha^{214}\,\alpha^{0}\right]$	4	37

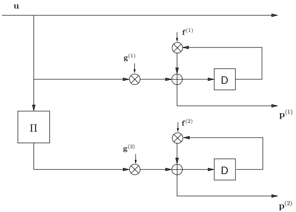

$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$(128, 64)$ LDPC $\mathbb{F}_{256}$
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB


10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
Block Error Rate
1
2
3
4
5
6
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$(128, 64)$ LDPC $\mathbb{F}_{256}$

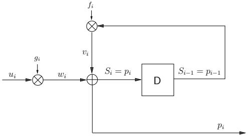
Non-Binary Turbo Codes
Basics

- Turbo codes on high-order fields[41] as LDPC codes[42]
- Memory-1 (in field symbols), time-variant convolutional codes

Turbo encoder



Detail of the RSCC encoder

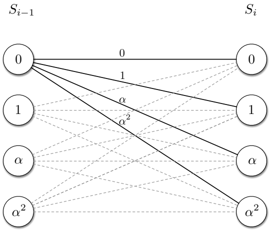


[41] J. Berkmann, *Iterative Decoding of Nonbinary Codes*. Munich, Germany: Ph.D. dissertation, Tech. Univ. München, 2000

[42] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani, "Short turbo codes over high order fields," *IEEE Trans. Commun.*, 2013

Turbo Codes based on Memory-1 Time-Variant RCCs
Decoding

- Decoding over the q-states trellis of the component codes
- q branches leaving/entering a state in each section
- Decoding complexity scales as $\mathcal{O}(q^2)$
- Can be reduced to $\mathcal{O}(q \log_2 q)$ by decoding through fast Fourier transform[43]

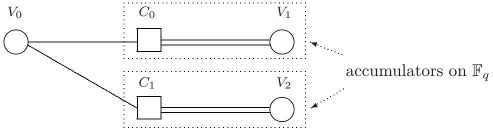


[43] J. Berkmann and C. Weiss, "On dualizing trellis-based APP decoding algorithms," *IEEE Trans. Commun.*, 2002

Non-Binary Turbo Codes

Memory-1 Time-Variant Recursive Convolutional Codes

- The codes admit a non-binary protograph LDPC code representation



- Decoding threshold as for cycle LDPC codes: $\mathbb{F}_{256}$ $R = 1/3$ protograph ensemble possesses $(E_b/N_0)^\star = -0.214$ dB vs. $(E_b/N_0)^\star = -0.225$ dB of the unstructured one

Turbo Codes based on Memory-1 Time-Variant RCCs

Interleaver Design and Graph Interpretation

- For binary turbo codes, a large-spread regular interleaver is not an option, due to the large multiplicity of minimum-weight codewords

[44] C. Radebaugh, C. Powell, and R. Koetter, "Wheel codes: Turbo-like codes on graphs of small order," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2003

Turbo Codes based on Memory-1 Time-Variant RCCs

Interleaver Design and Graph Interpretation

- For binary turbo codes, a large-spread regular interleaver is not an option, due to the large multiplicity of minimum-weight codewords
- The interleaver shall sacrifice spread for irregularity (randomness)

[44] C. Radebaugh, C. Powell, and R. Koetter, "Wheel codes: Turbo-like codes on graphs of small order," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2003

Turbo Codes based on Memory-1 Time-Variant RCCs

Interleaver Design and Graph Interpretation

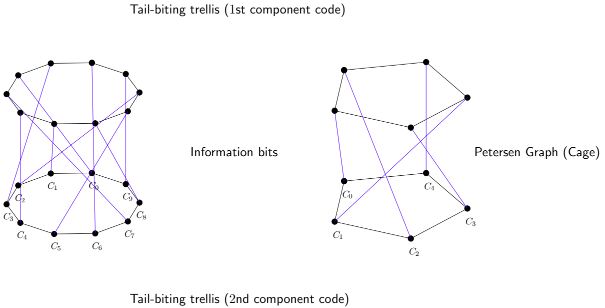
- For binary turbo codes, a large-spread regular interleaver is not an option, due to the large multiplicity of minimum-weight codewords
- The interleaver shall sacrifice spread for irregularity (randomness)
- For non-binary turbo codes, the component codes do not have anymore a periodic trellis (time-variant FB/FFW coefficients), thus large-spread regular interleavers can be used[44]

[44] C. Radebaugh, C. Powell, and R. Koetter, "Wheel codes: Turbo-like codes on graphs of small order," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2003

Turbo Codes based on Memory-1 Time-Variant RCCs

Interleaver Design and Graph Interpretation

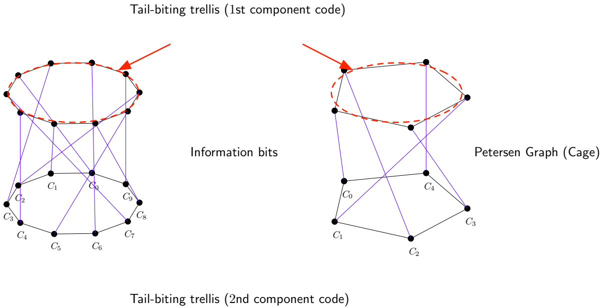
- Non-binary turbo codes can be defined by the graph of a cycle code



Turbo Codes based on Memory-1 Time-Variant RCCs

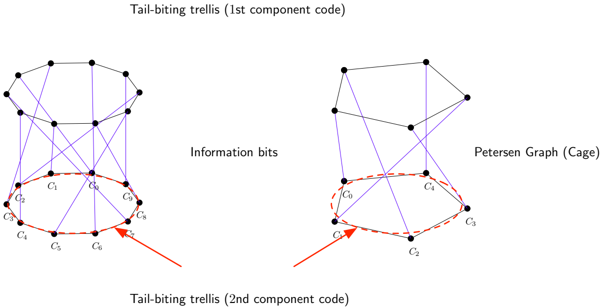
Interleaver Design and Graph Interpretation

- Non-binary turbo codes can be defined by the graph of a cycle code



Turbo Codes based on Memory-1 Time-Variant RCCs
Interleaver Design and Graph Interpretation

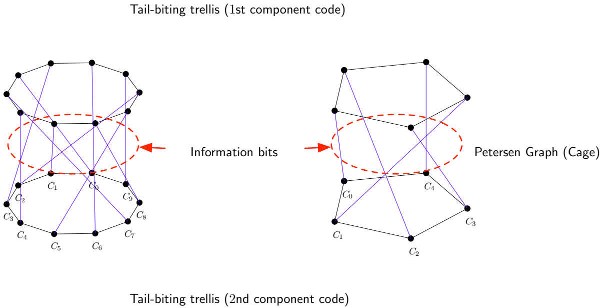
- Non-binary turbo codes can be defined by the graph of a cycle code

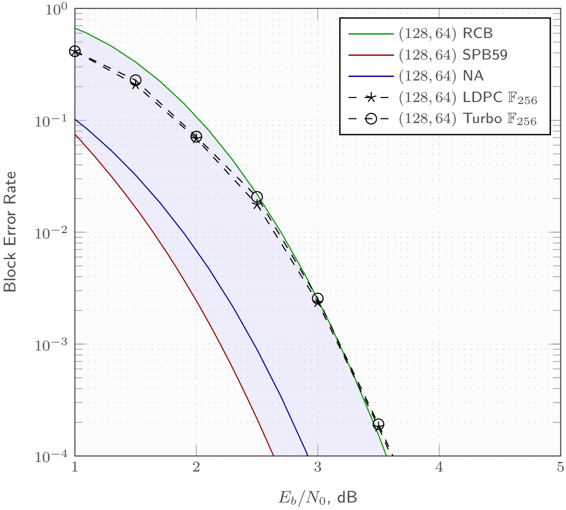


Turbo Codes based on Memory-1 Time-Variant RCCs

Interleaver Design and Graph Interpretation

- Non-binary turbo codes can be defined by the graph of a cycle code





10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$(128, 64)$ LDPC $\mathbb{F}_{256}$
$(128, 64)$ Turbo $\mathbb{F}_{256}$

Non-Binary Low-Density Parity-Check Codes
Observations

- Remarkably close to the RCB

Non-Binary Low-Density Parity-Check Codes
Observations

- Remarkably close to the RCB
- Ultra-sparse design is nearly-optimal with large field orders

Non-Binary Low-Density Parity-Check Codes
Observations

- Remarkably close to the RCB
- Ultra-sparse design is nearly-optimal with large field orders
- Low error floors

Non-Binary Low-Density Parity-Check Codes
Observations

- Remarkably close to the RCB
- Ultra-sparse design is nearly-optimal with large field orders
- Low error floors
- High decoding complexity

Outline

Polar Codes
Introduction

- Class of provably capacity achieving codes over memoryless binary input output symmetric channels under low-complexity (successive cancellation) decoding[45]

[45] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, 2009

[46] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, 2015

Polar Codes
Introduction

- Class of provably capacity achieving codes over memoryless binary input output symmetric channels under low-complexity (successive cancellation) decoding[45]
- Their performance at short block lengths is disappointing but...

---

[45] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, 2009

[46] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, 2015

Polar Codes
Introduction

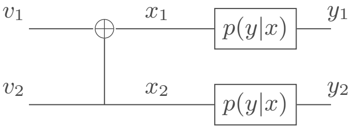
- Class of provably capacity achieving codes over memoryless binary input output symmetric channels under low-complexity (successive cancellation) decoding[45]
- Their performance at short block lengths is disappointing but...

  list decoding with the aid of an outer-high rate code[46] yields one of the best code constructions at short block lengths!

[45] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, 2009

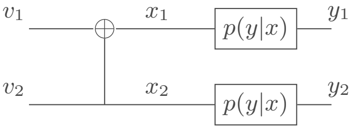
[46] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, 2015

Polar Codes



$$\mathbf{x} = \mathbf{v}\mathbf{G}_2 \qquad\qquad \mathbf{G}_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Polar Codes



$$\mathbf{x} = \mathbf{v}\mathbf{G}_2 \qquad \mathbf{G}_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Polar Codes

Denote $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Then

$$\mathbf{x} = \mathbf{u}\mathbf{G}_n$$

with $\mathbf{G}_n$ being a $n \times n$ matrix with structure

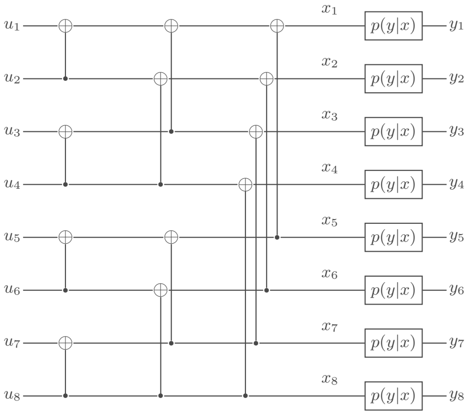
$$\mathbf{G}_n = \mathbf{G}_2 \otimes \mathbf{G}_2 \otimes \ldots \otimes \mathbf{G}_2$$

Polar Codes
Example

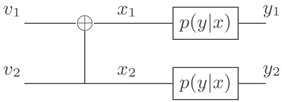
With $n = 8$, $\mathbf{G}_8 = \mathbf{G}_2 \otimes \mathbf{G}_2 \otimes \mathbf{G}_2$

$$\mathbf{G}_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

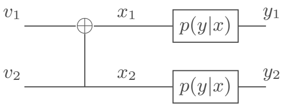
Polar Codes
Example



Polar Codes
Successive Cancellation Decoding

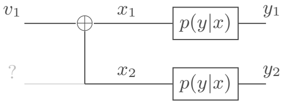


Polar Codes
Successive Cancellation Decoding



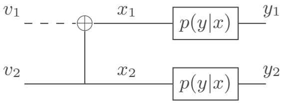
$$p(\mathbf{y}|\mathbf{v}) = p(y_1|v_1 + v_2)p(y_2|v_2)$$

Polar Codes
Successive Cancellation Decoding



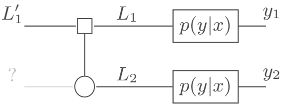
$$p(\mathbf{y}|v_1) = \sum_{v_2} p(\mathbf{y}, v_2|v_1) = \frac{1}{2} \sum_{v_2} p(y_1|v_1 + v_2)p(y_2|v_2)$$

Polar Codes
Successive Cancellation Decoding



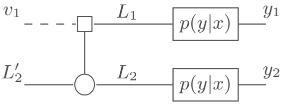
$$p(\mathbf{y}, v_1|v_2) = p(\mathbf{y}|v_1, v_2)p(v_1) = \frac{1}{2}p(y_1|v_1 + v_2)p(y_2|v_2)$$

Polar Codes
Successive Cancellation Decoding



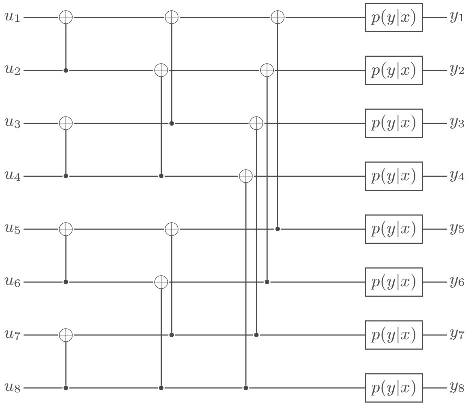
$$L_1' = 2\tanh^{-1}\left(\tanh\left(\frac{L_1}{2}\right)\tanh\left(\frac{L_2}{2}\right)\right) \qquad \text{with} \qquad L_i = \log\frac{p(y_i|0)}{p(y_i|1)}$$

Polar Codes
Successive Cancellation Decoding



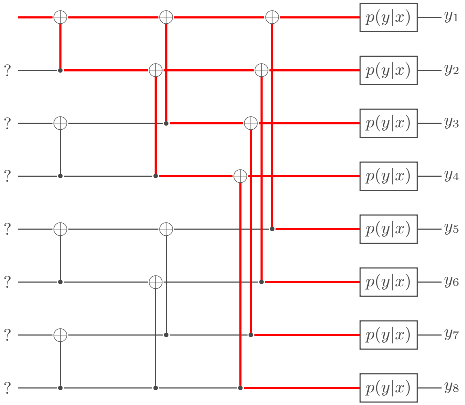
$$L_2' = L_2 + (-1)^{v_1} L_1$$

Polar Codes
Successive Cancellation Decoding



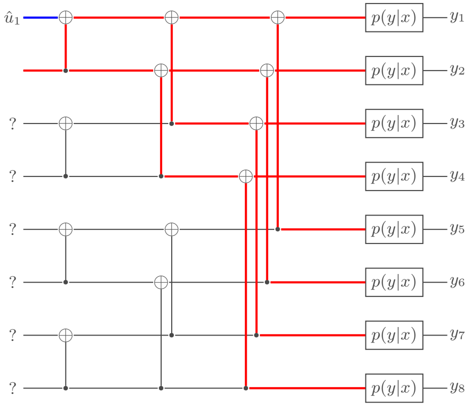
Polar Codes
Successive Cancellation Decoding

$$\hat{u}_1 = \arg\max_{u_1} p(\mathbf{y}|u_1)$$

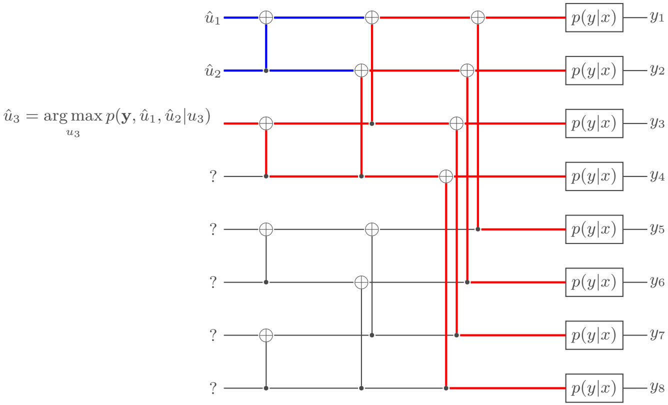


Polar Codes
Successive Cancellation Decoding

$$\hat{u}_2 = \arg\max_{u_2} p(\mathbf{y}, \hat{u}_1 | u_2)$$

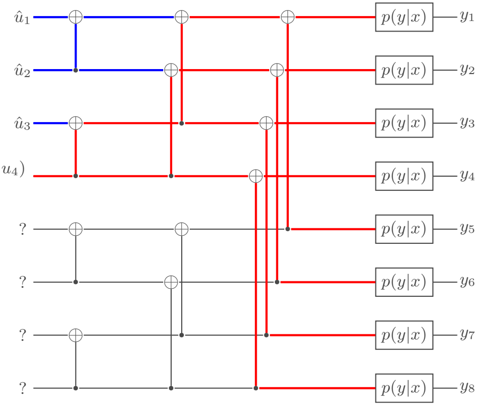


Polar Codes
Successive Cancellation Decoding



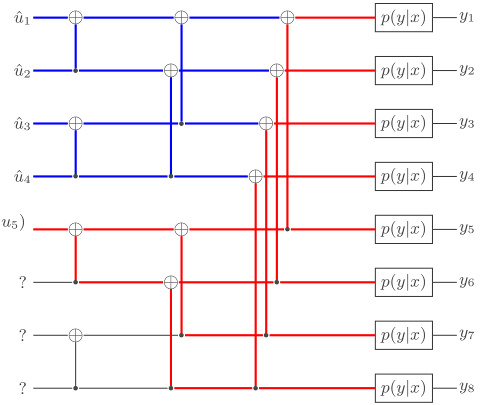
Polar Codes
Successive Cancellation Decoding

$$\hat{u}_4 = \arg\max_{u_4} p(\mathbf{y}, \hat{u}_1, \ldots, \hat{u}_3 | u_4)$$

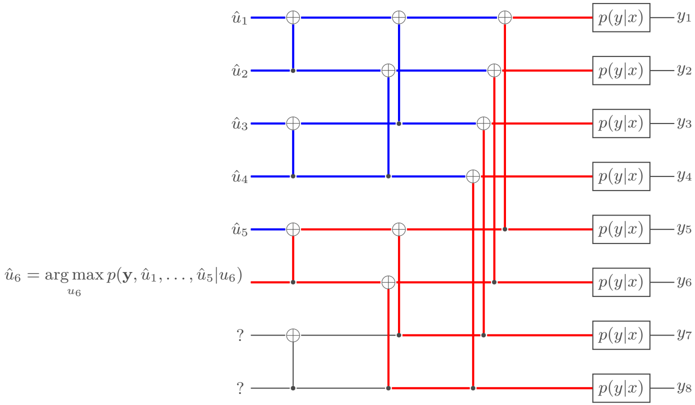


Polar Codes
Successive Cancellation Decoding

$$\hat{u}_5 = \arg\max_{u_5} p(\mathbf{y}, \hat{u}_1, \ldots, \hat{u}_4 | u_5)$$

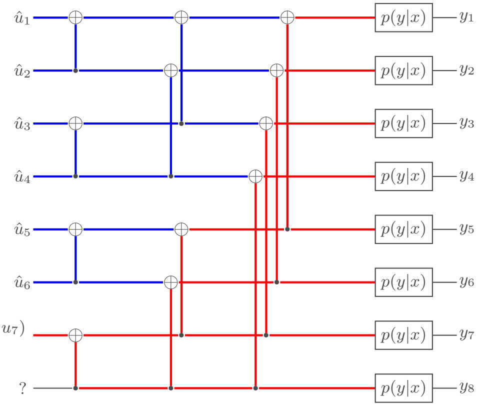


Polar Codes
Successive Cancellation Decoding

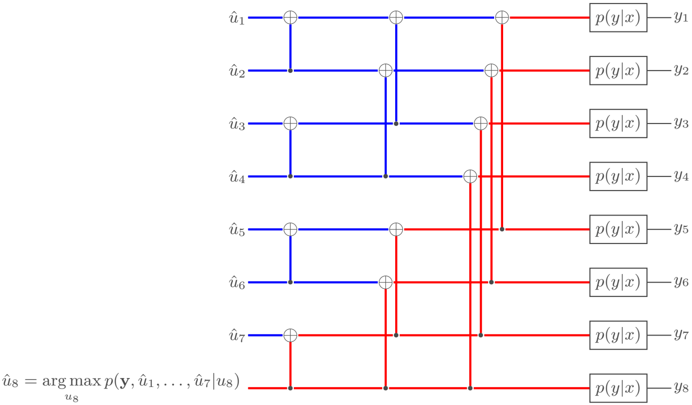


Polar Codes
Successive Cancellation Decoding

$$\hat{u}_7 = \arg\max_{u_7} p(\mathbf{y}, \hat{u}_1, \ldots, \hat{u}_6 | u_7)$$



Polar Codes
Successive Cancellation Decoding



Polar Codes
Code Design

- (n, k) polar code: $\mathcal{A} =$ set of k indexed in $\{1, 2, \ldots, n\}$
- Map the k information bits on u_i, $i \in \mathcal{A}$

[47] N. Stolte, "Rekursive Codes mit der Plotkin-Konstruktion und ihre Decodierung," Ph.D. dissertation, TU Darmstadt, 2002

[48] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, 2009

Polar Codes
Code Design

- (n, k) polar code: $\mathcal{A} =$ set of k indexed in $\{1, 2, \ldots, n\}$
- Map the k information bits on u_i, $i \in \mathcal{A}$
- Set the remaining elements of $\mathbf{u}$ to 0 (frozen bits)

[47] N. Stolte, “Rekursive Codes mit der Plotkin-Konstruktion und ihre Decodierung,” Ph.D. dissertation, TU Darmstadt, 2002

[48] E. Arikan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” *IEEE Trans. Inf. Theory*, 2009

Polar Codes
Code Design

- (n,k) polar code: $\mathcal{A} =$ set of k indexed in $\{1,2,\ldots,n\}$
- Map the k information bits on u_i, $i \in \mathcal{A}$
- Set the remaining elements of $\mathbf{u}$ to 0 (frozen bits)
- Selection of the frozen bits: For the target channel, find the least $n-k$ reliable bits in $\mathbf{u}$ under successive cancellation decoding[47][48]

[47] N. Stolte, "Rekursive Codes mit der Plotkin-Konstruktion und ihre Decodierung," Ph.D. dissertation, TU Darmstadt, 2002

[48] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, 2009

Polar Codes
Example

- $(8,4)$ polar code: $\mathcal{A}=\{4,6,7,8\}$

$$\mathbf{G}_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Polar Codes
Example

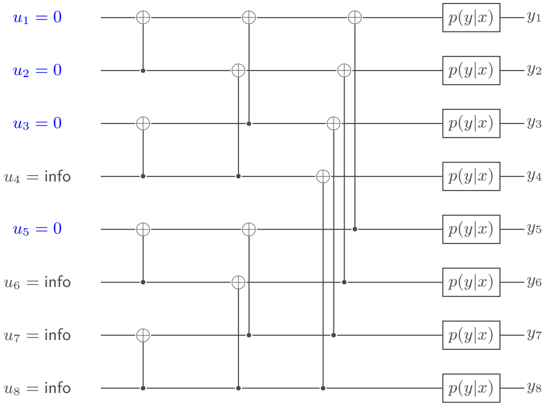
- $(8, 4)$ polar code: $\mathcal{A} = \{4, 6, 7, 8\}$

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

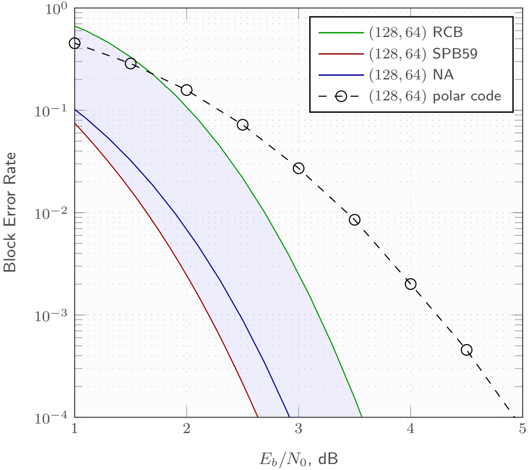
- $\mathbf{G}$: generator matrix of the $(8, 4)$ polar code

Polar Codes

Example: $u_1 = u_2 = u_3 = u_5 = 0$ (frozen bits)



Polar Codes: Shortcomings



Albeit capacity-achieving (for large n), at moderate-short block lengths polar codes under successive cancellation decoding perform poorly

Polar Codes
List Decoding: Principle

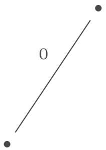
- List decoding: Exploit the serial bit decision process to improve the SC decoder performance

List size $L = 4$

•

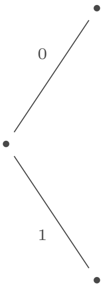
$\hat{u}_i$

List size $L = 4$



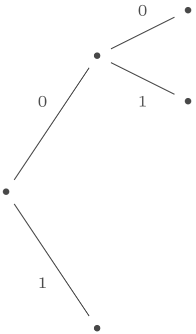
$\hat{u}_i$

List size $L = 4$



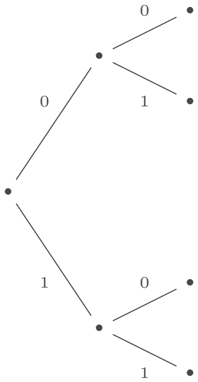
$\hat{u}_i$ $\hat{u}_{i+1}$

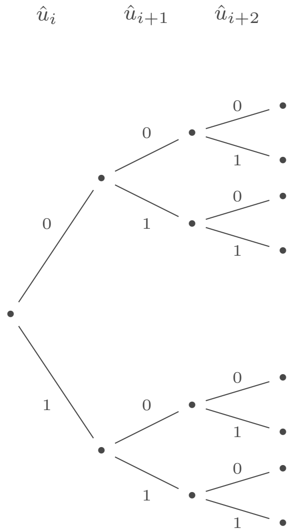
List size $L = 4$



$\hat{u}_i$ $\hat{u}_{i+1}$

List size $L = 4$

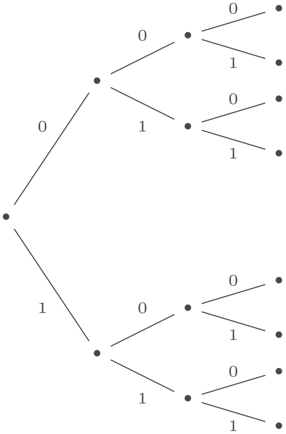




List size $L = 4$

$\hat{u}_i$ $\hat{u}_{i+1}$ $\hat{u}_{i+2}$

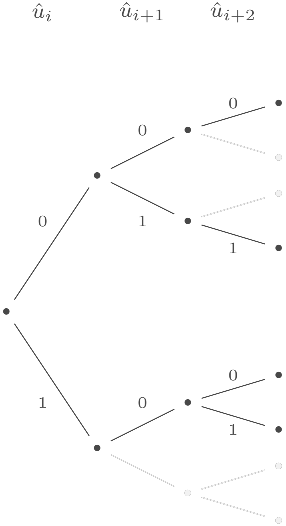
List size $L = 4$



Rank based on

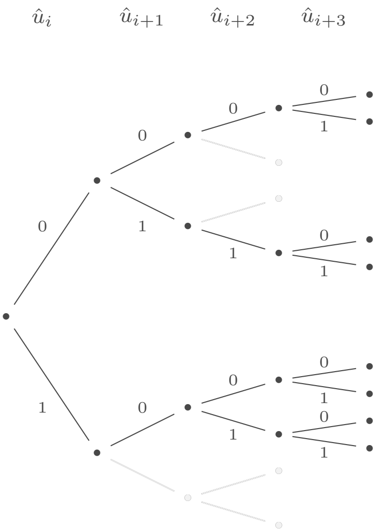
$$p\left(\mathbf{y}|\hat{u}_1, \ldots, \hat{u}_{i+2}\right)$$

List size $L = 4$



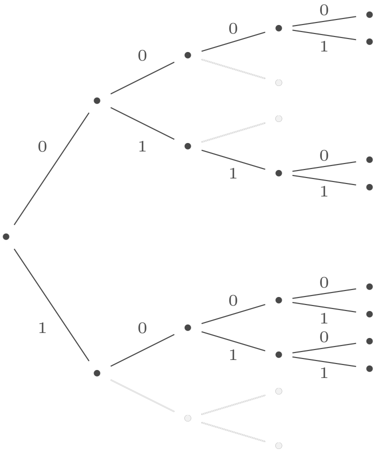
Discard the $L/2$ least likely paths

List size $L = 4$



$\hat{u}_i$ $\hat{u}_{i+1}$ $\hat{u}_{i+2}$ $\hat{u}_{i+3}$

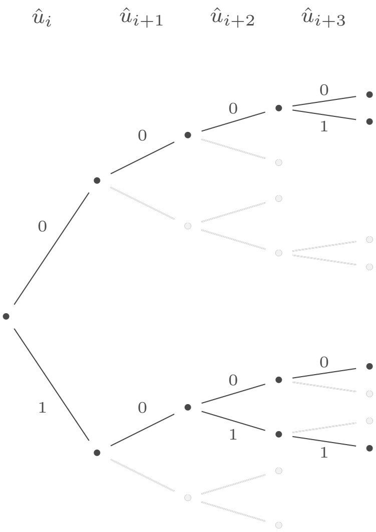
List size $L = 4$



Rank based on

$$p\left(\mathbf{y}|\hat{u}_1, \ldots, \hat{u}_{i+3}\right)$$

List size $L = 4$



Discard the $L/2$ least likely paths

Polar Codes
List Decoding: Principle

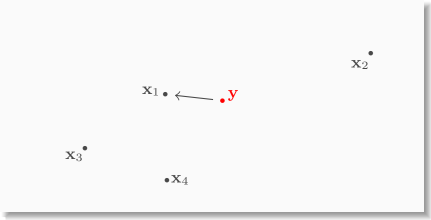
- After k steps, L codewords in the list $\mathcal{L}$

Polar Codes
List Decoding: Principle

- After k steps, L codewords in the list $\mathcal{L}$

Polar Codes
List Decoding: Principle

- After k steps, L codewords in the list $\mathcal{L}$

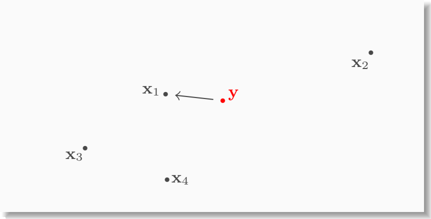


- Pick the codeword in $\mathcal{L}$ maximizing the likelihood

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}\in\mathcal{L}} p\left(\mathbf{y}|\mathbf{x}\right)$$

Polar Codes
List Decoding: Principle

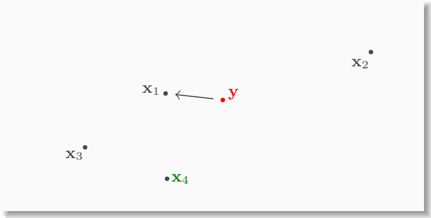
- Two error events:



- The correct codeword $\mathbf{x}$ is not in the list

Polar Codes
List Decoding: Principle

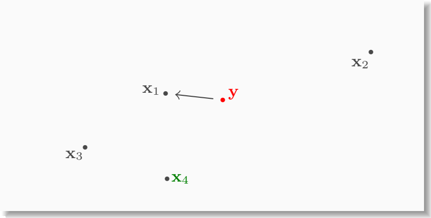
- Two error events:



- The correct codeword $\mathbf{x}$ is not in the list
- The correct codeword $\mathbf{x}$ is in the list but $\exists \mathbf{x}' \in \mathcal{L}$ s.t. $p(\mathbf{y}|\mathbf{x}') > p(\mathbf{y}|\mathbf{x})$

Polar Codes
List Decoding: Principle

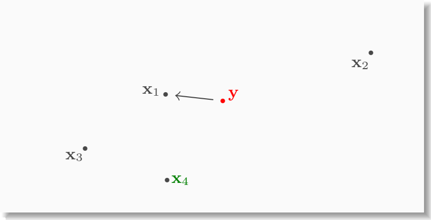
- Two error events:



- The correct codeword $\mathbf{x}$ is not in the list
- The correct codeword $\mathbf{x}$ is in the list but $\exists \mathbf{x}' \in \mathcal{L}$ s.t. $p(\mathbf{y}|\mathbf{x}') > p(\mathbf{y}|\mathbf{x})$
  The error would take place even with ML decoding...

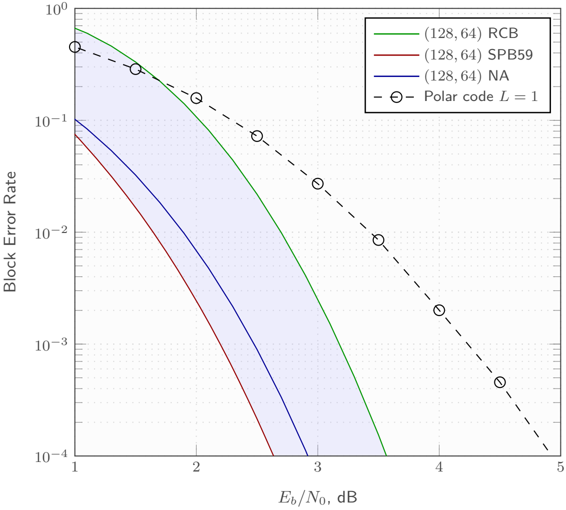
Polar Codes
List Decoding: Principle

- Two error events:

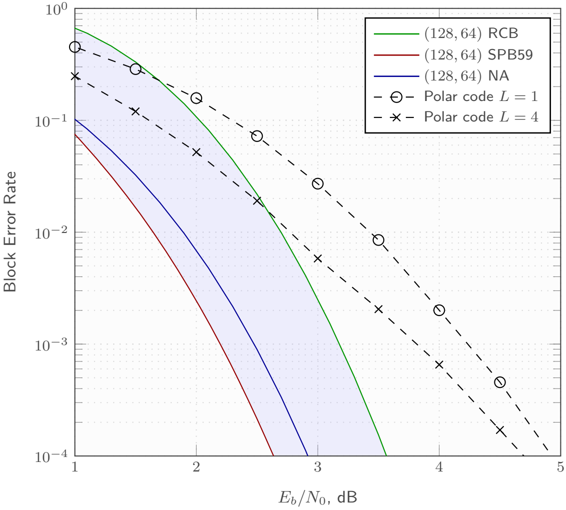


- The correct codeword $\mathbf{x}$ is not in the list
- The correct codeword $\mathbf{x}$ is in the list but $\exists \mathbf{x}' \in \mathcal{L}$ s.t. $p(\mathbf{y}|\mathbf{x}') > p(\mathbf{y}|\mathbf{x})$
  The error would take place even with ML decoding...

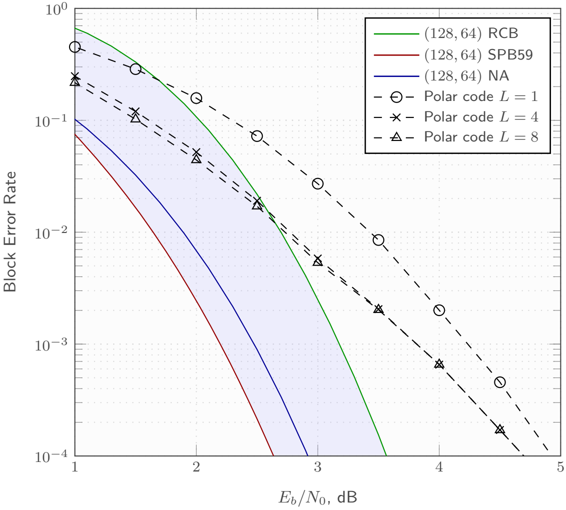
Performance limited by distance spectrum

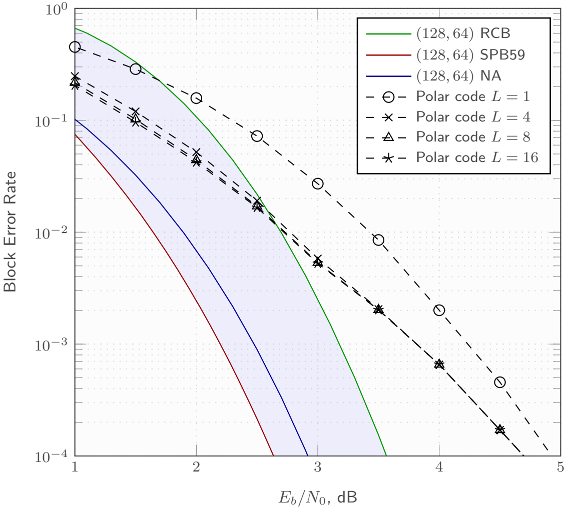

(128, 64) RCB
(128, 64) SPB59
(128, 64) NA
Polar code $L = 1$
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB

DLR

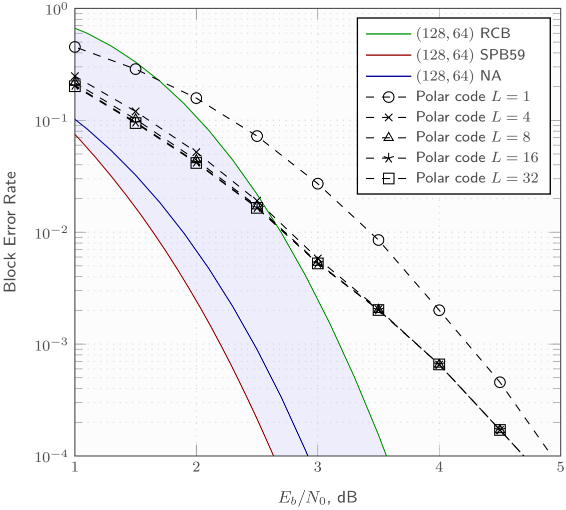


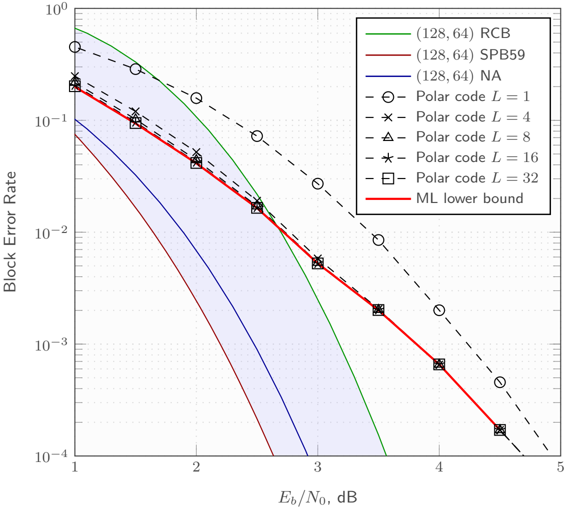
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
Polar code $L = 1$
Polar code $L = 4$


(128, 64) RCB
(128, 64) SPB59
(128, 64) NA
Polar code $L = 1$
Polar code $L = 4$
Polar code $L = 8$
Block Error Rate
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5
E_b/N_0, dB



10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
Polar code $L = 1$
Polar code $L = 4$
Polar code $L = 8$
Polar code $L = 16$

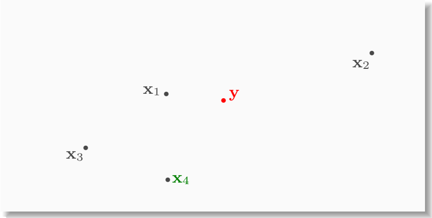

10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
Polar code $L = 1$
Polar code $L = 4$
Polar code $L = 8$
Polar code $L = 16$
Polar code $L = 32$



$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
Polar code $L = 1$
Polar code $L = 4$
Polar code $L = 8$
Polar code $L = 16$
Polar code $L = 32$
ML lower bound
Block Error Rate
E_b/N_0, dB
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
1
2
3
4
5

Polar Codes
List Decoding: Principle

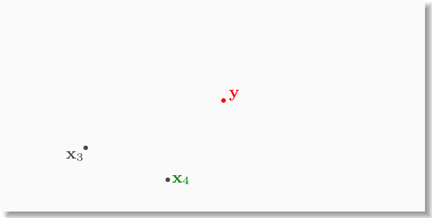
- Concatenation with an outer code to improve distance spectrum



- List decoding (inner code), followed by syndrome check with outer code

Polar Codes
List Decoding: Principle

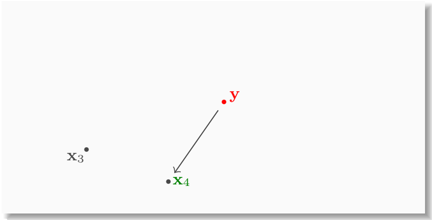
- Concatenation with an outer code to improve distance spectrum



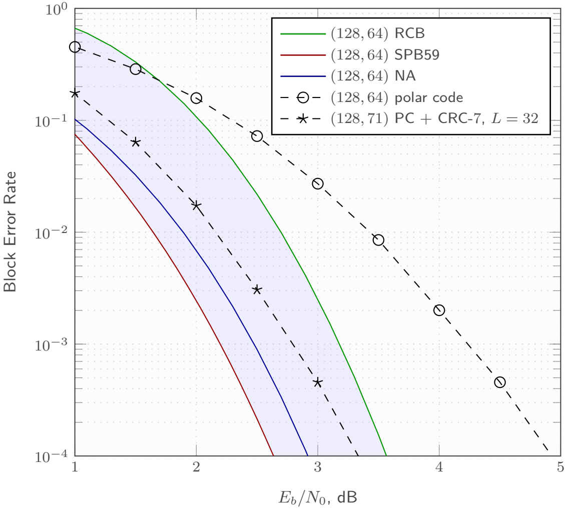
- List decoding (inner code), followed by syndrome check with outer code
- Expurgated list: all codewords not satisfying the check are removed

Polar Codes
List Decoding: Principle

- Concatenation with an outer code to improve distance spectrum



- List decoding (inner code), followed by syndrome check with outer code
- Expurgated list: all codewords not satisfying the check are removed
- Selection within the remaining codewords based on likelihood


10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Block Error Rate
1
2
3
4
5
E_b/N_0, dB
$(128, 64)$ RCB
$(128, 64)$ SPB59
$(128, 64)$ NA
$(128, 64)$ polar code
$(128, 71)$ PC + CRC-7, $L = 32$

Polar Codes
Observations

- With successive cancellation + list decoding and the aid of an outer code, consistently close to the normal approximation

[49] G. Ricciutelli, M. Baldi, F. Chiaraluce, and G. Liva, "On the error probability of short concatenated polar and cyclic codes with interleaving," *arXiv preprint arXiv:1701.07262*, 2017

Polar Codes
Observations

- With successive cancellation + list decoding and the aid of an outer code, consistently close to the normal approximation
- Complexity growing with the list size L

[49] G. Ricciutelli, M. Baldi, F. Chiaraluce, and G. Liva, "On the error probability of short concatenated polar and cyclic codes with interleaving," *arXiv preprint arXiv:1701.07262*, 2017

Polar Codes
Observations

- With successive cancellation $+$ list decoding and the aid of an outer code, consistently close to the normal approximation
- Complexity growing with the list size L
- Large list size: close to maximum-likelihood performance (but large complexity)

[49] G. Ricciutelli, M. Baldi, F. Chiaraluce, and G. Liva, "On the error probability of short concatenated polar and cyclic codes with interleaving," *arXiv preprint arXiv:1701.07262*, 2017

Polar Codes
Observations

- With successive cancellation + list decoding and the aid of an outer code, consistently close to the normal approximation
- Complexity growing with the list size L
- Large list size: close to maximum-likelihood performance (but large complexity)
- Error floor behavior only partially addressed[49]

[49] G. Ricciutelli, M. Baldi, F. Chiaraluce, and G. Liva, "On the error probability of short concatenated polar and cyclic codes with interleaving," *arXiv preprint arXiv:1701.07262*, 2017

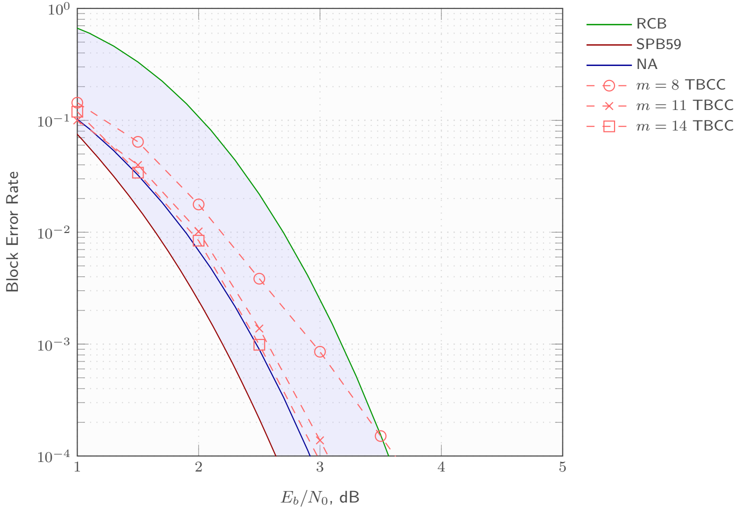
Polar Codes
Observations

- With successive cancellation + list decoding and the aid of an outer code, consistently close to the normal approximation
- Complexity growing with the list size L
- Large list size: close to maximum-likelihood performance (but large complexity)
- Error floor behavior only partially addressed[49]
- Good trade-off between decoding complexity and performance

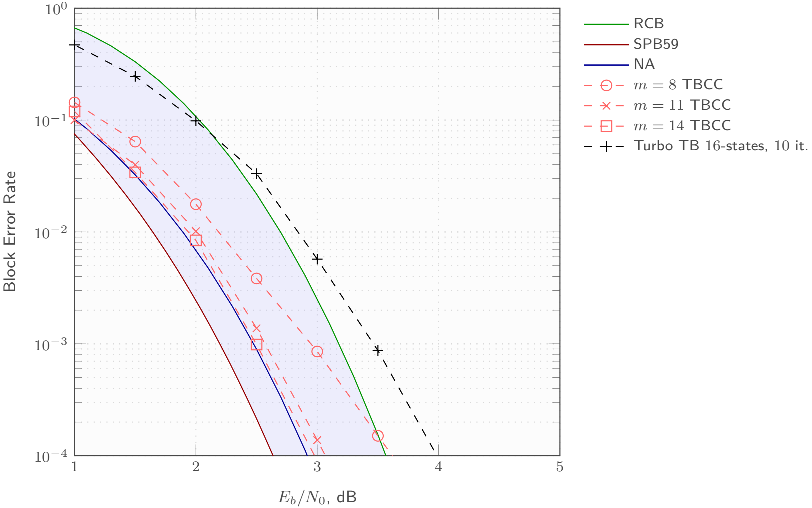
[49] G. Ricciutelli, M. Baldi, F. Chiaraluce, and G. Liva, "On the error probability of short concatenated polar and cyclic codes with interleaving," *arXiv preprint arXiv:1701.07262*, 2017

Outline

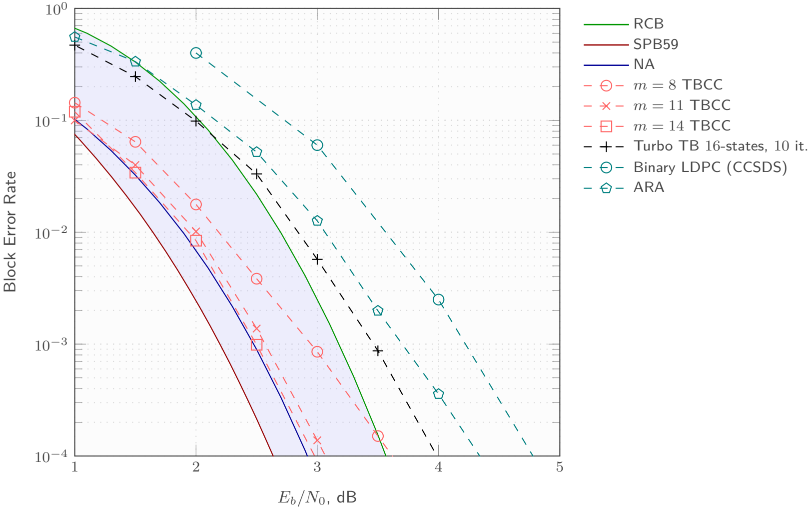
Case 1: $n = 128, k = 64$



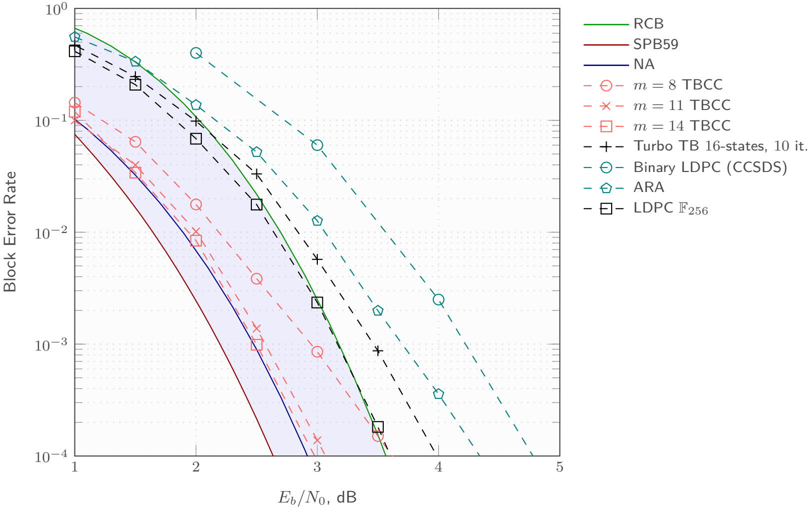
Case 1: $n = 128, k = 64$



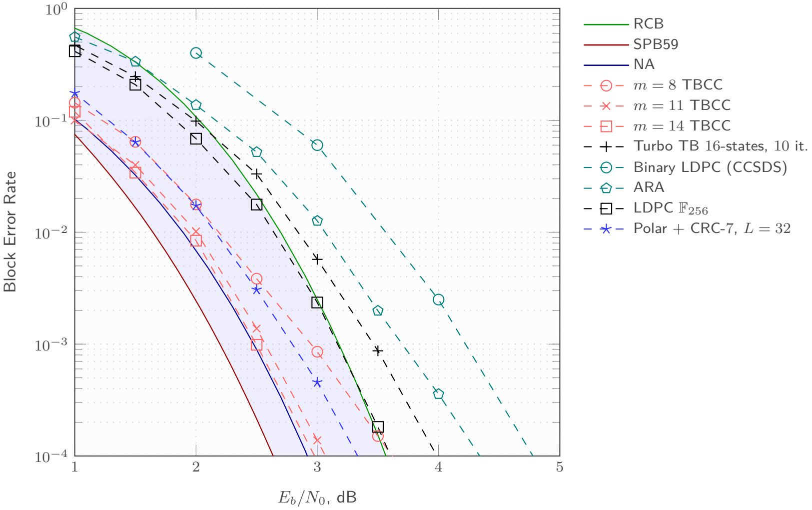
Case 1: $n = 128, k = 64$



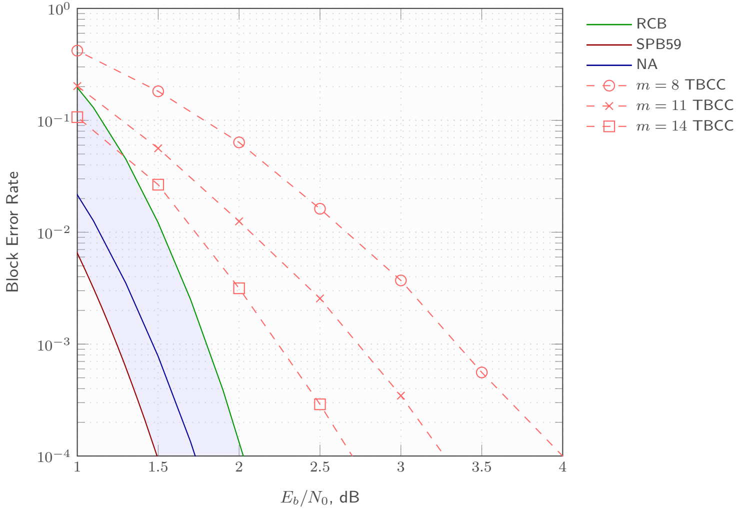
Case 1: $n = 128, k = 64$



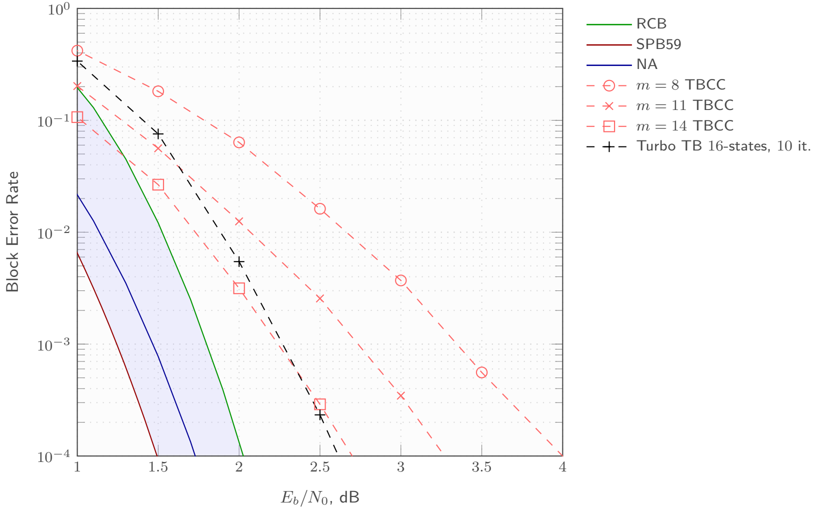
Case 1: $n = 128, k = 64$



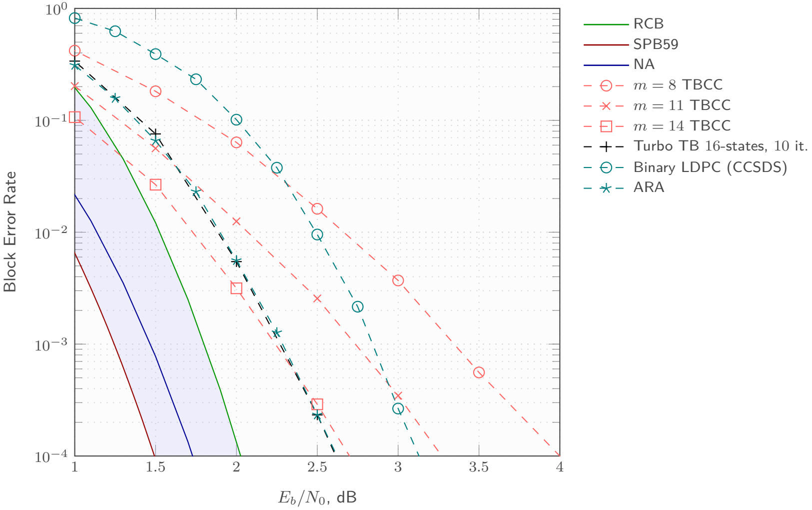
Case 2: $n = 512, k = 256$



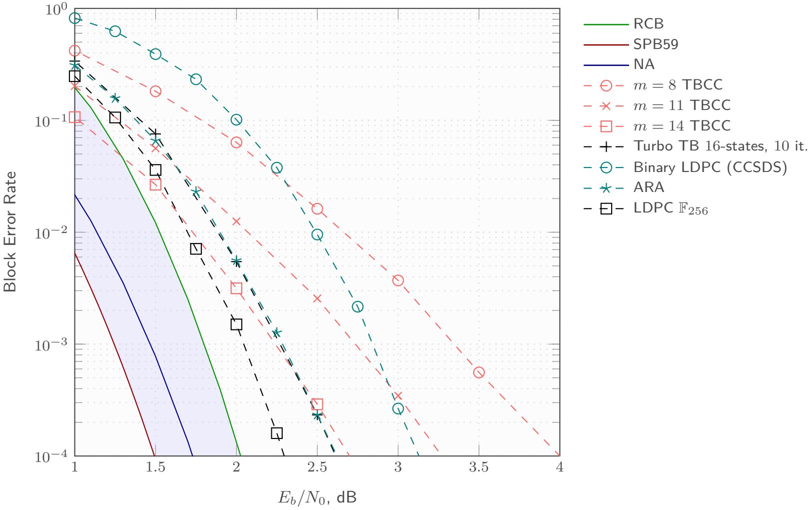
Case 2: $n = 512, k = 256$



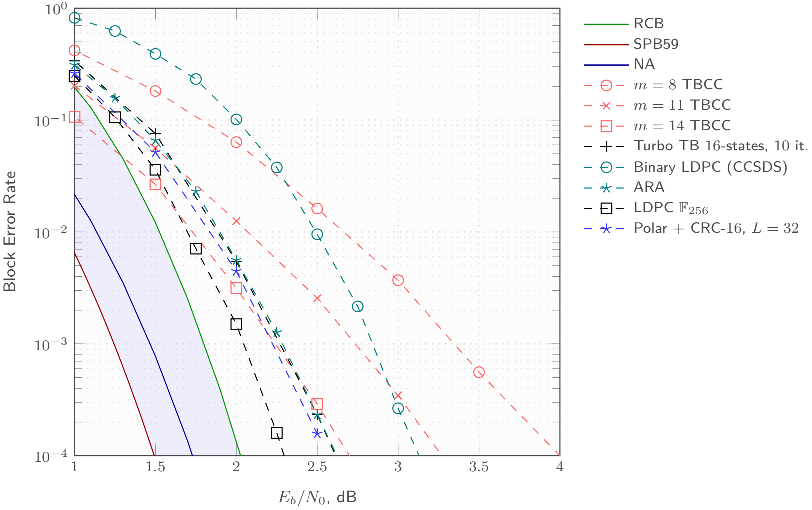
Case 2: $n = 512, k = 256$



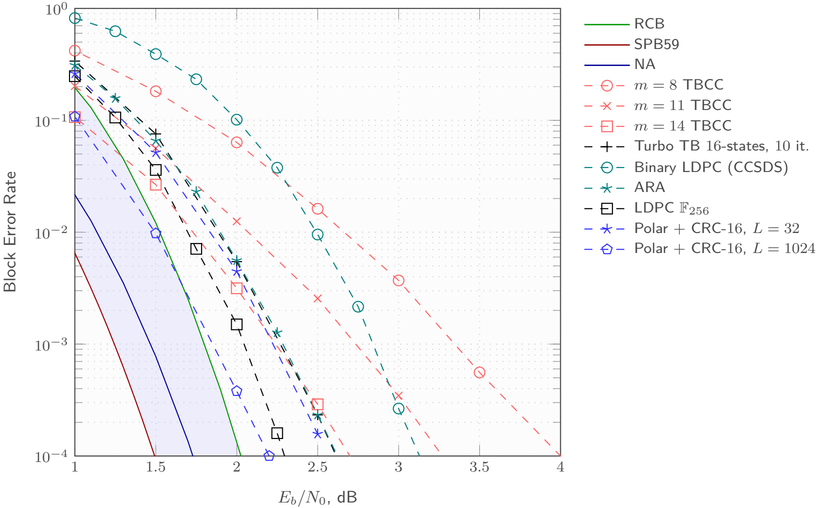
Case 2: $n = 512, k = 256$



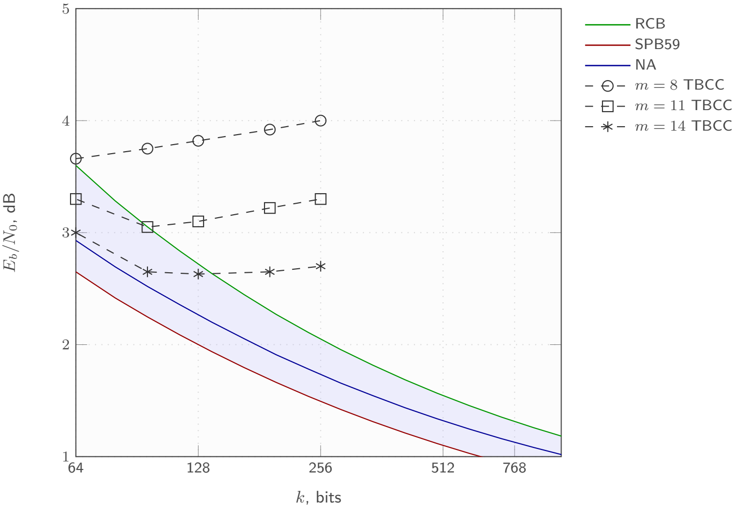
Case 2: $n = 512, k = 256$



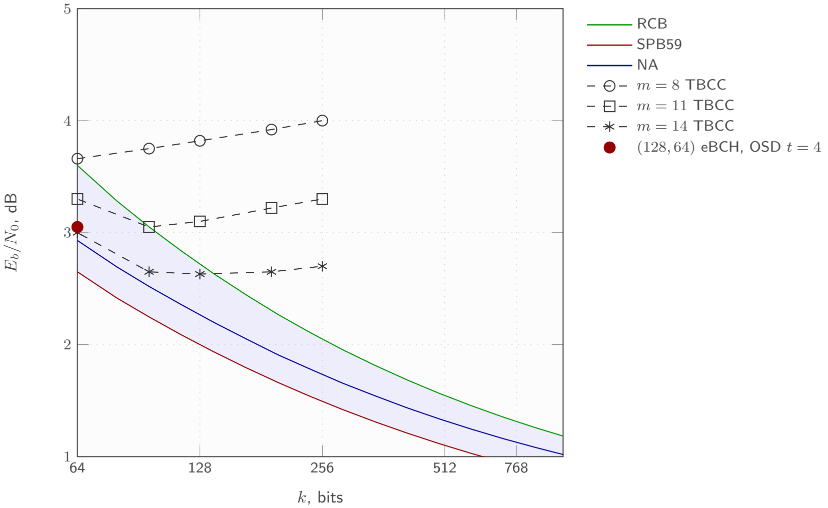
Case 2: $n = 512, k = 256$



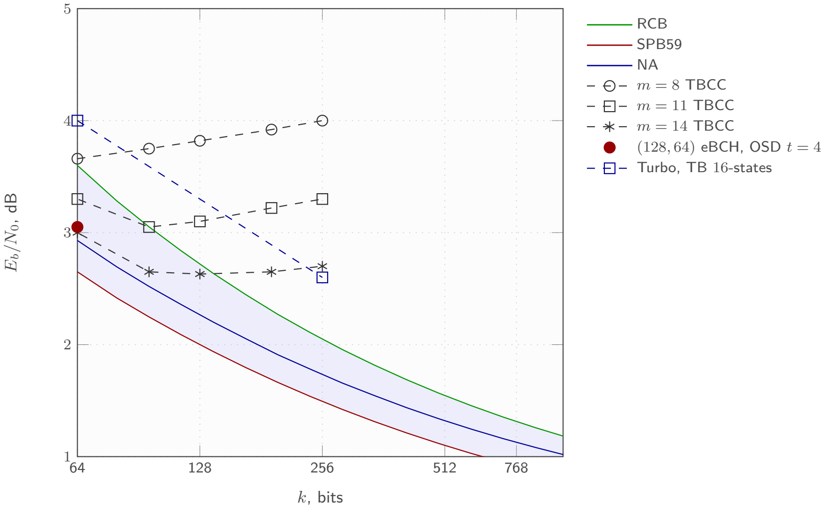
Comparing Rate-$1/2$ Codes at $P_B = 10^{-4}$



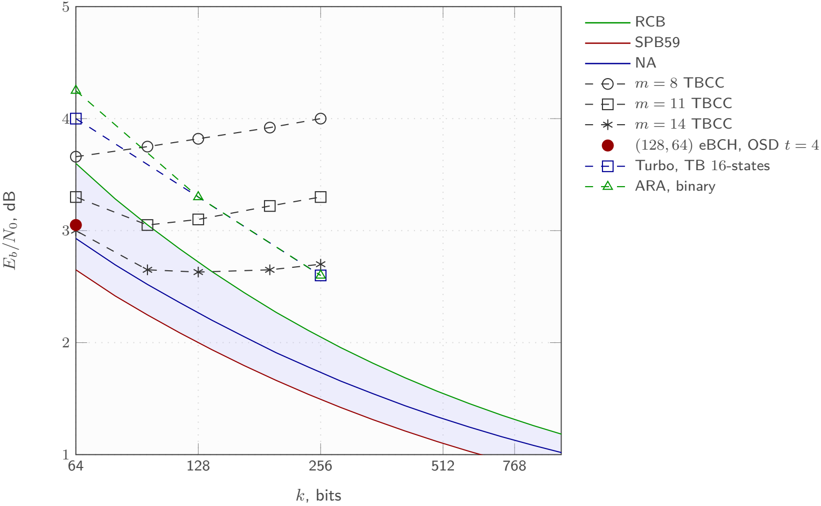
Comparing Rate-$1/2$ Codes at $P_B = 10^{-4}$



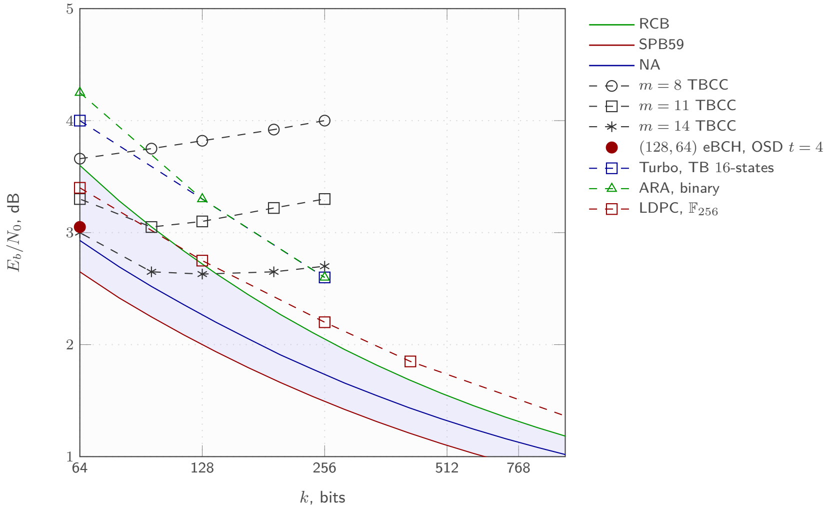
Comparing Rate-$1/2$ Codes at $P_B = 10^{-4}$



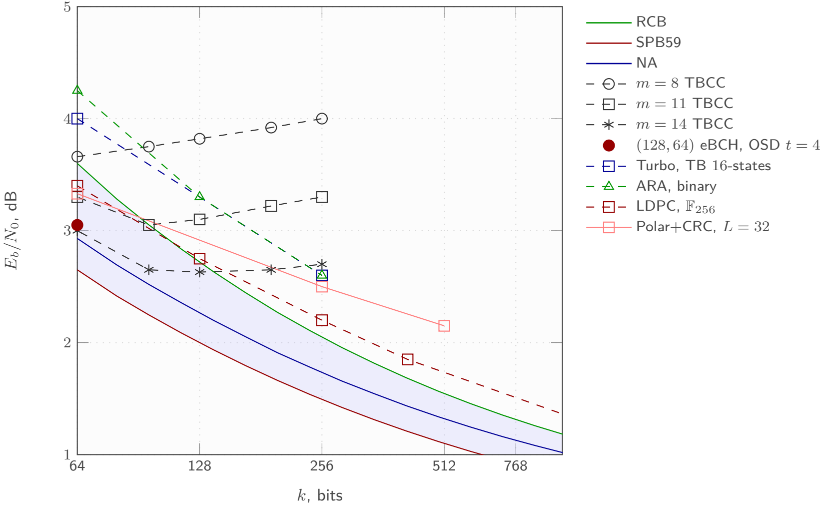
Comparing Rate-$1/2$ Codes at $P_B = 10^{-4}$



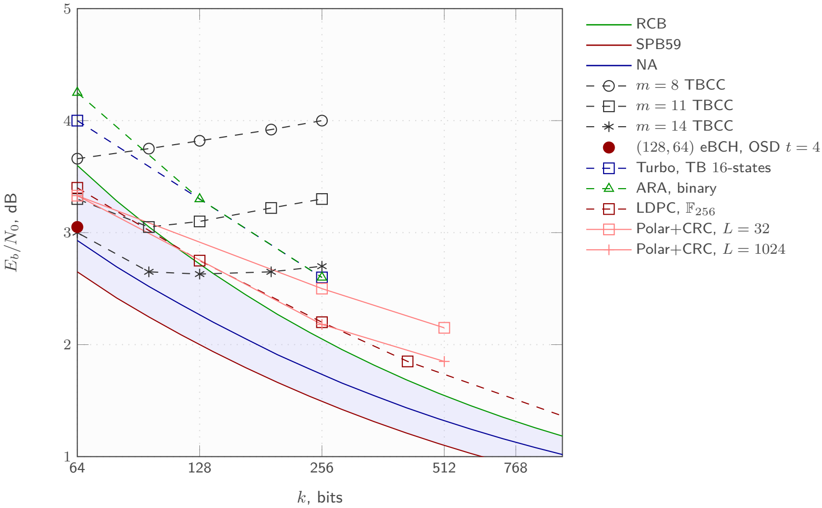
Comparing Rate-$1/2$ Codes at $P_B = 10^{-4}$



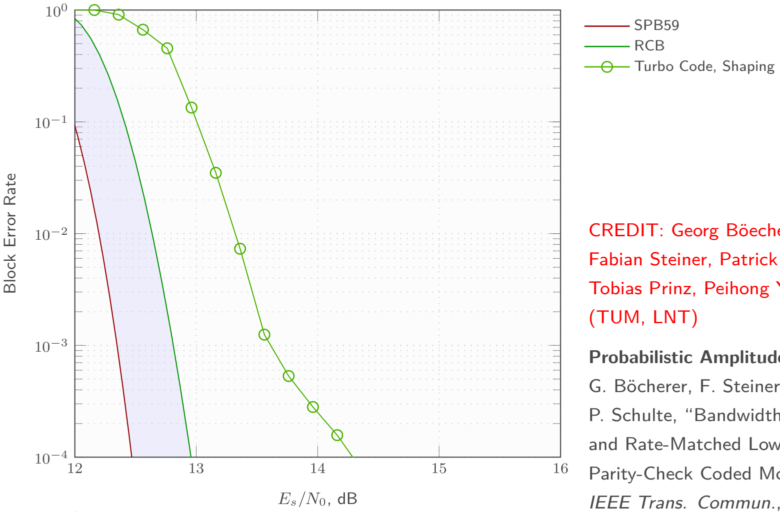
Comparing Rate-$1/2$ Codes at $P_B = 10^{-4}$



Comparing Rate-$1/2$ Codes at $P_B = 10^{-4}$



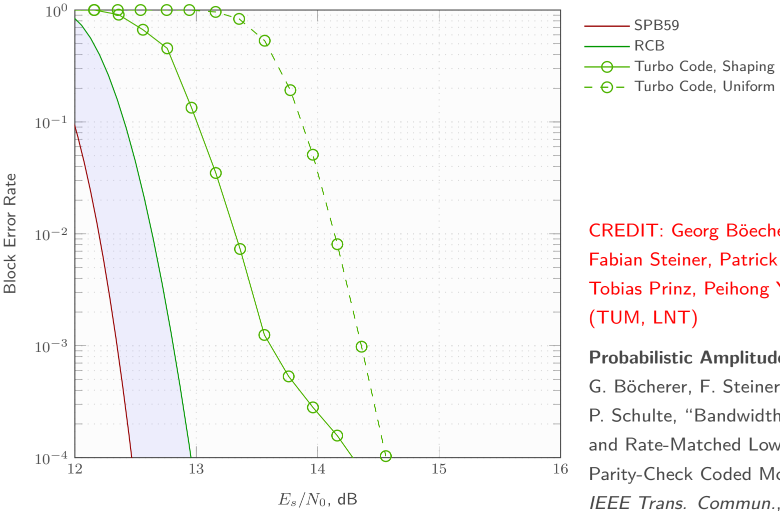
High-Order Modulations: $\eta = 4$ [bpcu], $n = 512$ symbols



CREDIT: Georg Böecherer, Fabian Steiner, Patrick Schulte, Tobias Prinz, Peihong Yuan (TUM, LNT)

Probabilistic Amplitude Shaping: G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth Efficient and Rate-Matched Low-Density Parity-Check Coded Modulation", *IEEE Trans. Commun.*, 2015

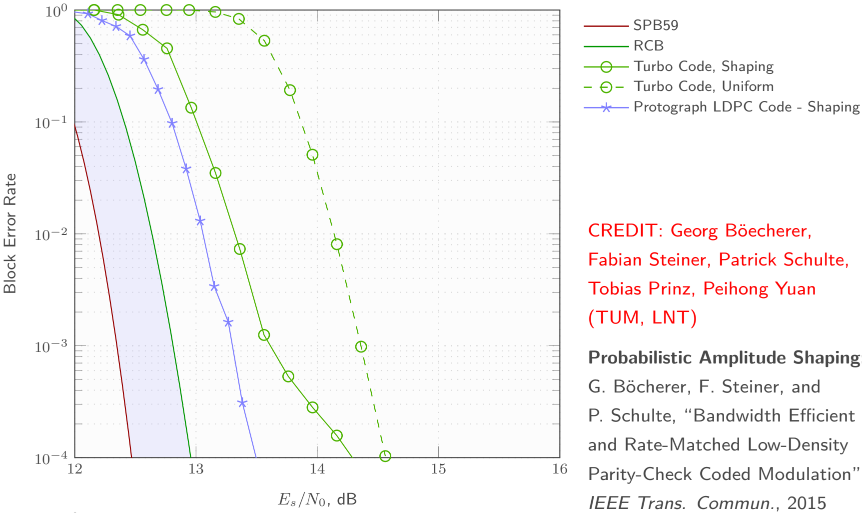
High-Order Modulations: $\eta = 4$ [bpcu], $n = 512$ symbols



CREDIT: Georg Böecherer, Fabian Steiner, Patrick Schulte, Tobias Prinz, Peihong Yuan (TUM, LNT)

Probabilistic Amplitude Shaping: G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth Efficient and Rate-Matched Low-Density Parity-Check Coded Modulation", *IEEE Trans. Commun.*, 2015

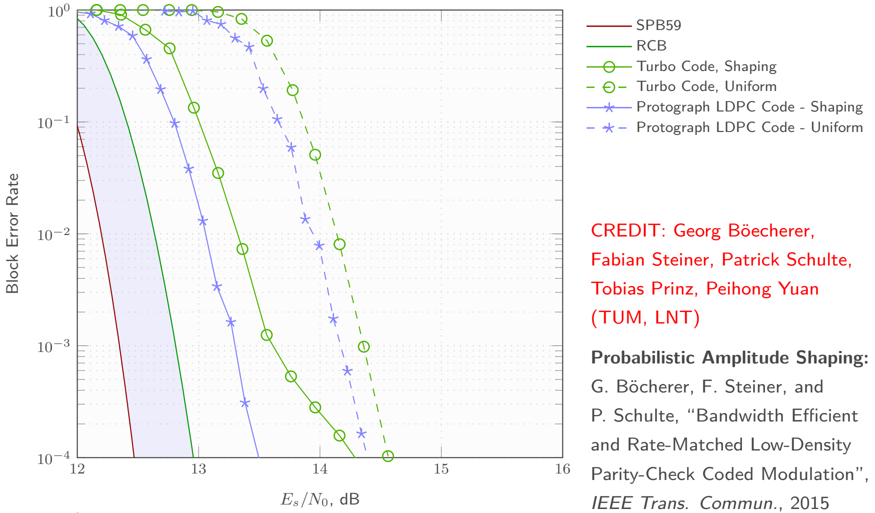
High-Order Modulations: $\eta = 4$ [bpcu], $n = 512$ symbols



CREDIT: Georg Böecherer, Fabian Steiner, Patrick Schulte, Tobias Prinz, Peihong Yuan (TUM, LNT)

Probabilistic Amplitude Shaping: G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth Efficient and Rate-Matched Low-Density Parity-Check Coded Modulation", *IEEE Trans. Commun.*, 2015

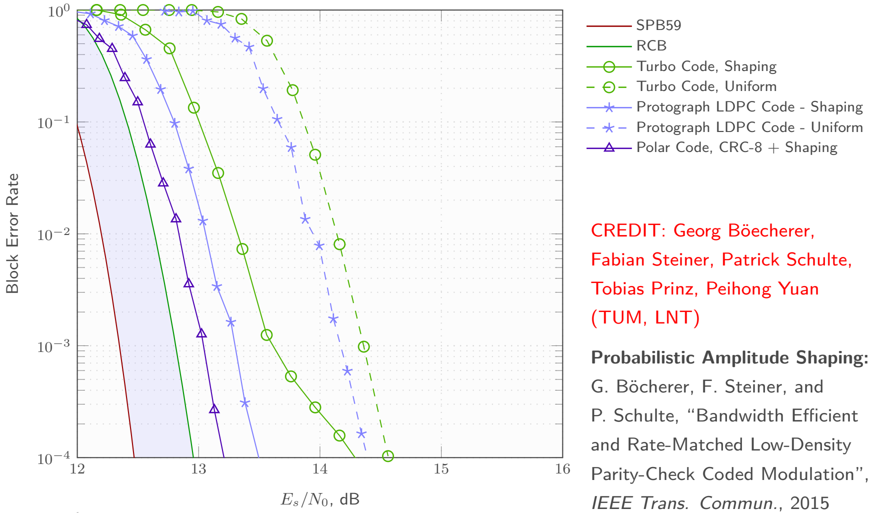
High-Order Modulations: $\eta = 4$ [bpcu], $n = 512$ symbols



CREDIT: Georg Böecherer, Fabian Steiner, Patrick Schulte, Tobias Prinz, Peihong Yuan (TUM, LNT)

Probabilistic Amplitude Shaping: G.Böcherer, F. Steiner, and P. Schulte, "Bandwidth Efficient and Rate-Matched Low-Density Parity-Check Coded Modulation", *IEEE Trans. Commun.*, 2015

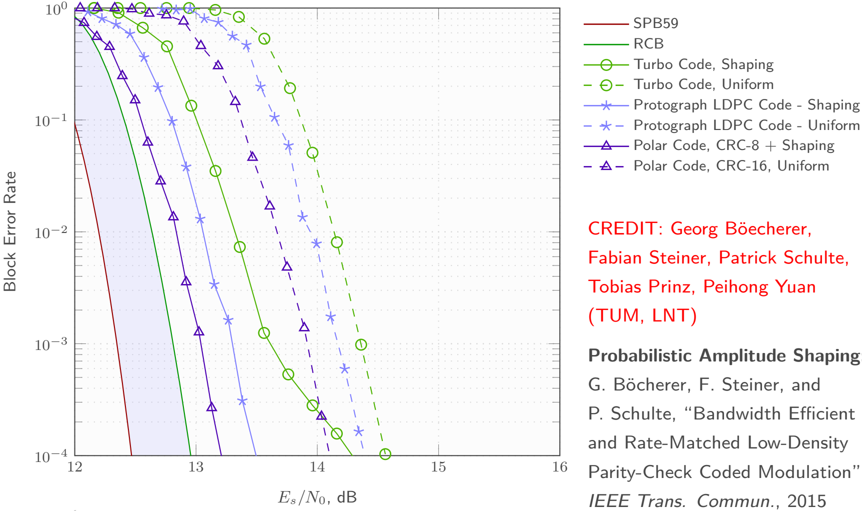
High-Order Modulations: $\eta = 4$ [bpcu], $n = 512$ symbols



CREDIT: Georg Böecherer, Fabian Steiner, Patrick Schulte, Tobias Prinz, Peihong Yuan (TUM, LNT)

Probabilistic Amplitude Shaping: G.Böcherer, F. Steiner, and P. Schulte, "Bandwidth Efficient and Rate-Matched Low-Density Parity-Check Coded Modulation", *IEEE Trans. Commun.*, 2015

High-Order Modulations: $\eta = 4$ [bpcu], $n = 512$ symbols



CREDIT: Georg Böecherer, Fabian Steiner, Patrick Schulte, Tobias Prinz, Peihong Yuan (TUM, LNT)

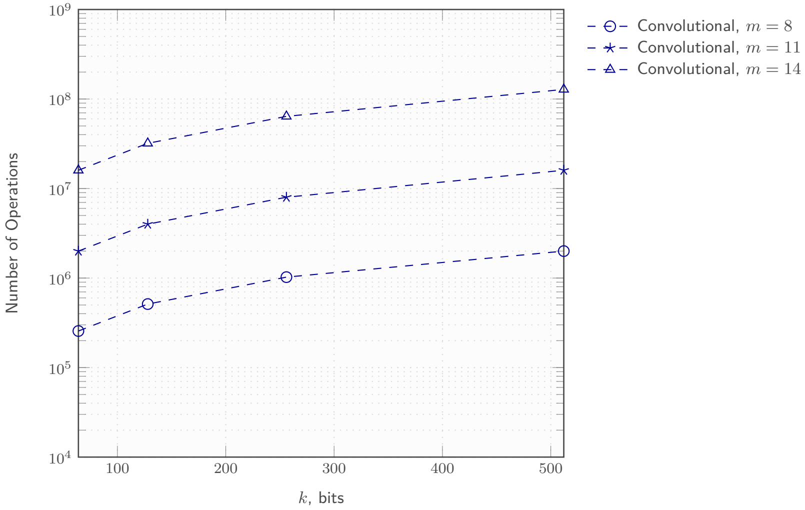
Probabilistic Amplitude Shaping: G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth Efficient and Rate-Matched Low-Density Parity-Check Coded Modulation", *IEEE Trans. Commun.*, 2015

Complexity

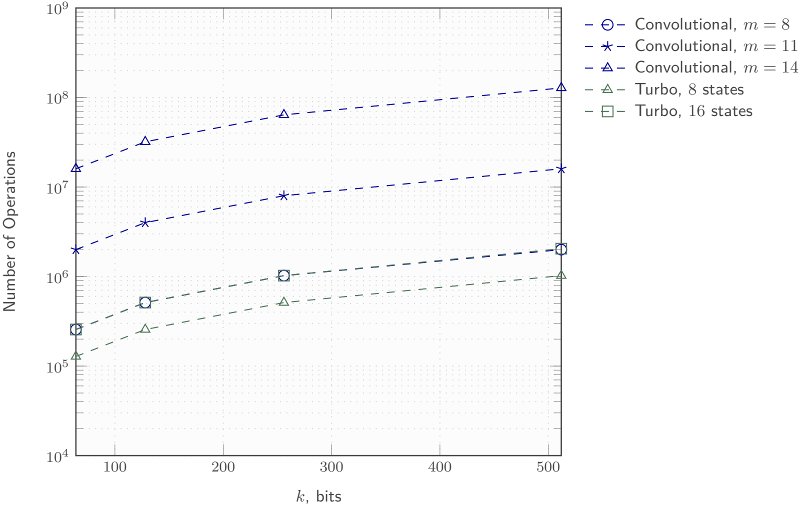
- Model from

O. İșcan, D. Lentner, and W. Xu, “A comparison of channel coding schemes for 5G short message transmission,” in *Proc. Globecom*, 2016

Complexity

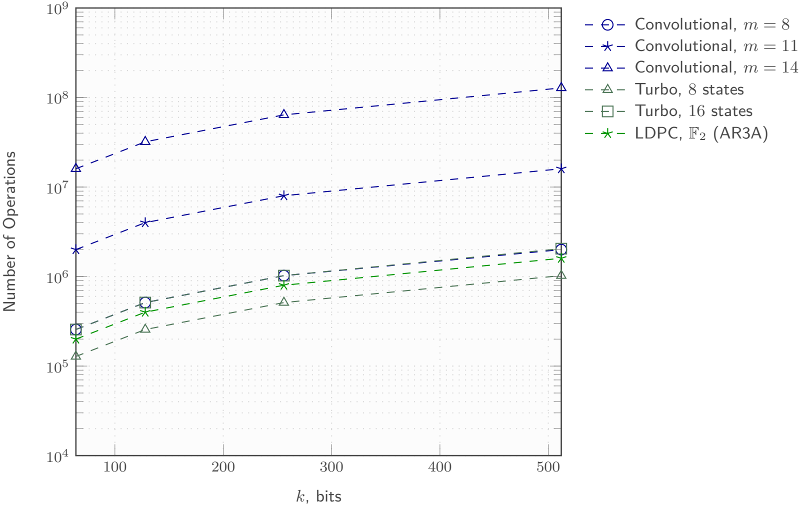

10^9
10^8
10^7
10^6
10^5
10^4
Number of Operations
100
200
300
400
500
k, bits
Convolutional, $m = 8$
Convolutional, $m = 11$
Convolutional, $m = 14$

Complexity



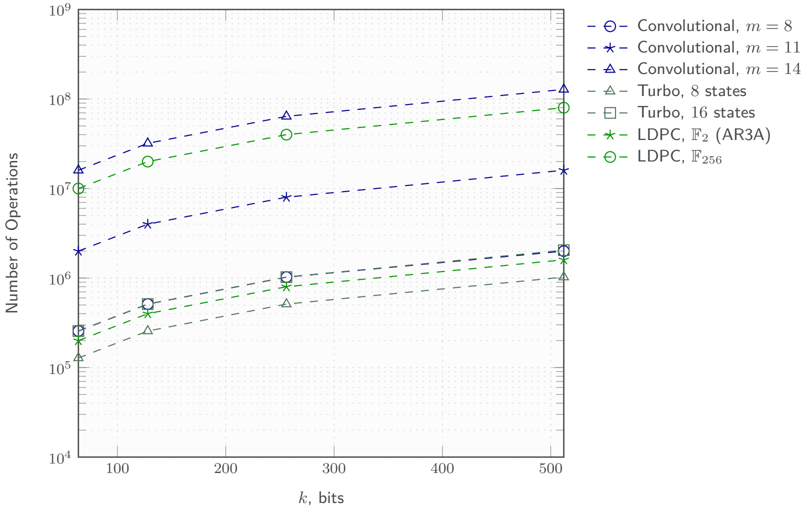
10^9
10^8
10^7
10^6
10^5
10^4
Number of Operations
100
200
300
400
500
k, bits
Convolutional, $m = 8$
Convolutional, $m = 11$
Convolutional, $m = 14$
Turbo, 8 states
Turbo, 16 states

Complexity



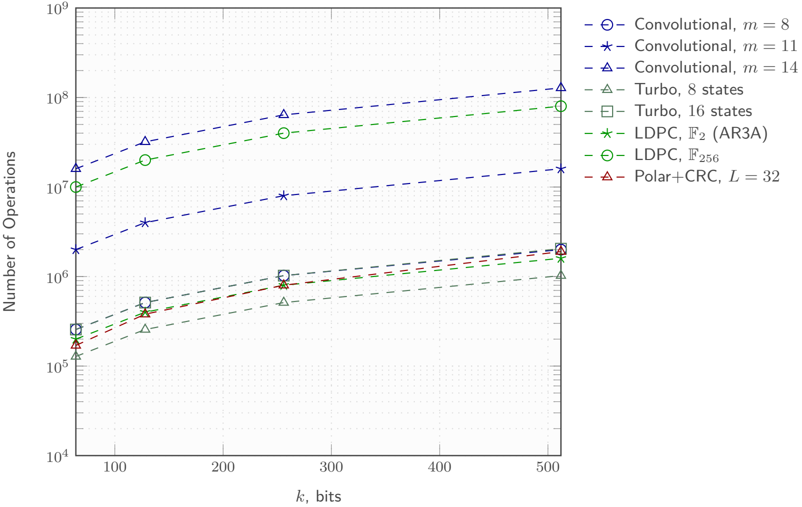
Number of Operations
10^4
10^5
10^6
10^7
10^8
10^9
100
200
300
400
500
k, bits
Convolutional, $m = 8$
Convolutional, $m = 11$
Convolutional, $m = 14$
Turbo, 8 states
Turbo, 16 states
LDPC, $\mathbb{F}_2$ (AR3A)

Complexity



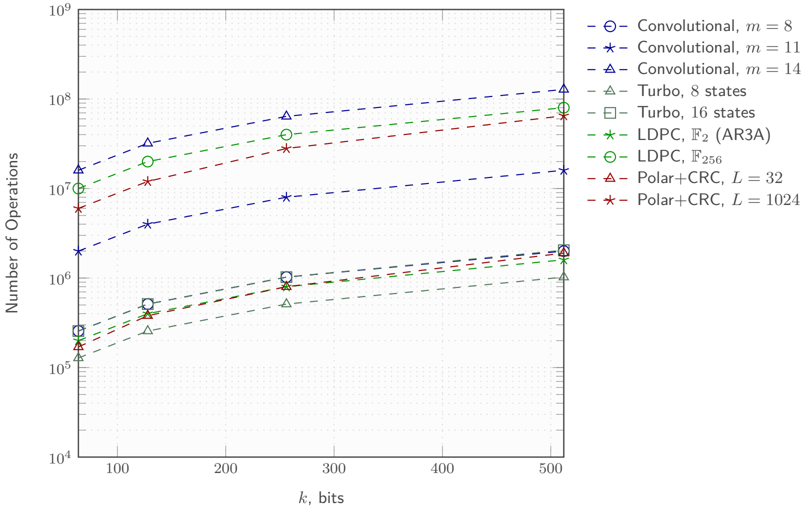
DLR

Complexity



Number of Operations
10^9
10^8
10^7
10^6
10^5
10^4
100
200
300
400
500
k, bits
Convolutional, $m = 8$
Convolutional, $m = 11$
Convolutional, $m = 14$
Turbo, 8 states
Turbo, 16 states
LDPC, $\mathbb{F}_2$ (AR3A)
LDPC, $\mathbb{F}_{256}$
Polar+CRC, $L = 32$

Complexity



10^9
10^8
10^7
10^6
10^5
10^4
Number of Operations
100
200
300
400
500
k, bits
Convolutional, $m = 8$
Convolutional, $m = 11$
Convolutional, $m = 14$
Turbo, 8 states
Turbo, 16 states
LDPC, $\mathbb{F}_2$ (AR3A)
LDPC, $\mathbb{F}_{256}$
Polar+CRC, $L = 32$
Polar+CRC, $L = 1024$

Complete vs. Incomplete: Some Observations on Error Detection

Code Family	Decoding Algorithm	Complete/Incomplete
TBCC	WAVA	"Almost" complete
Linear Block	OSD	Complete
Polar+CRC	List	"Almost" complete for large lists
LDPCC	BP	Incomplete
Turbo	BP	Complete?

Outline

- Preliminaries
- Efficient Short Classical Codes
- Efficient Short Modern Codes
- Two Case Studies
- **Beyond Error Correction**
- Conclusions

The Channel Estimation Problem

- Decoding algorithms often requires some knowledge of the channel state
- Example: Communication over static fading channel

$$\mathbf{y} = h\mathbf{x} + \mathbf{n} \qquad\qquad h \in \mathbb{C}$$

The Channel Estimation Problem

- Decoding algorithms often requires some knowledge of the channel state
- Example: Communication over static fading channel

$$\mathbf{y} = h\mathbf{x} + \mathbf{n} \qquad\qquad h \in \mathbb{C}$$

- Pragmatic (naive?) approach:

The Channel Estimation Problem

- Decoding algorithms often requires some knowledge of the channel state
- Example: Communication over static fading channel

$$\mathbf{y} = h\mathbf{x} + \mathbf{n} \qquad\qquad h \in \mathbb{C}$$

- Pragmatic (naive?) approach:
  - Estimate the channel h first by means of pilot tones (preamble)

The Channel Estimation Problem

- Decoding algorithms often requires some knowledge of the channel state
- Example: Communication over static fading channel

$$\mathbf{y} = h\mathbf{x} + \mathbf{n} \qquad\qquad h \in \mathbb{C}$$

- Pragmatic (naive?) approach:
  - Estimate the channel h first by means of pilot tones (preamble)
  - Decode assuming the channel

$$\mathbf{y} = \hat{h}\mathbf{x} + \mathbf{n}$$

The Channel Estimation Problem

- Decoding algorithms often requires some knowledge of the channel state
- Example: Communication over static fading channel

$$\mathbf{y} = h\mathbf{x} + \mathbf{n} \qquad\qquad h \in \mathbb{C}$$

- Pragmatic (naive?) approach:
  - Estimate the channel h first by means of pilot tones (preamble)
  - Decode assuming the channel

$$\mathbf{y} = \hat{h}\mathbf{x} + \mathbf{n}$$

- For long blocks (and slow fading), the cost of channel estimation is negligible

The Channel Estimation Problem

- Decoding algorithms often requires some knowledge of the channel state
- Example: Communication over static fading channel

$$\mathbf{y} = h\mathbf{x} + \mathbf{n} \qquad\qquad h \in \mathbb{C}$$

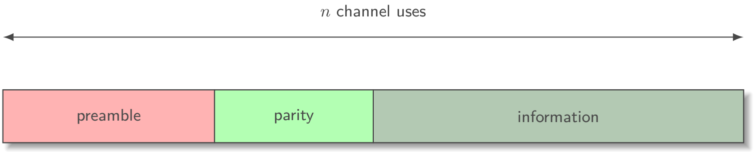
- Pragmatic (naive?) approach:
  - Estimate the channel h first by means of pilot tones (preamble)
  - Decode assuming the channel

$$\mathbf{y} = \hat{h}\mathbf{x} + \mathbf{n}$$

- For long blocks (and slow fading), the cost of channel estimation is negligible
- At short block lengths, the number of channel uses required to get a good channel estimate may be comparable with the number of information bits!

Redundancy vs. Channel Knowledge[50]

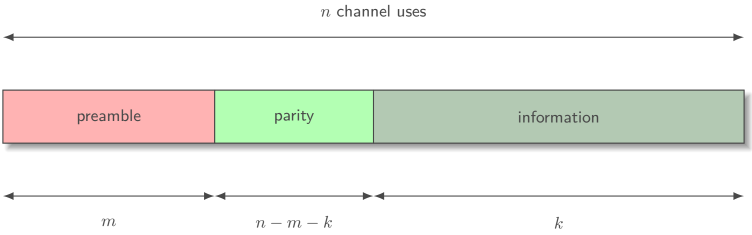
- Problem statement: Transmit k information bits in n channel uses



[50] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, "Short Codes with Mismatched Channel State Information: A Case Study," in *SPAWC*, Jun. 2017

Redundancy vs. Channel Knowledge[50]

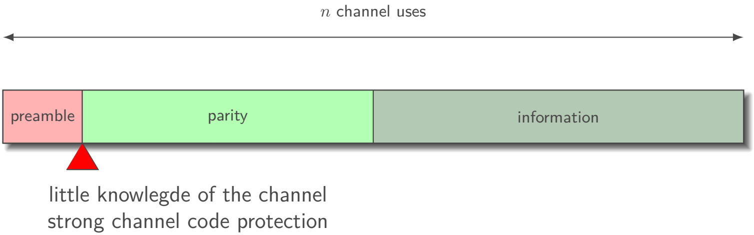
- Problem statement: Transmit k information bits in n channel uses



[50] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, "Short Codes with Mismatched Channel State Information: A Case Study," in *SPAWC*, Jun. 2017

Redundancy vs. Channel Knowledge[50]

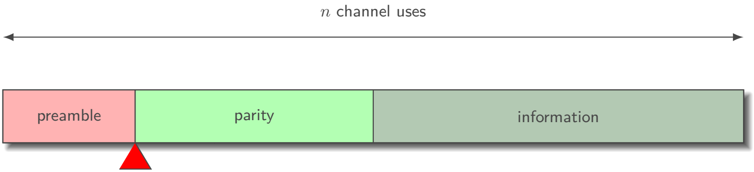
- Problem statement: Transmit k information bits in n channel uses



[50] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, "Short Codes with Mismatched Channel State Information: A Case Study," in *SPAWC*, Jun. 2017

Redundancy vs. Channel Knowledge[50]

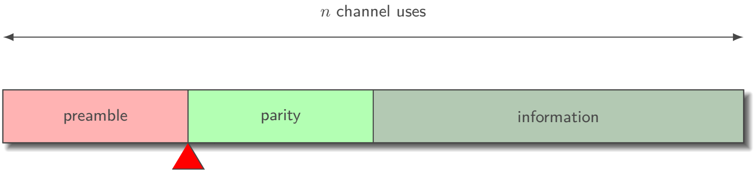
- Problem statement: Transmit k information bits in n channel uses



[50] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, "Short Codes with Mismatched Channel State Information: A Case Study," in *SPAWC*, Jun. 2017

Redundancy vs. Channel Knowledge[50]

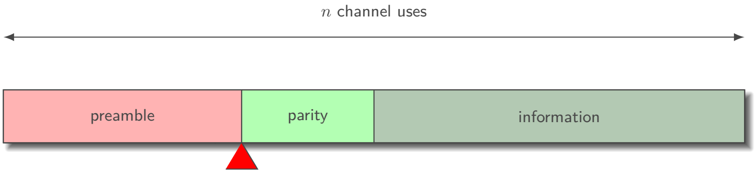
- Problem statement: Transmit k information bits in n channel uses



[50] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, "Short Codes with Mismatched Channel State Information: A Case Study," in *SPAWC*, Jun. 2017

Redundancy vs. Channel Knowledge[50]

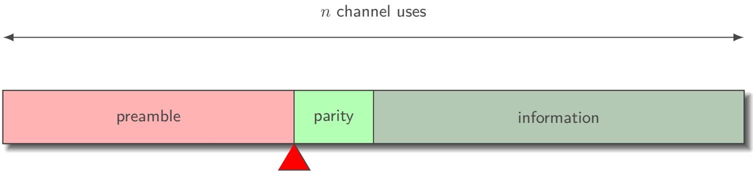
- Problem statement: Transmit k information bits in n channel uses



[50] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, "Short Codes with Mismatched Channel State Information: A Case Study," in *SPAWC*, Jun. 2017

Redundancy vs. Channel Knowledge[50]

- Problem statement: Transmit k information bits in n channel uses

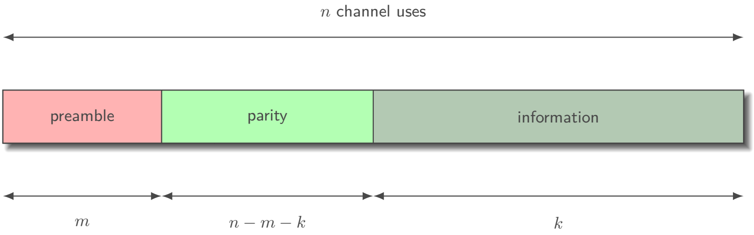


almost perfect knowlegde of the channel
weak channel code protection

[50] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, "Short Codes with Mismatched Channel State Information: A Case Study," in *SPAWC*, Jun. 2017

Redundancy vs. Channel Knowledge[50]

- Problem statement: Transmit k information bits in n channel uses



- Overall rate: $R = k/n$ FIXED
- Channel code rate: $R_c = k/(n-m)$

[50] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, "Short Codes with Mismatched Channel State Information: A Case Study," in *SPAWC*, Jun. 2017

Mismatched Decoding[51][52]

- Suppose the channel estimate $\hat{h}$ to be available by observing the preamble
- The decoder may treat $\hat{h}$ as reliable and proceed by

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}\in\mathcal{C}} p(\mathbf{y}|\mathbf{x}, \hat{h})$$

$$= \arg\max_{\mathbf{x}\in\mathcal{C}} \prod_{i=1}^{n-m} p(y_i|x_i, \hat{h})$$

with

$$p(y|x, \hat{h}) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{1}{2\sigma^2}|y - \hat{h}x|^2\right]$$

[51] G. Kaplan and S. Shamai, "Information rates and error exponents of compound channels with application to antipodal signaling in a fading environment," *AEU. Archiv für Elektronik und Übertragungstechnik*, 1993

[52] N. Merhav, G. Kaplan, A. Lapidoth, and S. S. Shitz, "On information rates for mismatched decoders," *IEEE Trans. Inf. Theory*, 1994

Mismatched Decoding

- Mismatched decoding metric

$$q(y, x; \hat{h}) = p(y|x, \hat{h})$$

- Maximum metric decoding

$$\begin{aligned}\hat{\mathbf{x}} &= \arg\max_{\mathbf{x}\in\mathcal{C}} q(\mathbf{y}, \mathbf{x}; \hat{h}) \\ &= \arg\max_{\mathbf{x}\in\mathcal{C}} \prod_{i=1}^{n-m} q(y_i, x_i; \hat{h})\end{aligned}$$

- Maximum metric decoding = maximum likelihood decoding if $\hat{h} = h$

Gallager's Random Coding Bound
Mismatched Decoding Metric

Upper bound on $P_B(\mathcal{C}^\star;\hat{h})$ under the mismatched metric $q(y,x;\hat{h})$

$$P_B(\mathcal{C}^\star;\hat{h}) \le \mathbb{E}\left[P_B(\mathcal{C};\hat{h})\right] \le P_U(n-m,k;\hat{h})$$

where $$P_U(n-m,k;\hat{h}) = 2^{-(n-m)E_G(R_c;\hat{h})}$$

with $$E_G(R_c;\hat{h})) = \max_{0\le\rho\le 1}\sup_{s>0}\left[E_0(\rho,s;\hat{h}) - \rho R_c\right]$$

and

$$E_0(\rho,s;\hat{h}) = -\log_2 \mathbb{E}\left[\left(\frac{\mathbb{E}\left[q(Y,\tilde{X};\hat{h})^s \middle| Y\right]}{q(Y,X;\hat{h})^s}\right)^\rho\right]$$

Gallager's Random Coding Bound
Mismatched Decoding Metric

If the distribution of $\hat{H}$ is available, then

$$\bar{P}_B(\mathcal{C}^\star) = \mathbb{E}\left[P_B(\mathcal{C}^\star; \hat{H})\right]$$

can be upper bounded by averaging $P_U(n-m, k; \hat{h})$ over the distribution of $\hat{H}$

$$\bar{P}_B(\mathcal{C}^\star) \leq \mathbb{E}\left[2^{-(n-m)E_G(R_c;\hat{H})}\right]$$

Dimensioning the Preamble
Example

- Transmit $k = 256$ bits in $n = 512$ channel uses
- Channel code: $(512, 256)$ irregular repeat accumulate code
- Puncturing m parity bits to leave space for the preamble

Dimensioning the Preamble
Example

- Transmit $k = 256$ bits in $n = 512$ channel uses
- Channel code: $(512, 256)$ irregular repeat accumulate code
- Puncturing m parity bits to leave space for the preamble
- Maximum likelihood channel estimation, h as deterministic unknown parameter

$$\frac{E_b}{N_0} = \frac{|h|^2}{2R\sigma^2}$$

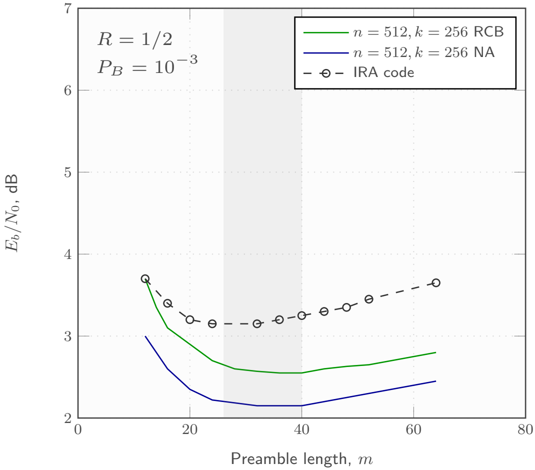
Dimensioning the Preamble
Example

- Transmit $k = 256$ bits in $n = 512$ channel uses
- Channel code: $(512, 256)$ irregular repeat accumulate code
- Puncturing m parity bits to leave space for the preamble
- Maximum likelihood channel estimation, h as deterministic unknown parameter

$$\frac{E_b}{N_0} = \frac{|h|^2}{2R\sigma^2}$$

- Minimize the signal-to-noise ratio required to achieve $P_B = 10^{-3}$

Dimensioning the Preamble



Outline

Thank you!

- Thomas Jerkovits (DLR) for the turbo codes and polar codes simulations (with very short notice...)
- Tudor Ninacs, Thomas Jerkovits (DLR) and Lorenzo Gaudio (Univ. of Parma) for the tail-biting convolutional codes simulations
- The Information Transmission Group at DLR: Mustafa Cemil Coşkun, Federico Clazzer, Giuseppe Cocco, Thomas Jerkovits, Francisco Lazaro Blasco, Balazs Matuz, Estefania Recayte
- Enrico Paolini and Marco Chiani (Univ. of Bologna)
- Gerhard Kramer, Georg Böcherer, Fabian Steiner, Patrick Schulte, Tobias Prinz and Peihong Yuan (TUM-LNT)