Performance Analysis and Anchor Decoding of Staircase Codes

Christian Häger[1,2]

Joint work with: Henry D. Pfister[2], Alexandre Graell i Amat[1], Fredrik Brännström[1], and Erik Agrell[1]

[1] Department of Electrical Engineering, Chalmers University of Technology, Gothenburg

[2] Department of Electrical and Computer Engineering, Duke University, Durham

TU/e, Eindhoven, May 15, 2018

CHALMERS

FORCE
FIBER-OPTIC COMMUNICATIONS
RESEARCH CENTER

Duke
UNIVERSITY

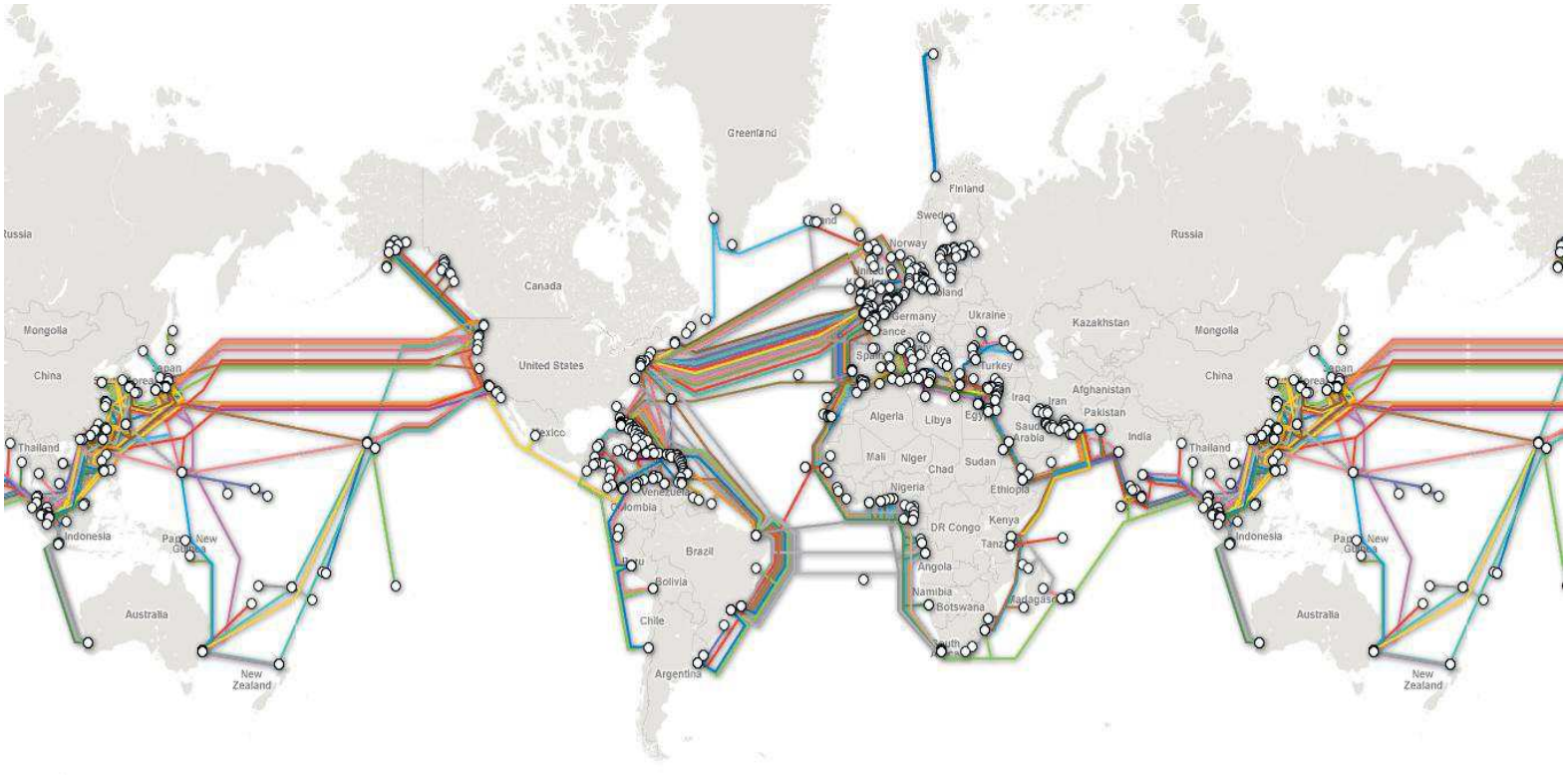
Outline

1. Introduction: Forward-Error Correction for Fiber-Optic Systems
2. Generalized Product Codes and Iterative Decoding
3. Asymptotic Performance Analysis with Density Evolution
4. Avoiding Miscorrections via Anchor Decoding
5. Conclusions

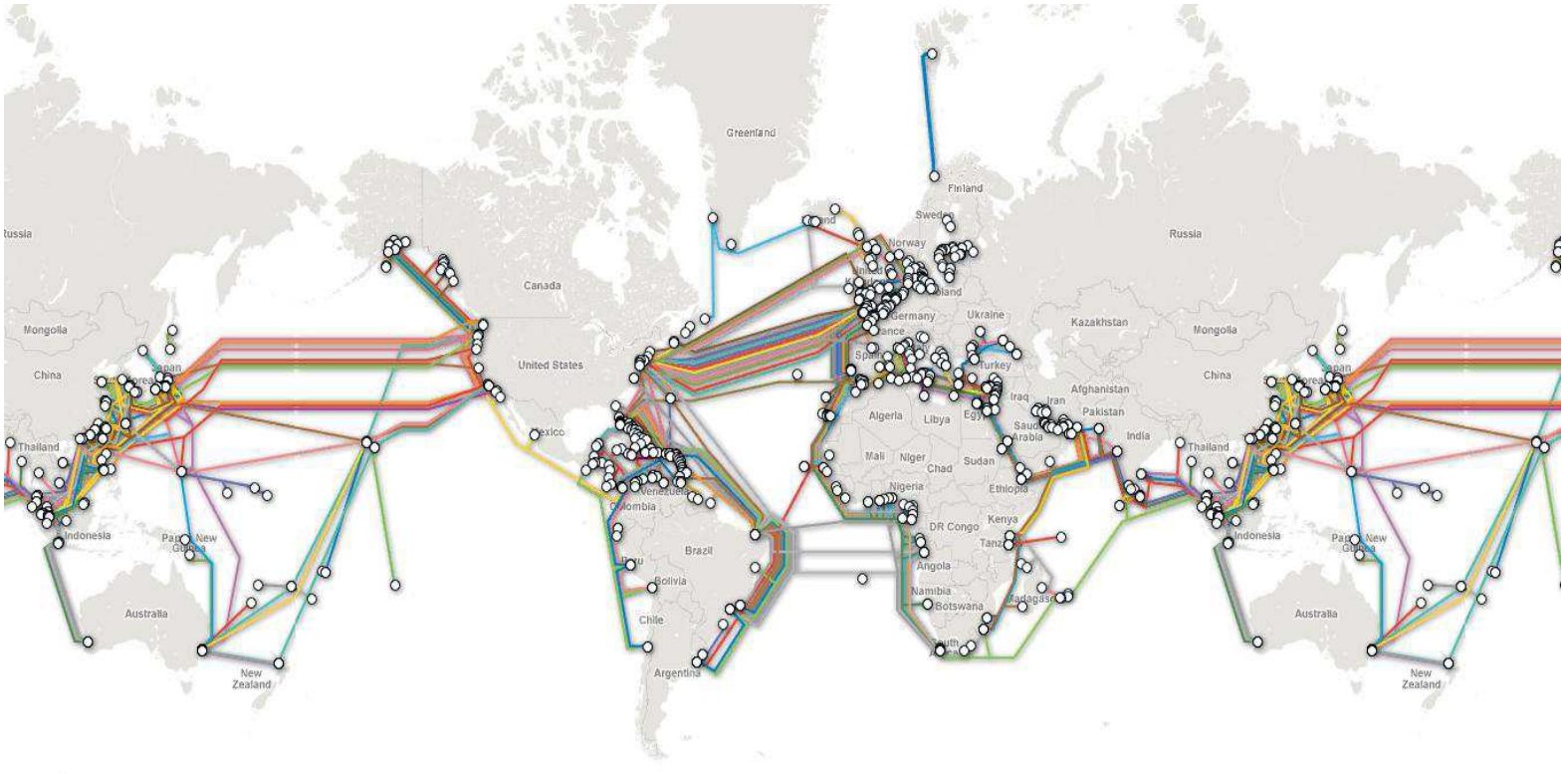
Outline

1. Introduction: Forward-Error Correction for Fiber-Optic Systems
2. Generalized Product Codes and Iterative Decoding
3. Asymptotic Performance Analysis with Density Evolution
4. Avoiding Miscorrections via Anchor Decoding
5. Conclusions

Fiber-Optic Communications



Fiber-Optic Communications



Fiber-optic communication systems enable data traffic over very long distances connecting cities, countries, and continents.

Fiber-Optic Communications

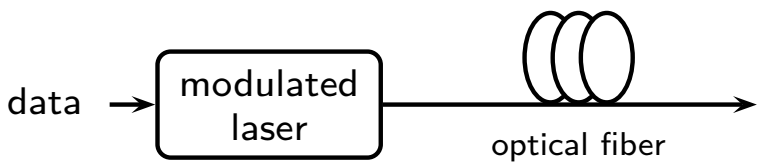
Fiber-Optic Communications

- Long distances result in significant signal attenuation

Fiber-Optic Communications

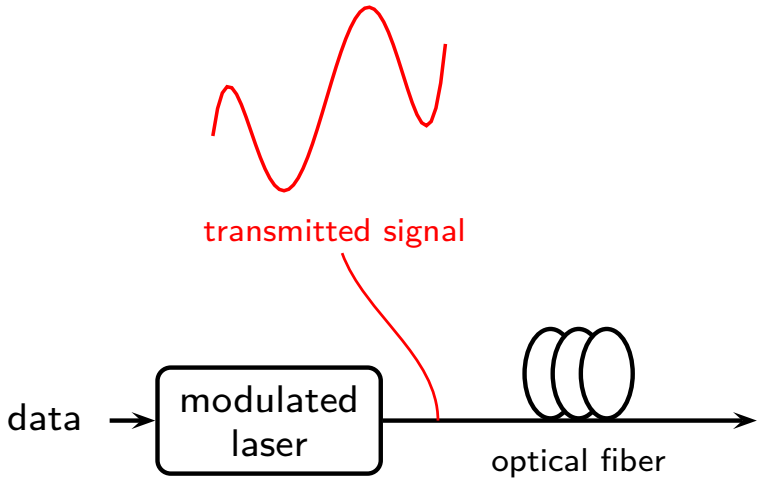
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Fiber-Optic Communications



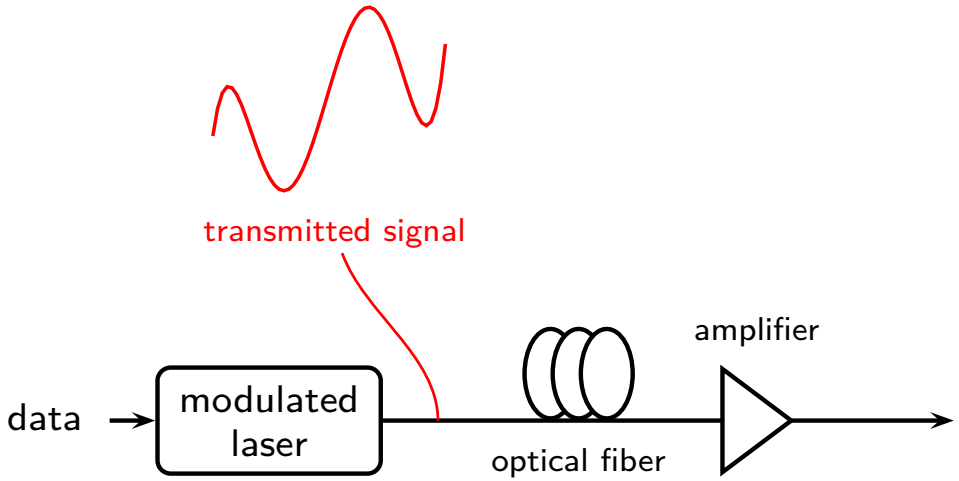
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Fiber-Optic Communications



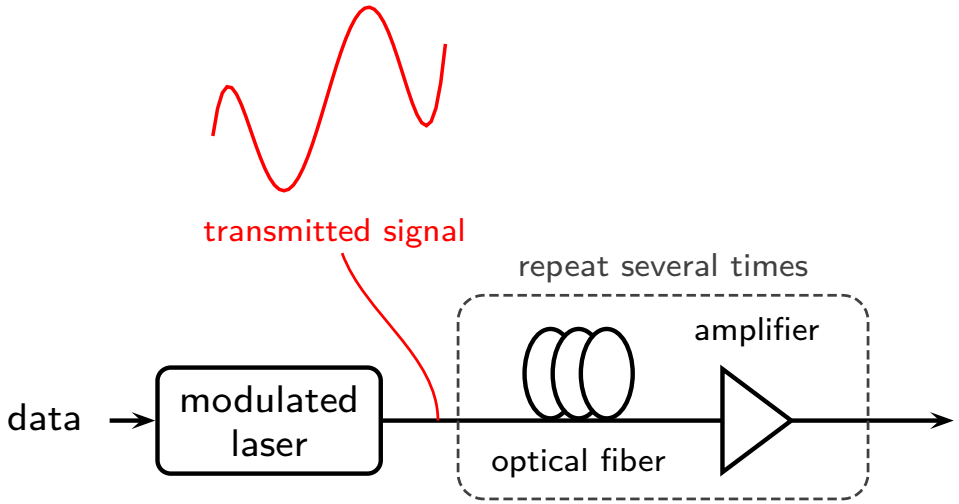
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Fiber-Optic Communications



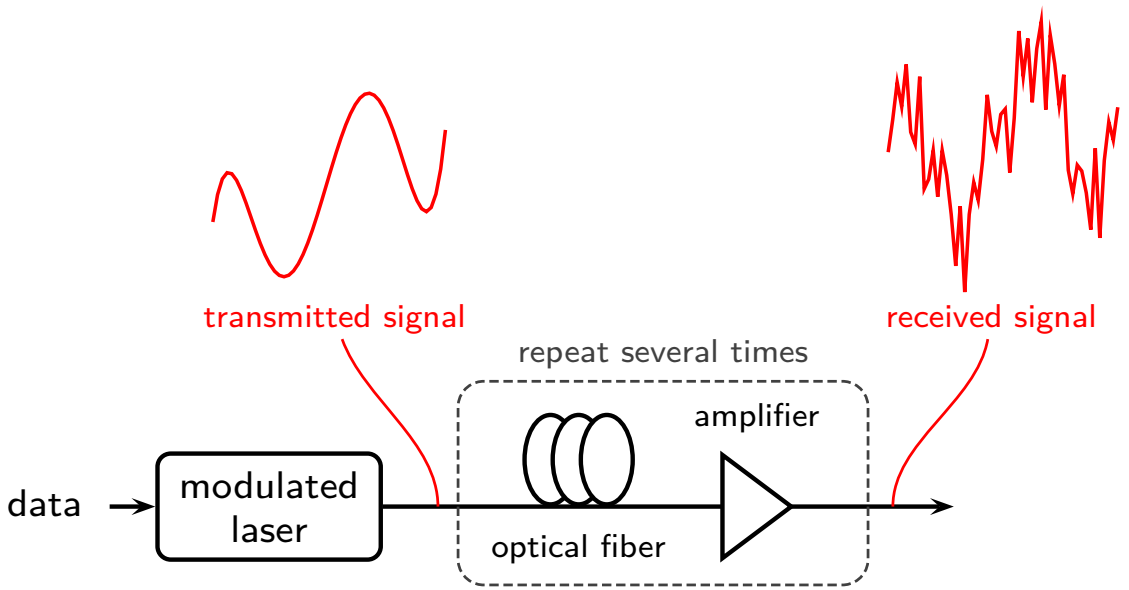
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Fiber-Optic Communications



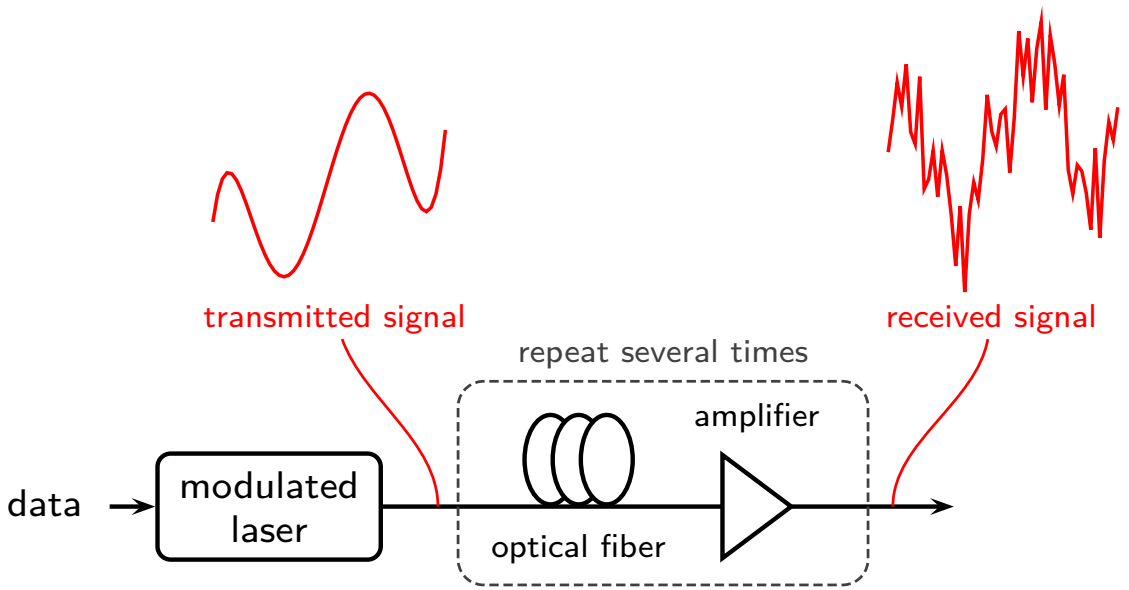
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Fiber-Optic Communications



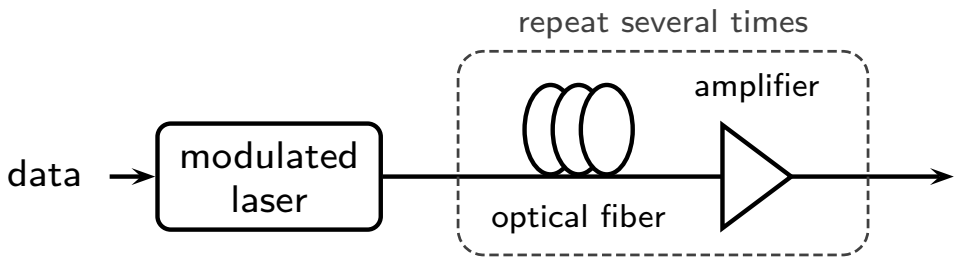
- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise

Fiber-Optic Communications

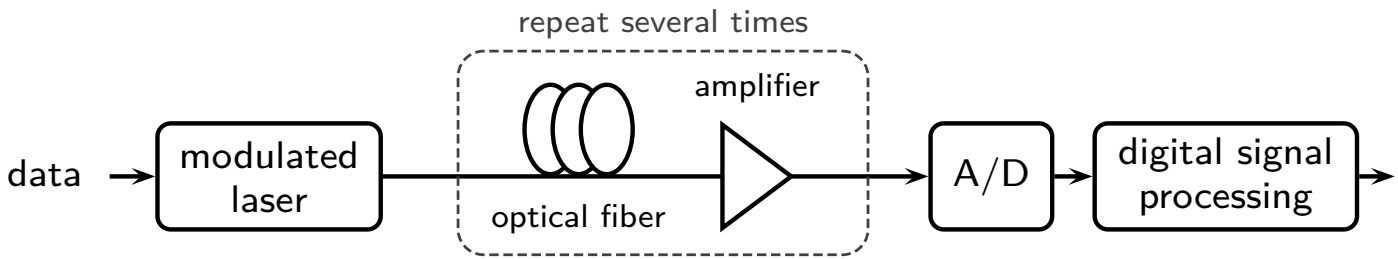


- Long distances result in significant signal attenuation
- Periodic amplification necessary $\Longrightarrow$ random distortions or noise
- Error-correcting codes ensure reliable data transmission

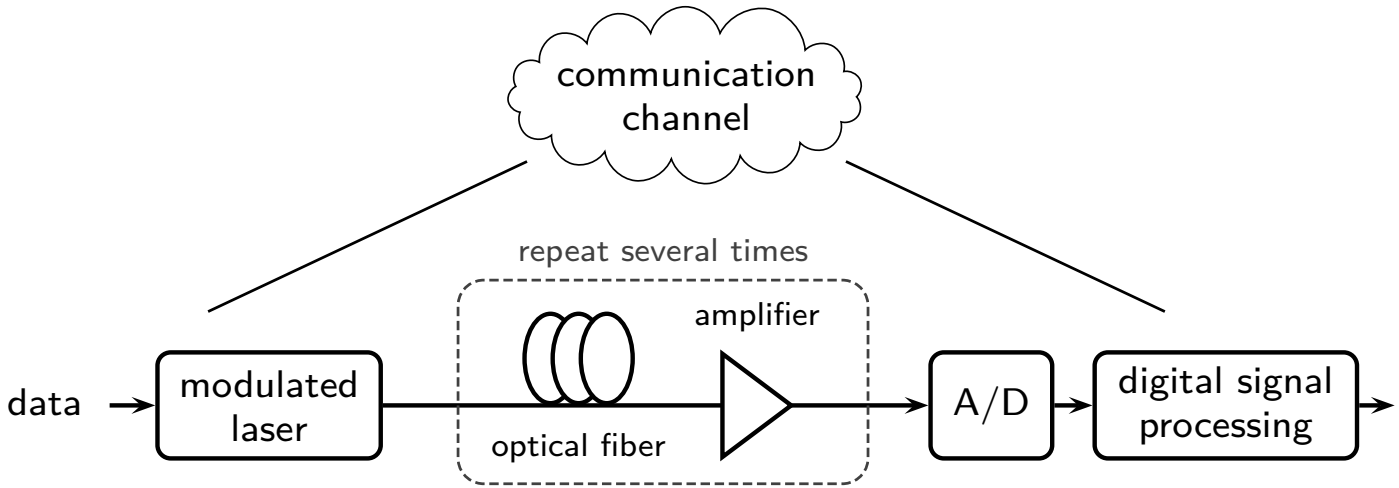
Fiber-Optic Communications



Error-Correcting Codes

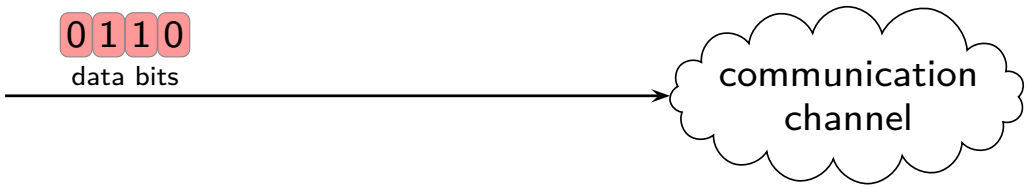


Error-Correcting Codes

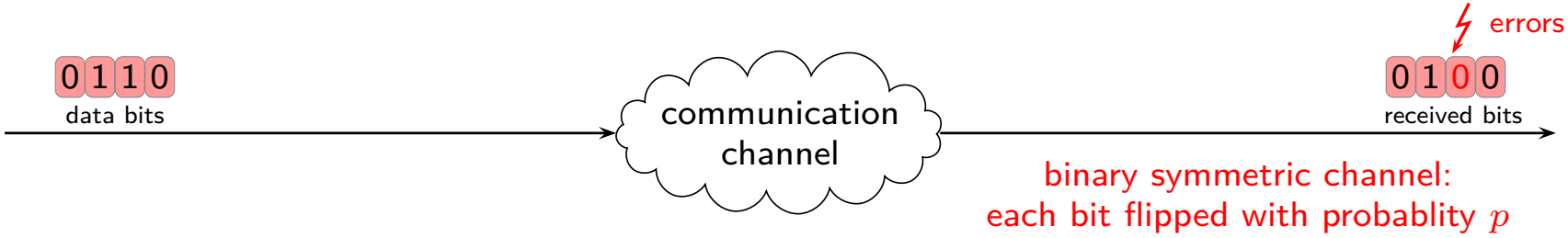


Error-Correcting Codes

Error-Correcting Codes



Error-Correcting Codes

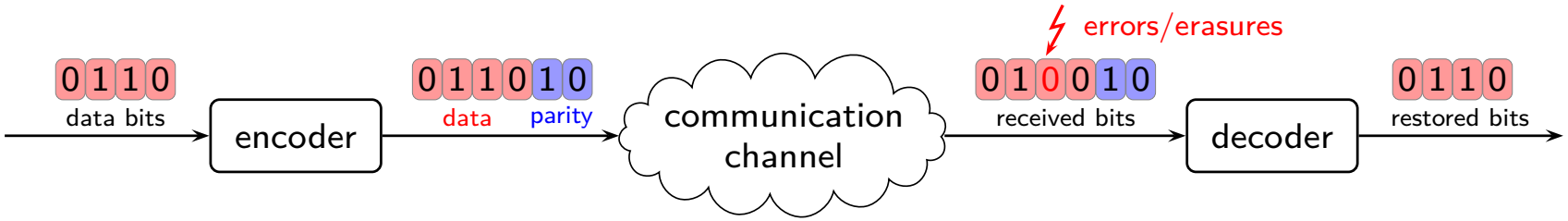


binary symmetric channel:
each bit flipped with probablity p

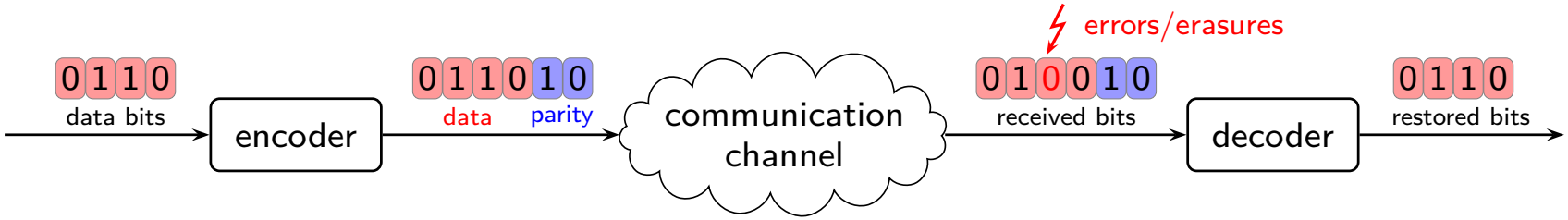
(Motivation: limited soft-information in metro networks, outer clean-up codes, . . .)

Error-Correcting Codes

Error-Correcting Codes



Error-Correcting Codes



Requirements for Fiber-Optic Communications

- Very high throughputs (100 Gigabits per second or higher)
- Very high net coding gains (close-to-capacity performance)
- Very low bit error rates (below 10^{-15})

Error-Correcting Codes

Requirements for Fiber-Optic Communications

- Very high throughputs (100 Gigabits per second or higher)
- Very high net coding gains (close-to-capacity performance)
- Very low bit error rates (below 10^{-15})

This talk

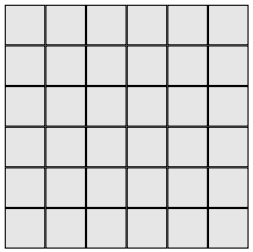
1. Asymptotic performance of deterministic generalized product codes
2. Binary erasure channel vs. binary symmetric channel

Outline

Product Codes and Staircase Codes

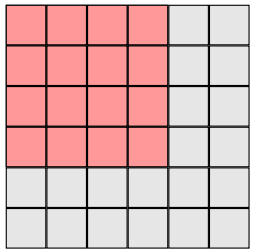
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



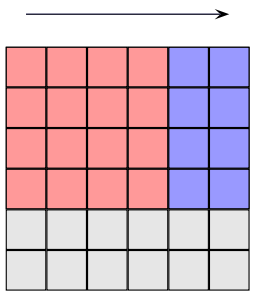
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



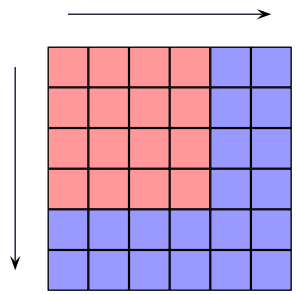
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



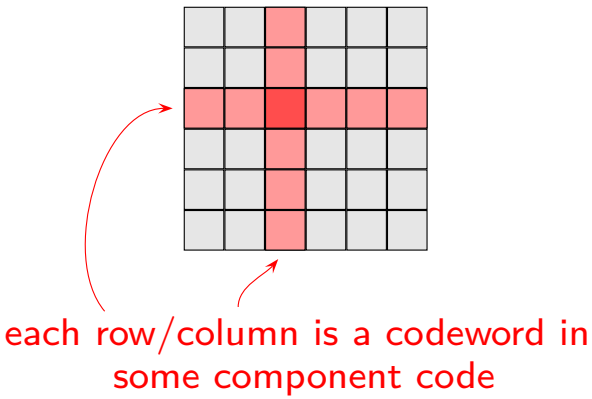
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



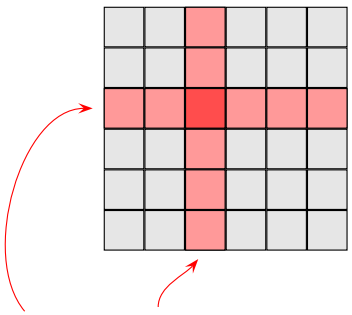
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



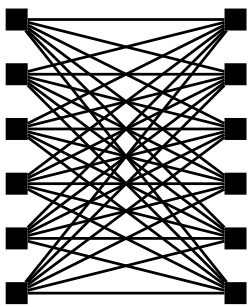
Product Codes and Staircase Codes

rectangular array [Elias, 1954]



each row/column is a codeword in some component code

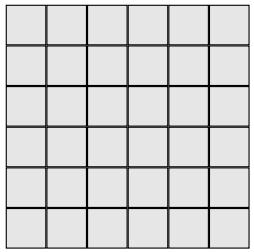
Tanner graph



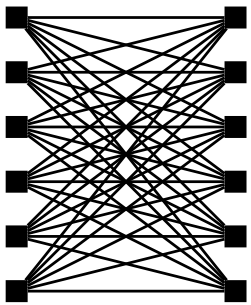
constraint node degree = component code length

Product Codes and Staircase Codes

rectangular array [Elias, 1954]



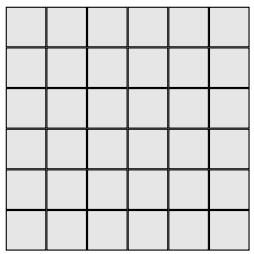
Tanner graph

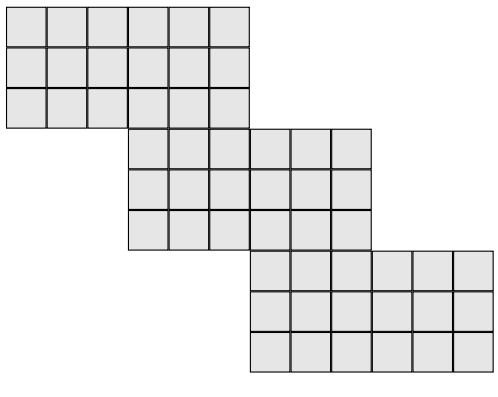


Product Codes and Staircase Codes

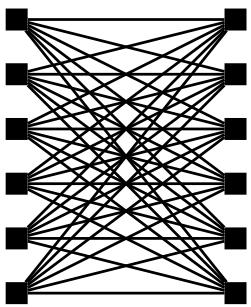
rectangular array [Elias, 1954]

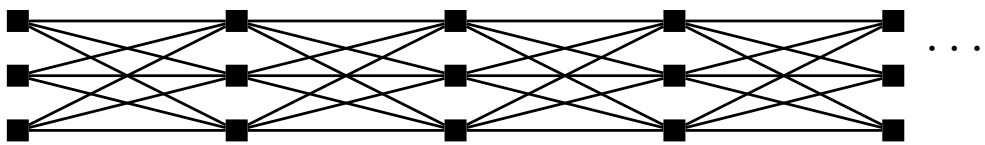
staircase array [Smith et al., 2012]





Tanner graph

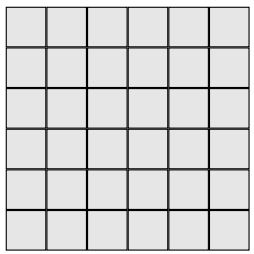


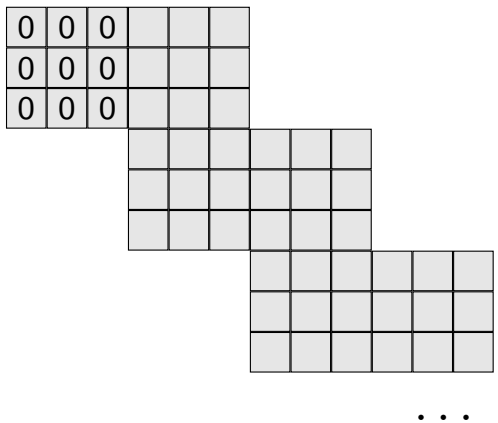


Product Codes and Staircase Codes

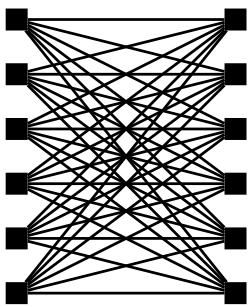
rectangular array [Elias, 1954]

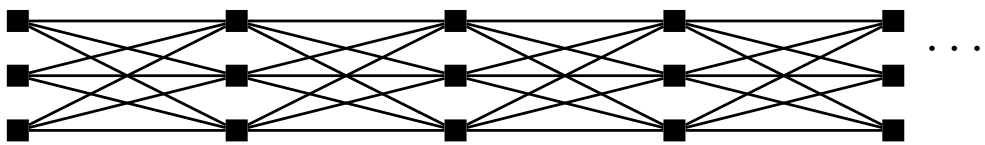
staircase array [Smith et al., 2012]





Tanner graph

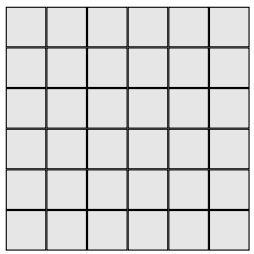


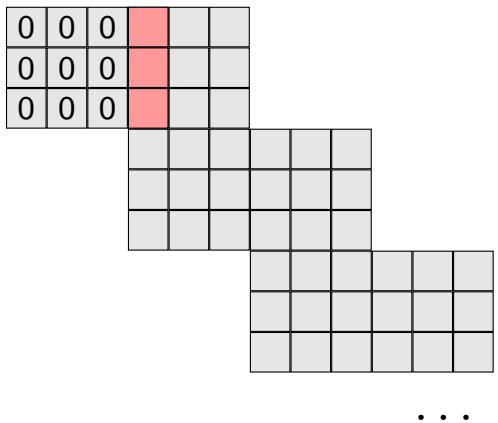


Product Codes and Staircase Codes

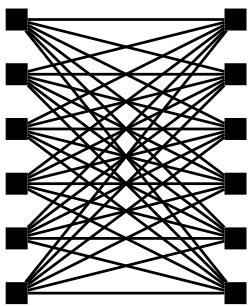
rectangular array [Elias, 1954]

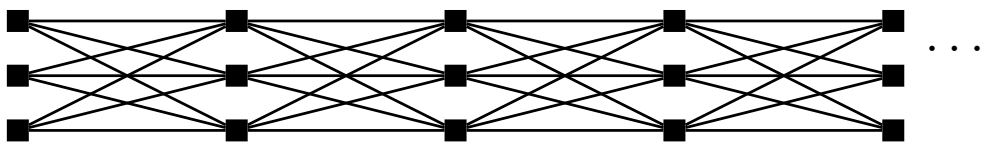
staircase array [Smith et al., 2012]





Tanner graph

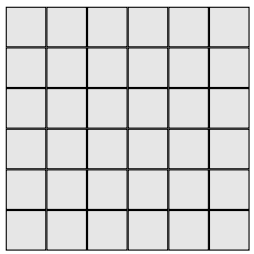


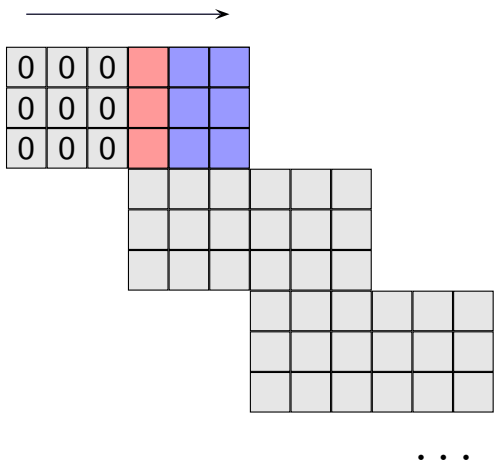


Product Codes and Staircase Codes

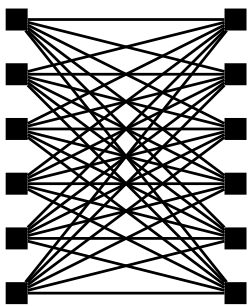
rectangular array [Elias, 1954]

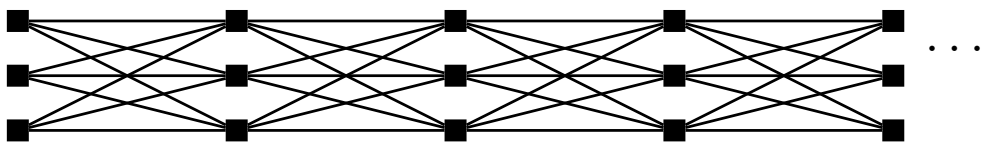
staircase array [Smith et al., 2012]





Tanner graph

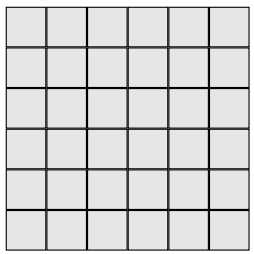


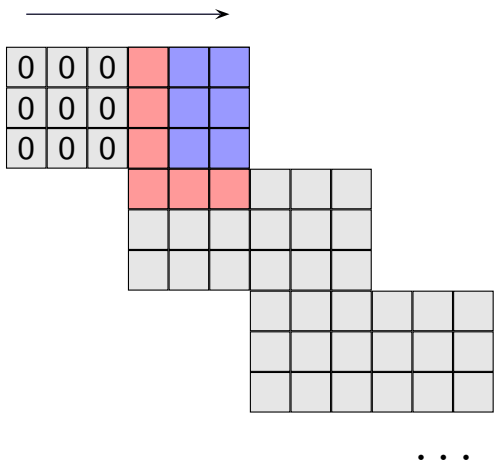


Product Codes and Staircase Codes

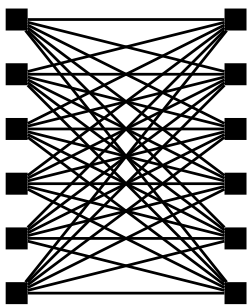
rectangular array [Elias, 1954]

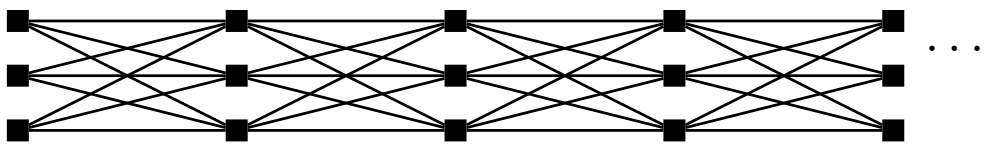
staircase array [Smith et al., 2012]





Tanner graph

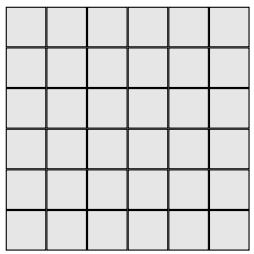


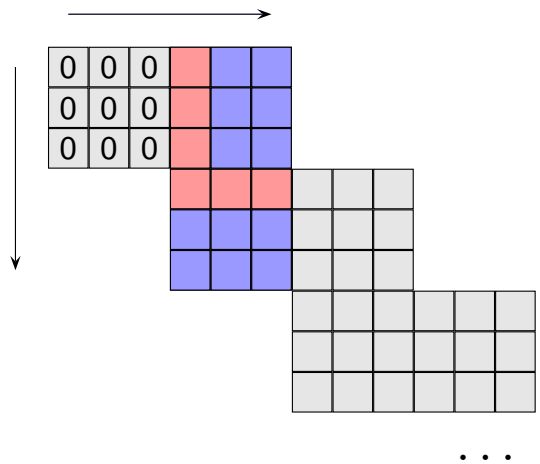


Product Codes and Staircase Codes

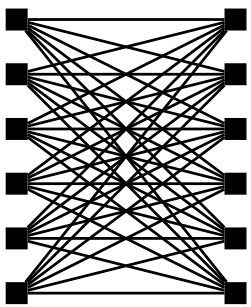
rectangular array [Elias, 1954]

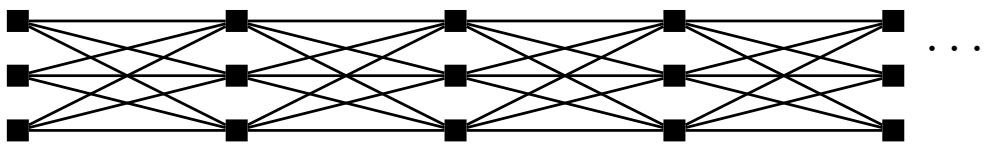
staircase array [Smith et al., 2012]





Tanner graph

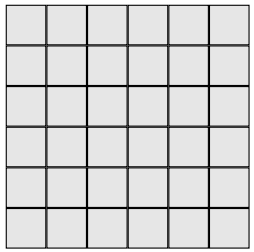


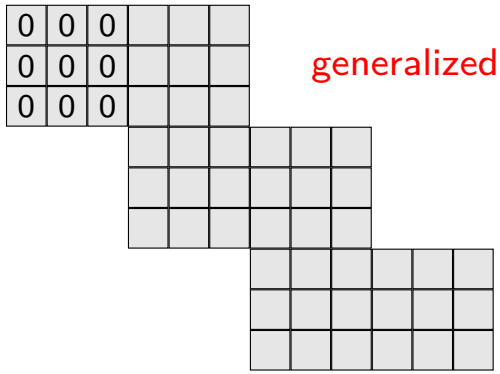


Product Codes and Staircase Codes

rectangular array [Elias, 1954]

staircase array [Smith et al., 2012]

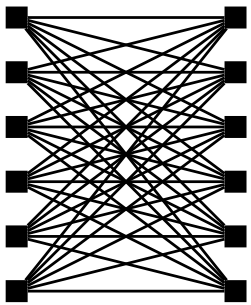


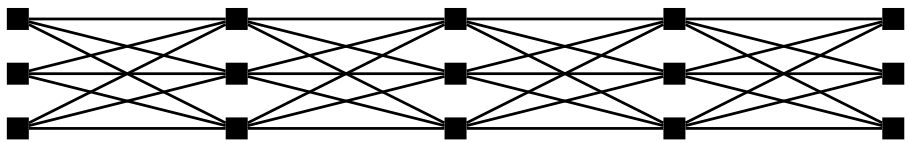


generalized product code (GPC)

. . .

Tanner graph



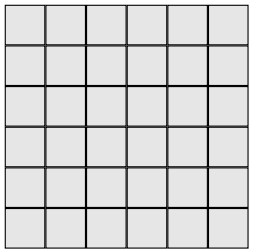


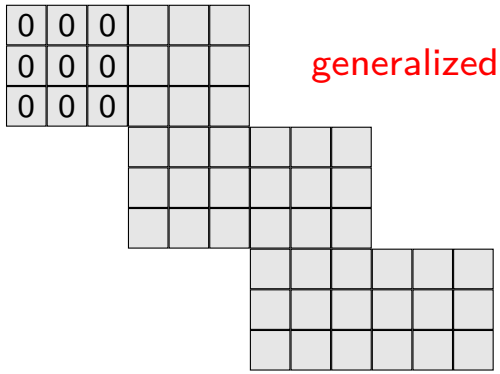
. . .

Product Codes and Staircase Codes

rectangular array [Elias, 1954]

staircase array [Smith et al., 2012]

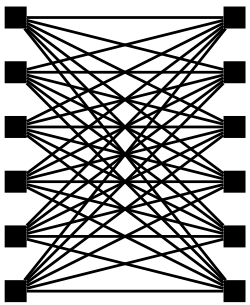


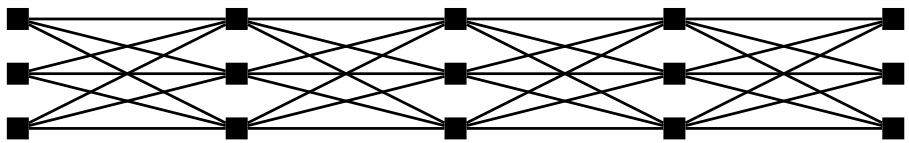


generalized product code (GPC)

. . .

Tanner graph

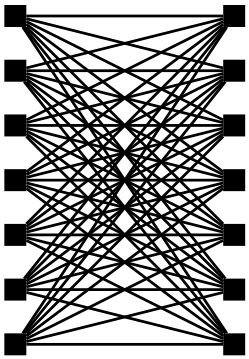


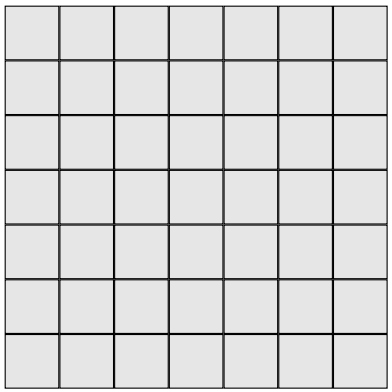


. . .

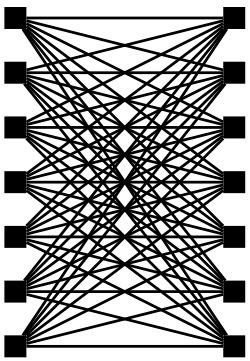
- Deterministic codes with fixed and structured Tanner graph

Iterative Bounded-Distance Decoding



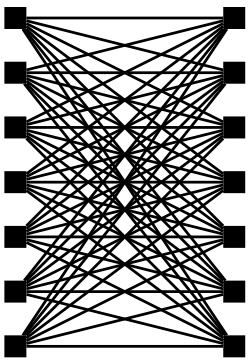


Iterative Bounded-Distance Decoding



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

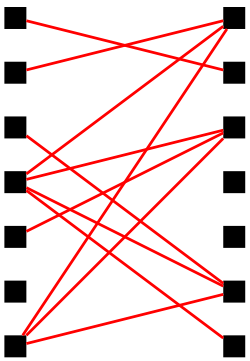
Iterative Bounded-Distance Decoding



0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p

Iterative Bounded-Distance Decoding

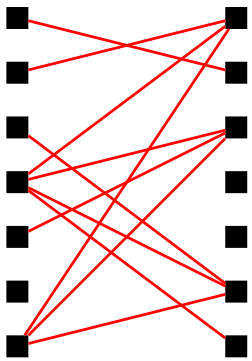


residual graph

0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p

Iterative Bounded-Distance Decoding

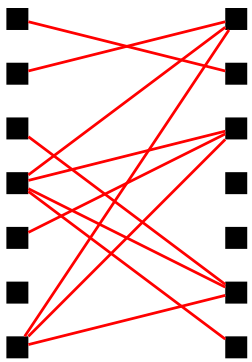


residual graph

0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures

Iterative Bounded-Distance Decoding



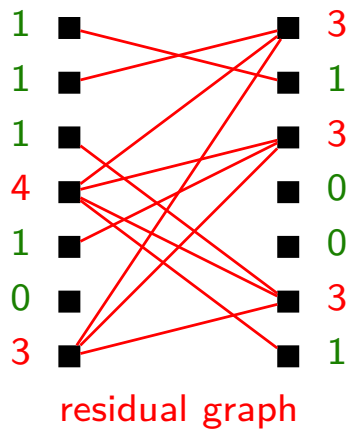
residual graph

0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures
- ℓ iterations of bounded-distance decoding $=$ peeling of vertices with degree $\leq t$ (in parallel)

Iterative Bounded-Distance Decoding

1st iteration $(t = 2)$

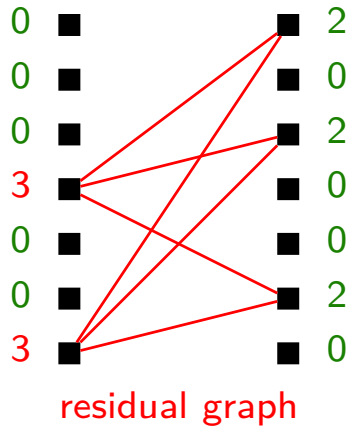


0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures
- ℓ iterations of bounded-distance decoding $=$ peeling of vertices with degree $\leq t$ (in parallel)

Iterative Bounded-Distance Decoding

2nd iteration $(t = 2)$

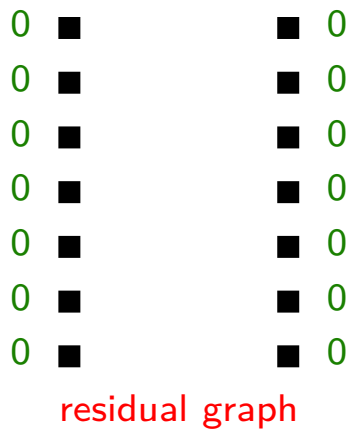


0	1	0	?	0	1	?
0	1	0	1	1	0	1
0	1	0	?	0	1	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	0	?	1	1	?
0	1	0	0	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures
- ℓ iterations of bounded-distance decoding $=$ peeling of vertices with degree $\leq t$ (in parallel)

Iterative Bounded-Distance Decoding

2nd iteration $(t = 2)$

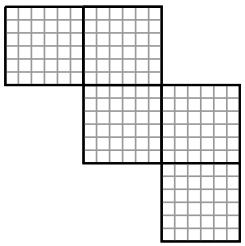


0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	0	1	0	1	0
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	0	0	1	1	1
0	1	0	0	0	1	1

- Codeword transmission over binary erasure channel with erasure probability p
- Each component code corrects $\leq t$ erasures
- ℓ iterations of bounded-distance decoding $=$ peeling of vertices with degree $\leq t$ (in parallel)

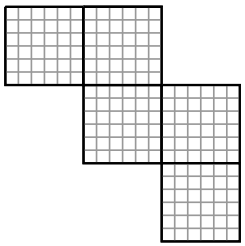
Outline

Performance Prediction

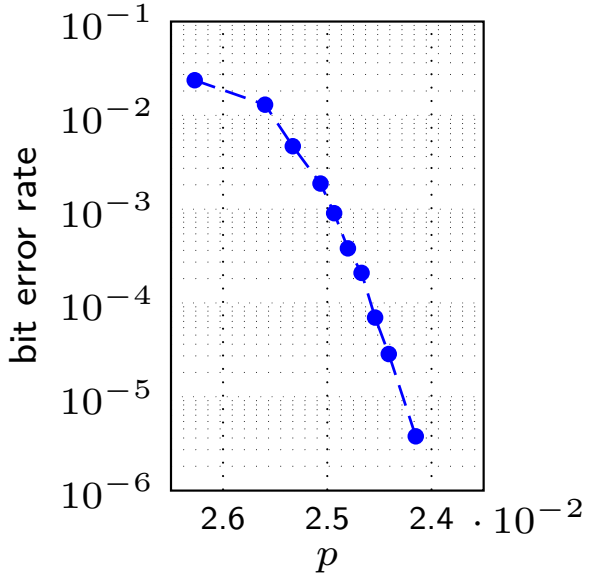


- Example: staircase code with a fixed component code

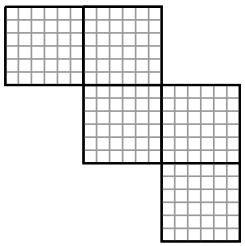
Performance Prediction

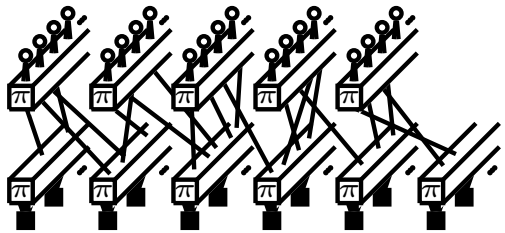


- Example: staircase code with a fixed component code
- Use simulations to predict performance → computationally intensive

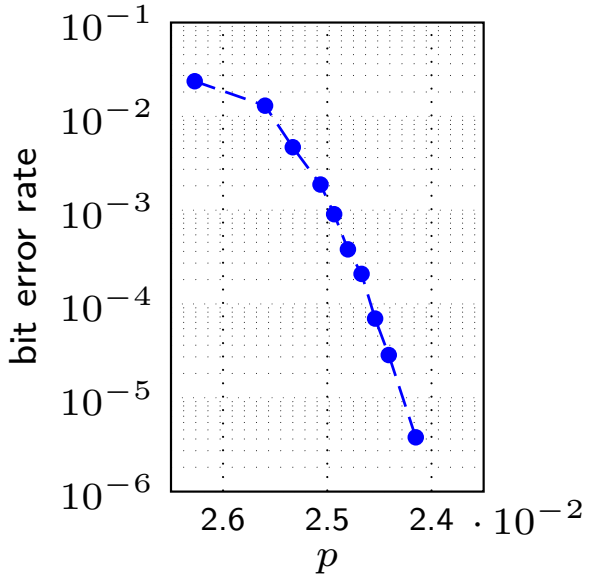


Performance Prediction

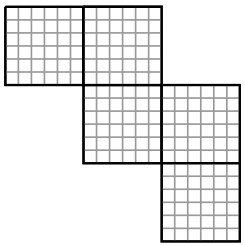


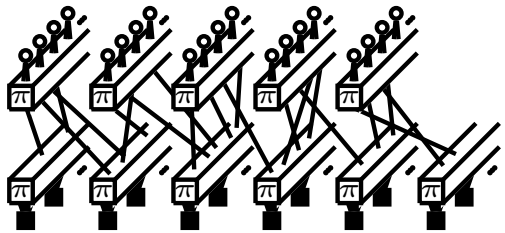


- Example: staircase code with a fixed component code
- Use simulations to predict performance → computationally intensive
- Define randomized set of generalized product codes [Jian et al., 2012], [Zhang et al., 2015]

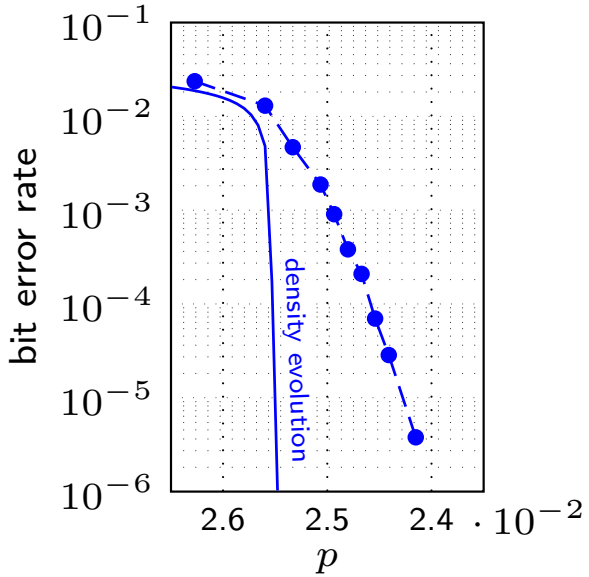


Performance Prediction

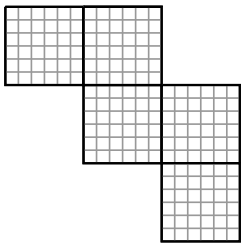


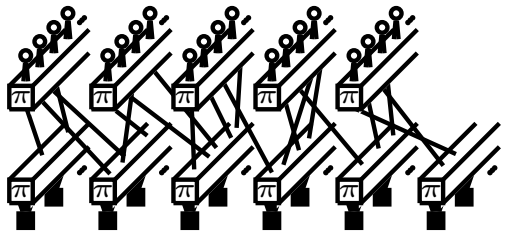


- Example: staircase code with a fixed component code
- Use simulations to predict performance → computationally intensive
- Define randomized set of generalized product codes [Jian et al., 2012], [Zhang et al., 2015]
- Study average performance assuming very long codes via density evolution [Luby et al., 1998], [Richardson and Urbanke, 2001]

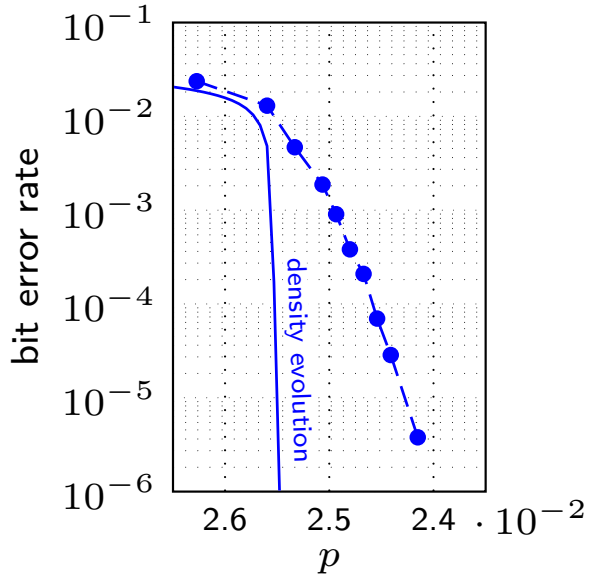


Performance Prediction





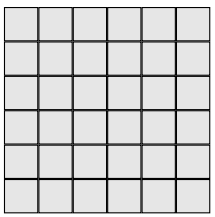
- Example: staircase code with a fixed component code
- Use simulations to predict performance → computationally intensive
- Define randomized set of generalized product codes [Jian et al., 2012], [Zhang et al., 2015]
- Study average performance assuming very long codes via density evolution [Luby et al., 1998], [Richardson and Urbanke, 2001]

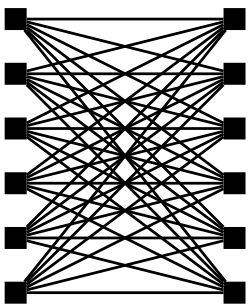


Is it possible to directly analyze deterministic generalized product codes?

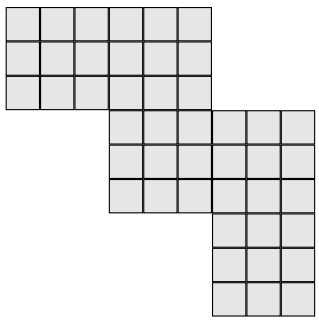
Parameterized Construction of Generalized Product Codes

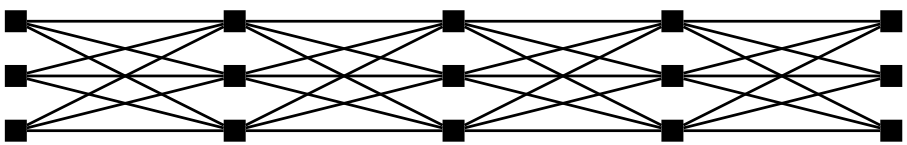
product codes





staircase codes

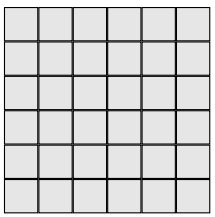


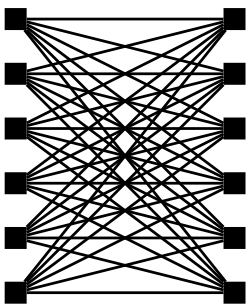


positions: 1 2 3 4 5

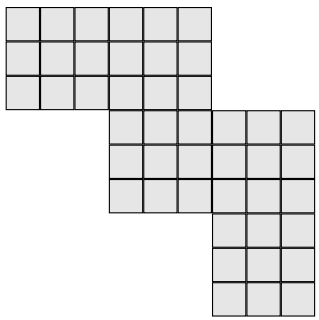
Parameterized Construction of Generalized Product Codes

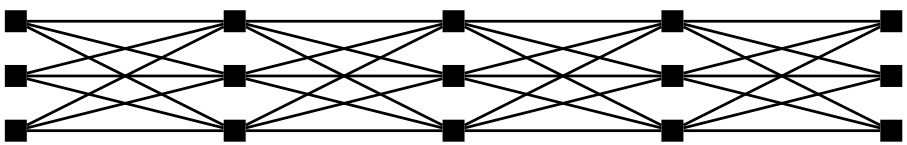
product codes





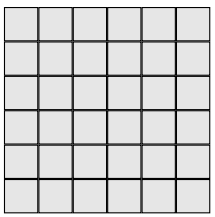
staircase codes

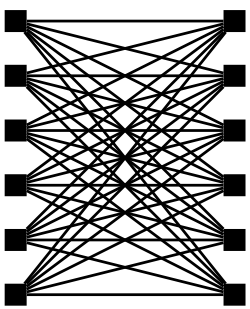




Parameterized Construction of Generalized Product Codes

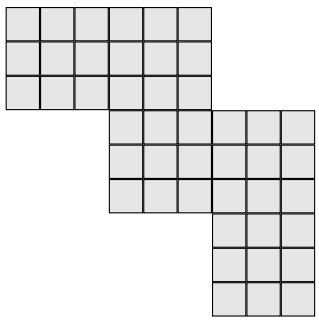
product codes



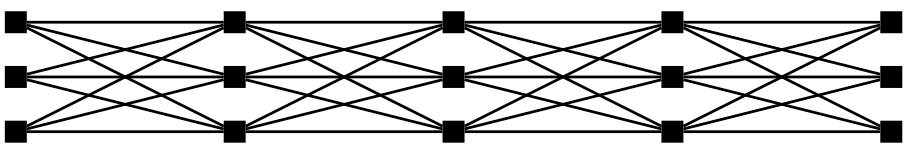


1 $L = 2$

staircase codes

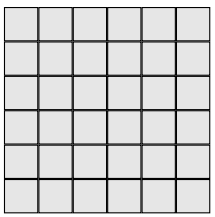


positions: 1 2 3 4 $L = 5$

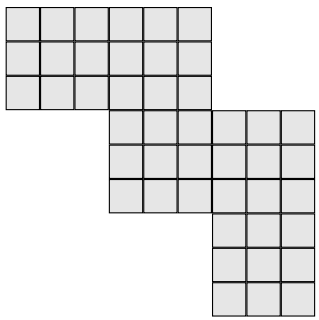


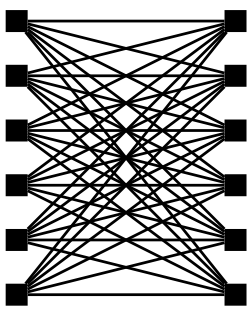
Parameterized Construction of Generalized Product Codes

product codes



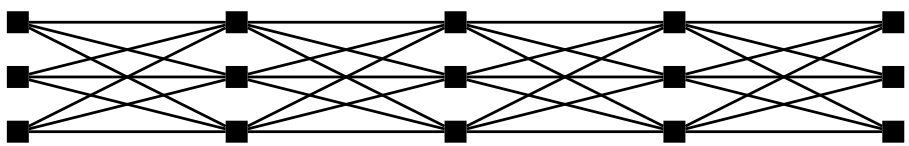
staircase codes





1 $L = 2$

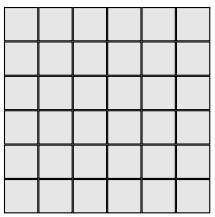
positions: 1 2 3 4 $L = 5$



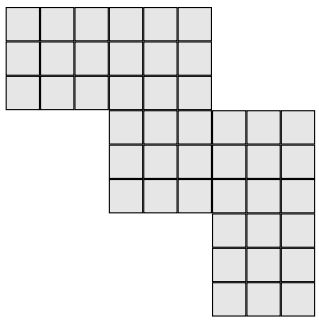
$\boldsymbol{\eta}$: symmetric $L \times L$ matrix that defines graph connectivity

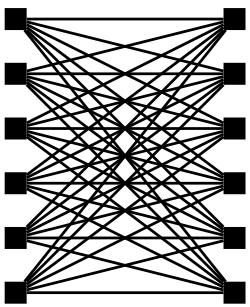
Parameterized Construction of Generalized Product Codes

product codes



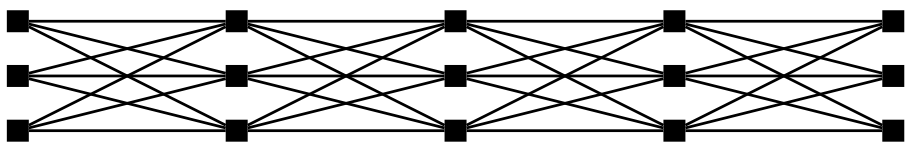
staircase codes





1 $L = 2$

positions: 1 2 3 4 $L = 5$

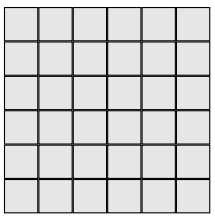


$$\boldsymbol{\eta} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

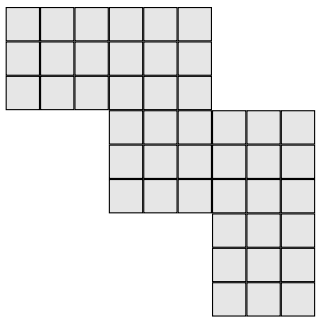
$\boldsymbol{\eta}$: symmetric $L \times L$ matrix that defines graph connectivity

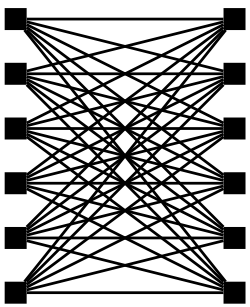
Parameterized Construction of Generalized Product Codes

product codes



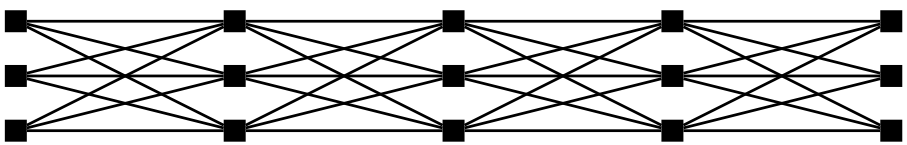
staircase codes





1 $L = 2$

positions: 1 2 3 4 $L = 5$



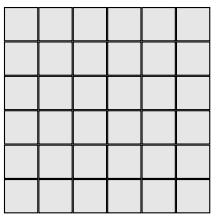
$$\boldsymbol{\eta} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

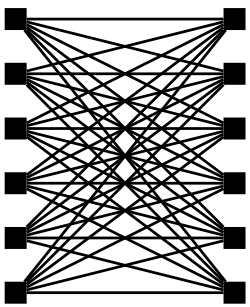
$\boldsymbol{\eta}$: symmetric $L \times L$ matrix that defines graph connectivity

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

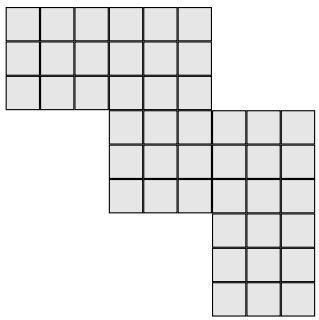
Parameterized Construction of Generalized Product Codes

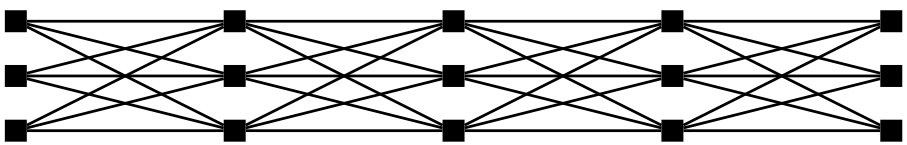
product codes





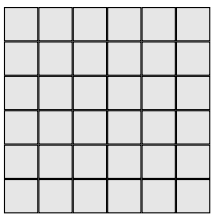
staircase codes



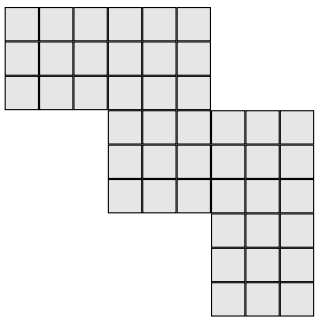


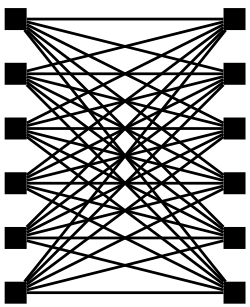
Parameterized Construction of Generalized Product Codes

product codes

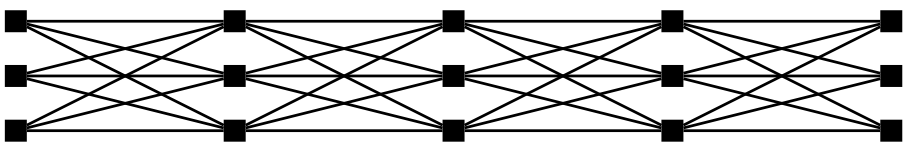


staircase codes



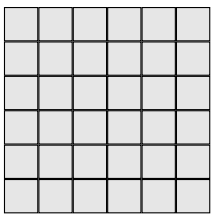


n: "problem size", proportional to the number of constraint nodes

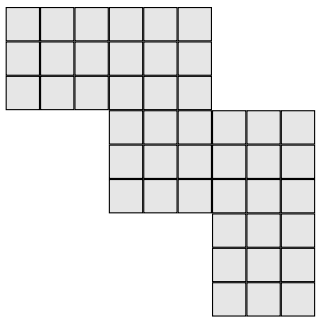


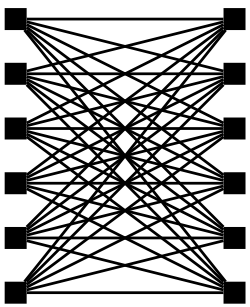
Parameterized Construction of Generalized Product Codes

product codes

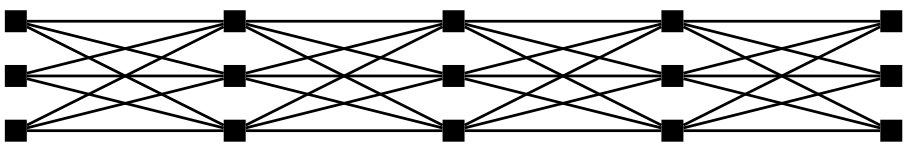


staircase codes





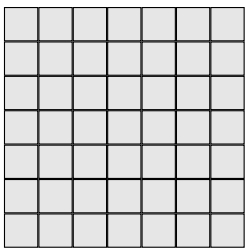
n: "problem size", proportional to the number of constraint nodes



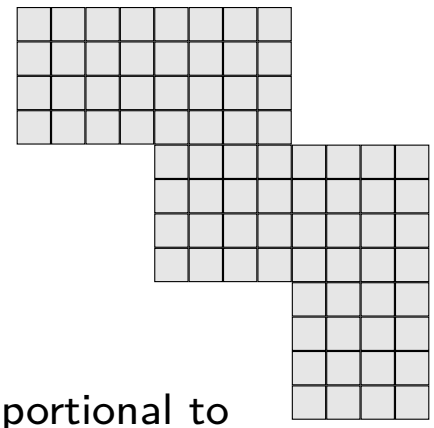
increasing n

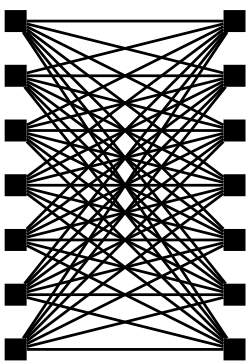
Parameterized Construction of Generalized Product Codes

product codes

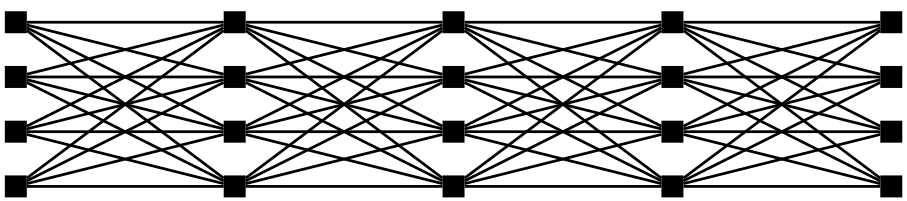


staircase codes





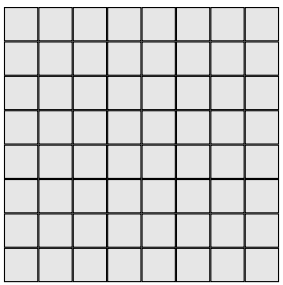
n: "problem size", proportional to the number of constraint nodes



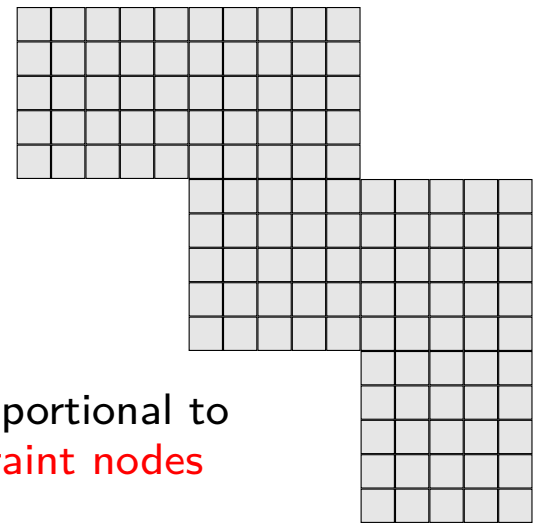
increasing n

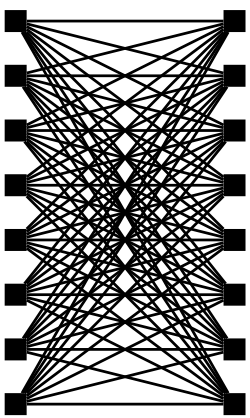
Parameterized Construction of Generalized Product Codes

product codes

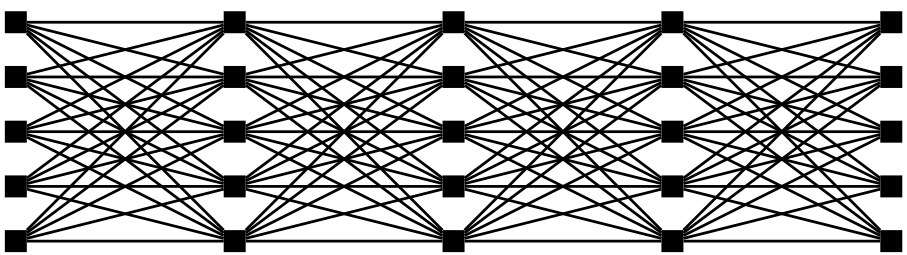


staircase codes





n: "problem size", proportional to the number of constraint nodes

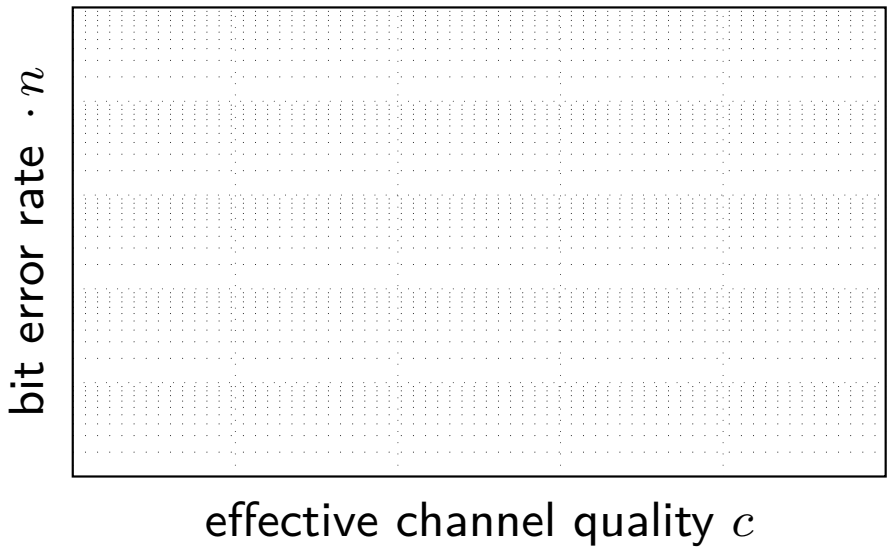


increasing n

Density Evolution

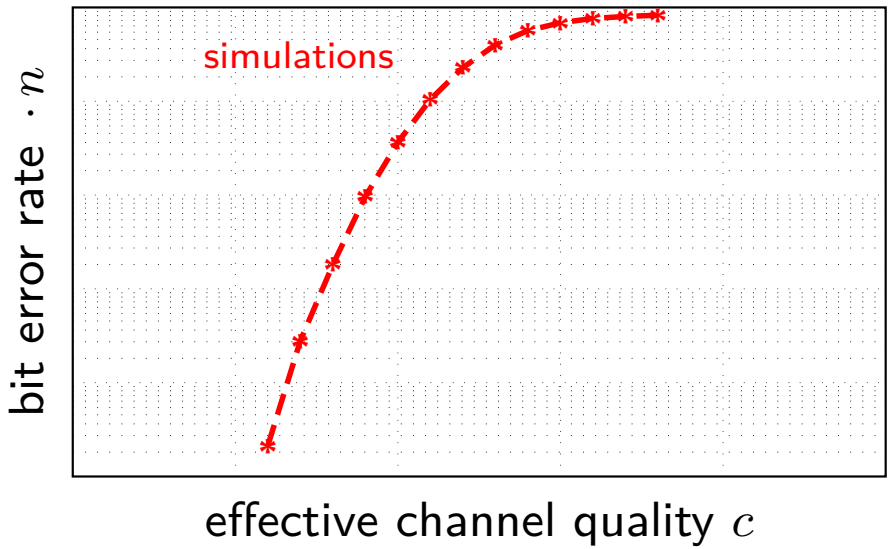
Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



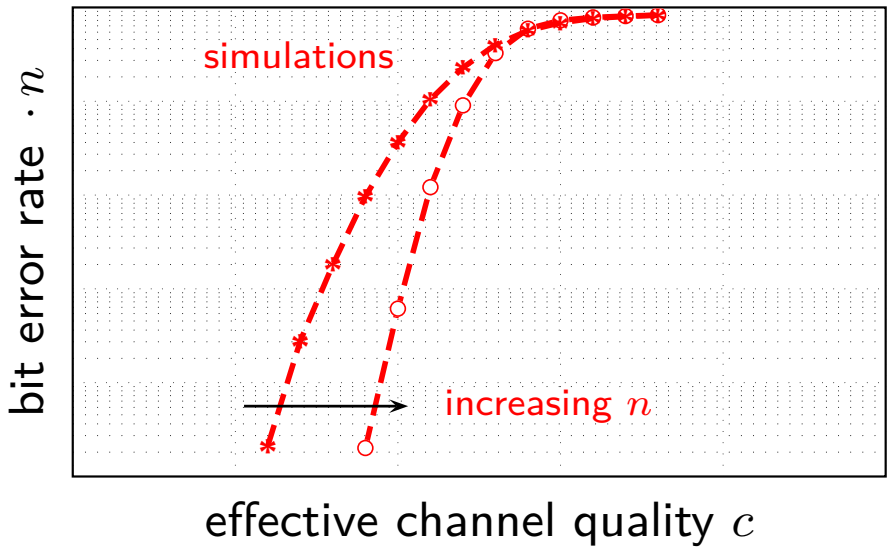
Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



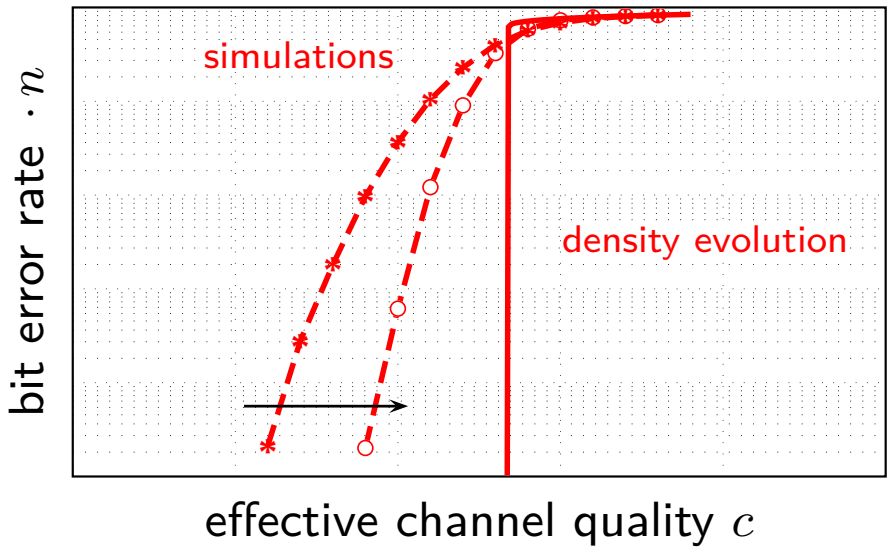
Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



$$\boldsymbol{x}^{(\ell)} = \boldsymbol{\Psi}_{\geq t}(c\boldsymbol{B}\boldsymbol{x}^{(\ell-1)})$$

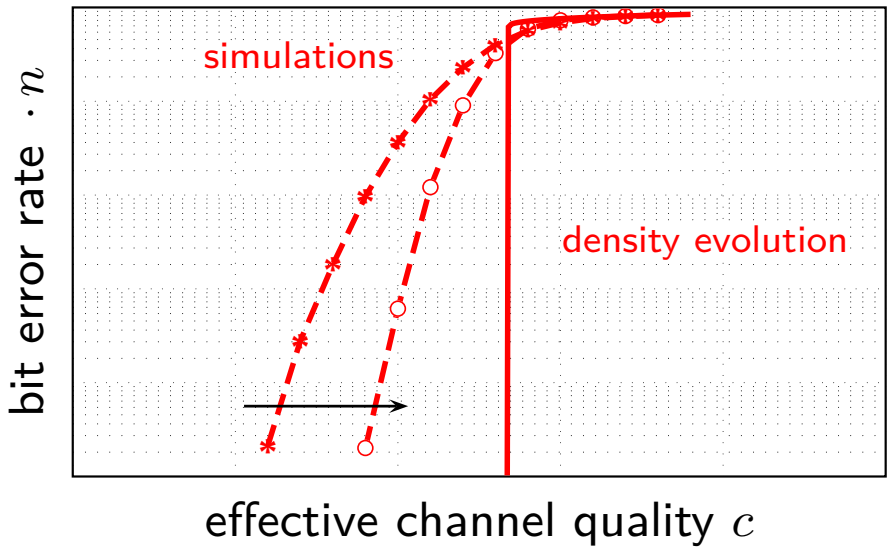
$\boldsymbol{B} \triangleq \gamma\boldsymbol{\eta}$

initial condition $\boldsymbol{x}^{(0)} = (1, \ldots, 1)$

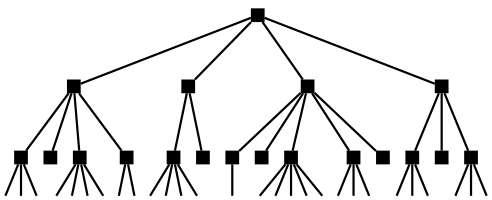
element-wise application of $\Psi_{\geq t}(x) \triangleq 1 - \sum_{i=0}^{t-1} \frac{x^i}{i!} e^{-x}$

Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



Proof and details in [Häger et al., 2017].
Key: convergence results for sparse random graphs [Bollobás et al., 2007]



initial condition $\boldsymbol{x}^{(0)} = (1, \ldots, 1)$

$\boldsymbol{B} \triangleq \gamma\boldsymbol{\eta}$

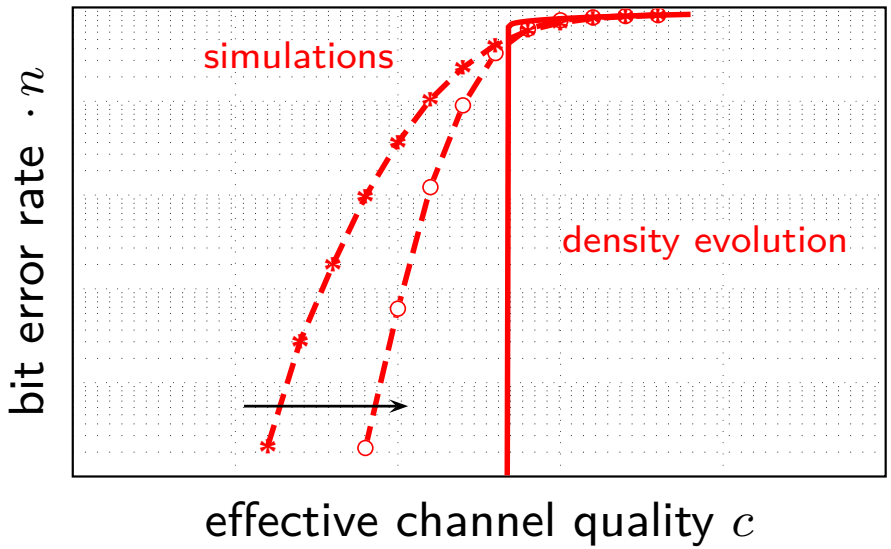
$$\boldsymbol{x}^{(\ell)} = \boldsymbol{\Psi}_{\geq t}(c\boldsymbol{B}\boldsymbol{x}^{(\ell-1)})$$

element-wise application of

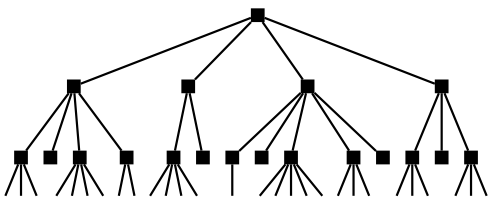
$\Psi_{\geq t}(x) \triangleq 1 - \sum_{i=0}^{t-1} \frac{x^i}{i!} e^{-x}$

Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



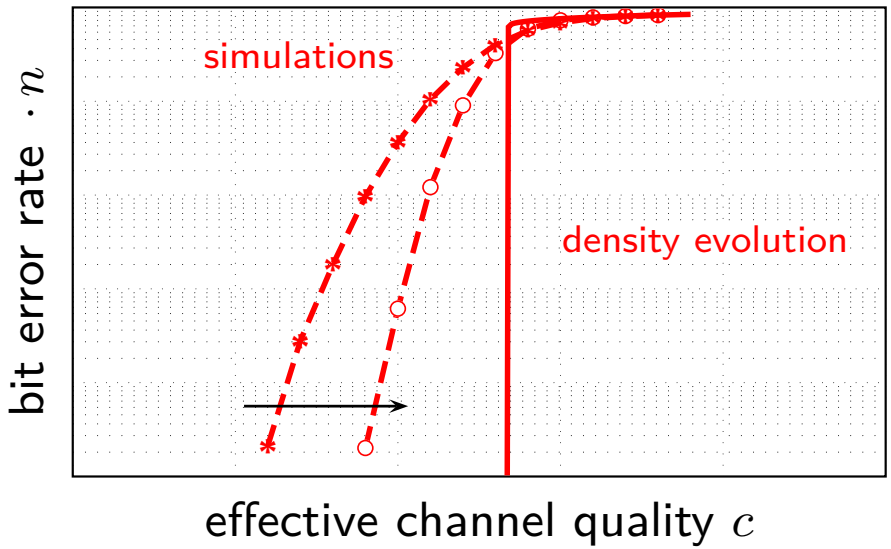
Proof and details in [Häger et al., 2017].
Key: convergence results for sparse random graphs [Bollobás et al., 2007]



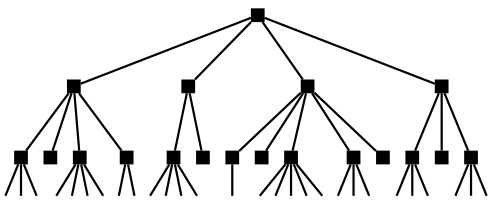
- Generalizes [Schwartz et al., 2005], [Justesen and Høholdt, 2007] to a large class of deterministic codes (staircase, braided, etc.); also works for different decoding schedules (e.g., window decoding)
- Applications: (asymptotic) performance prediction, code comparison via thresholds, efficient parameter optimization, ...

Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



Proof and details in [Häger et al., 2017].
Key: convergence results for sparse random graphs [Bollobás et al., 2007]

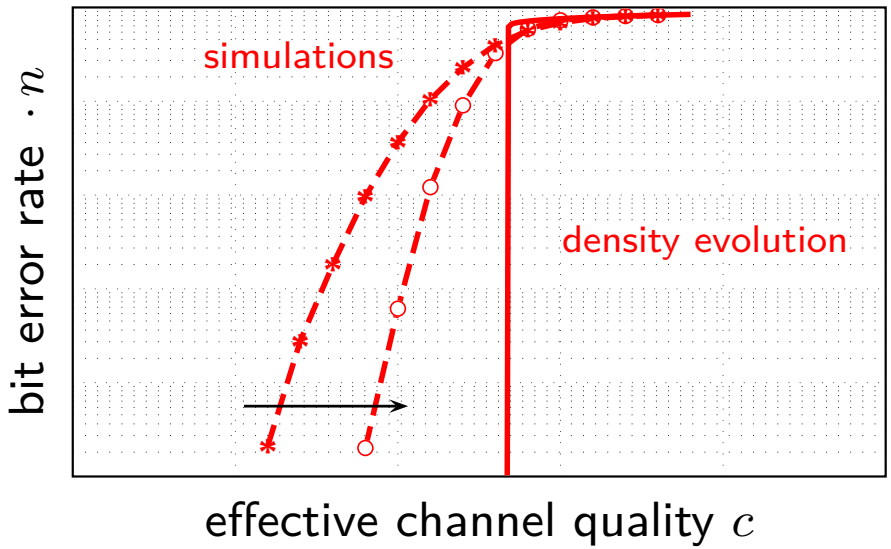


- Generalizes [Schwartz et al., 2005], [Justesen and Høholdt, 2007] to a large class of deterministic codes (staircase, braided, etc.); also works for different decoding schedules (e.g., window decoding)
- Applications: (asymptotic) performance prediction, code comparison via thresholds, efficient parameter optimization, ...

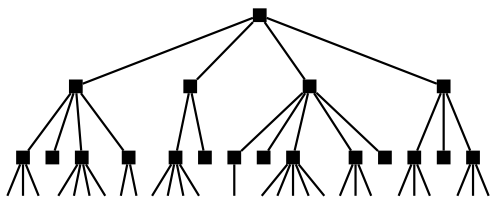
only one small problem ...

Density Evolution

- Let $p = c/n$ for $c > 0$, where c is the effective channel quality



Proof and details in [Häger et al., 2017].
Key: convergence results for sparse random graphs [Bollobás et al., 2007]

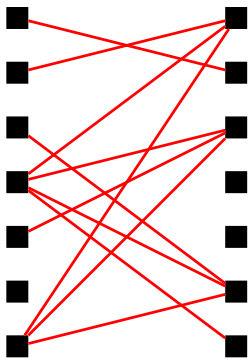


- Generalizes [Schwartz et al., 2005], [Justesen and Høholdt, 2007] to a large class of deterministic codes (staircase, braided, etc.); also works for different decoding schedules (e.g., window decoding)
- Applications: (asymptotic) performance prediction, code comparison via thresholds, efficient parameter optimization, ...

only one small problem ... binary erasure channel is not the target channel

Outline

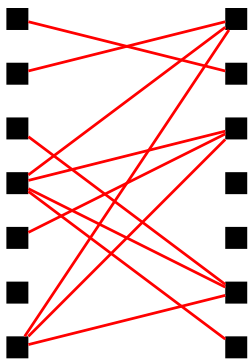
Binary Symmetric Channel vs. Binary Erasure Channel



residual graph

0	?	0	?	0	1	?
?	1	0	1	1	0	1
0	1	0	?	0	?	?
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	?	?	1	1	?
0	1	0	?	0	1	1

Binary Symmetric Channel vs. Binary Erasure Channel



residual graph

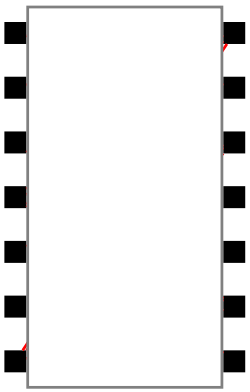
0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

Binary Symmetric Channel vs. Binary Erasure Channel

residual graph

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

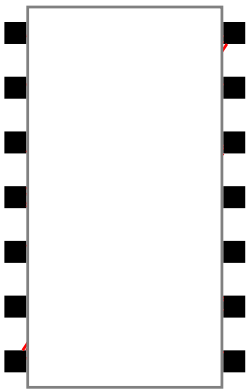
Binary Symmetric Channel vs. Binary Erasure Channel



residual graph

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

Binary Symmetric Channel vs. Binary Erasure Channel

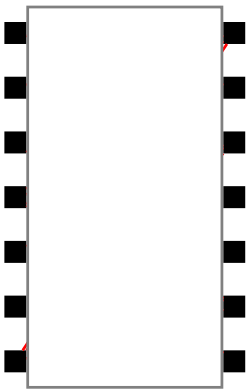


residual graph

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

- Each component code corrects $\leq t$ errors

Binary Symmetric Channel vs. Binary Erasure Channel

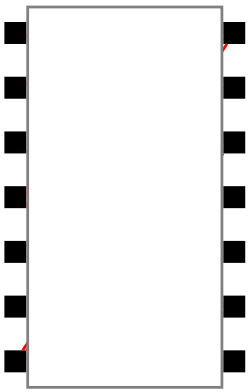


residual graph

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

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections

Binary Symmetric Channel vs. Binary Erasure Channel

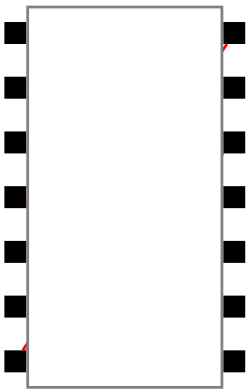


residual graph

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

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections

Binary Symmetric Channel vs. Binary Erasure Channel

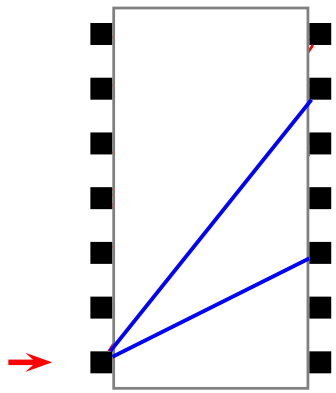


residual graph

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

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections

Binary Symmetric Channel vs. Binary Erasure Channel

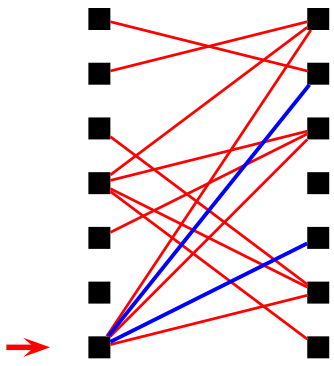


residual graph

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

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections

Binary Symmetric Channel vs. Binary Erasure Channel

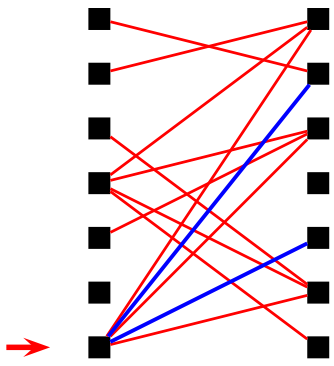


residual graph

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

- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections

Binary Symmetric Channel vs. Binary Erasure Channel



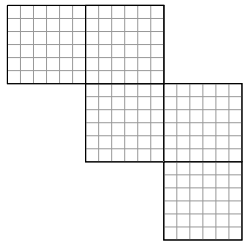
residual graph

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

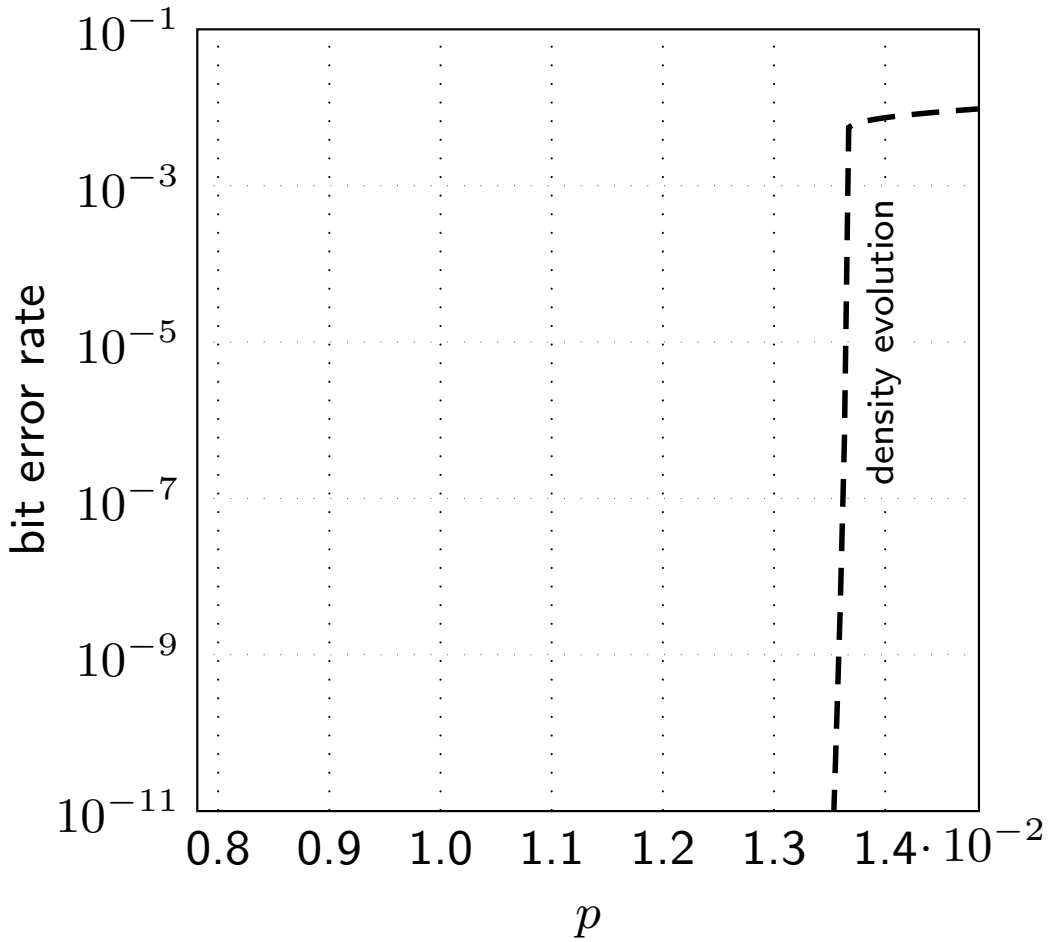
- Each component code corrects $\leq t$ errors
- Undetected errors during component decoding $\Longrightarrow$ miscorrections
- Additional errors during iterative decoding

Performance Loss

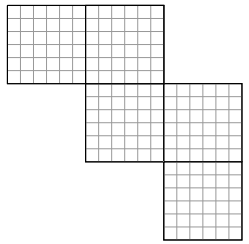
Performance Loss



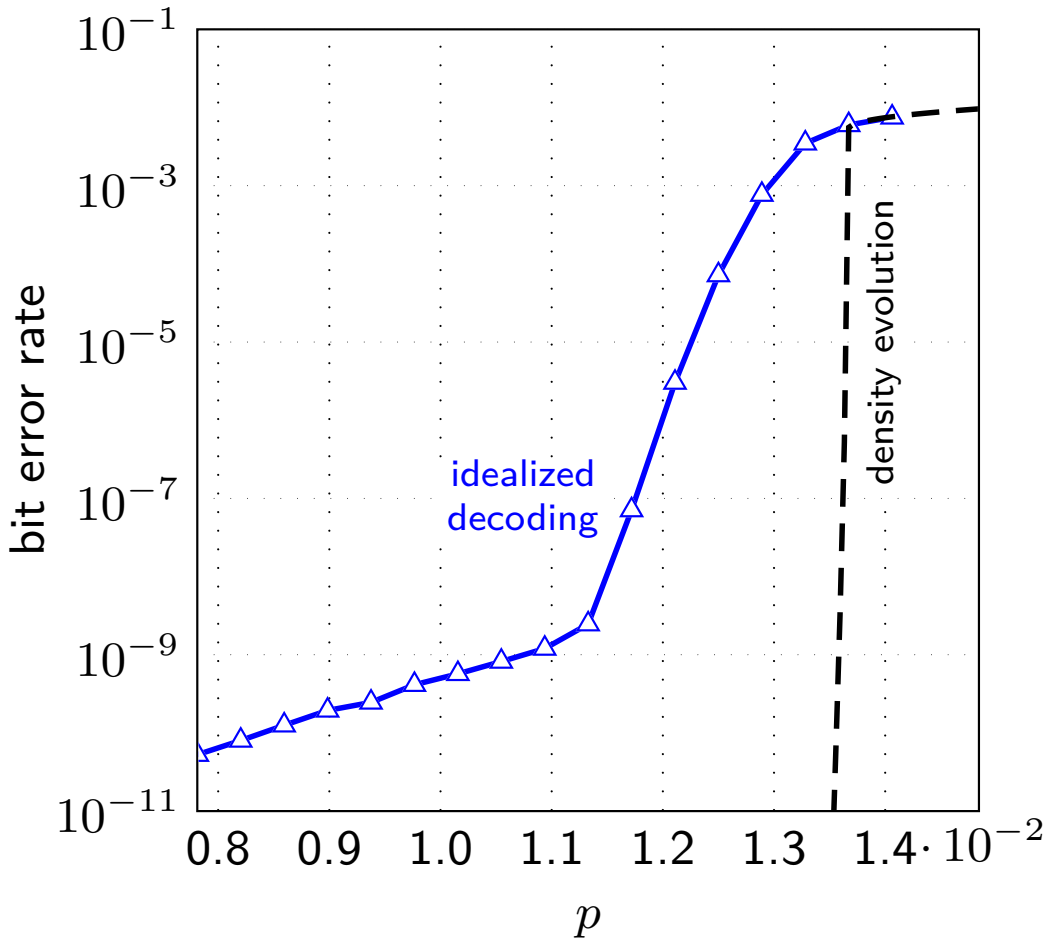
- Staircase code with $n = 256$ and $t = 2$



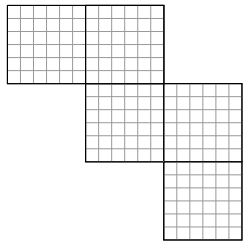
Performance Loss



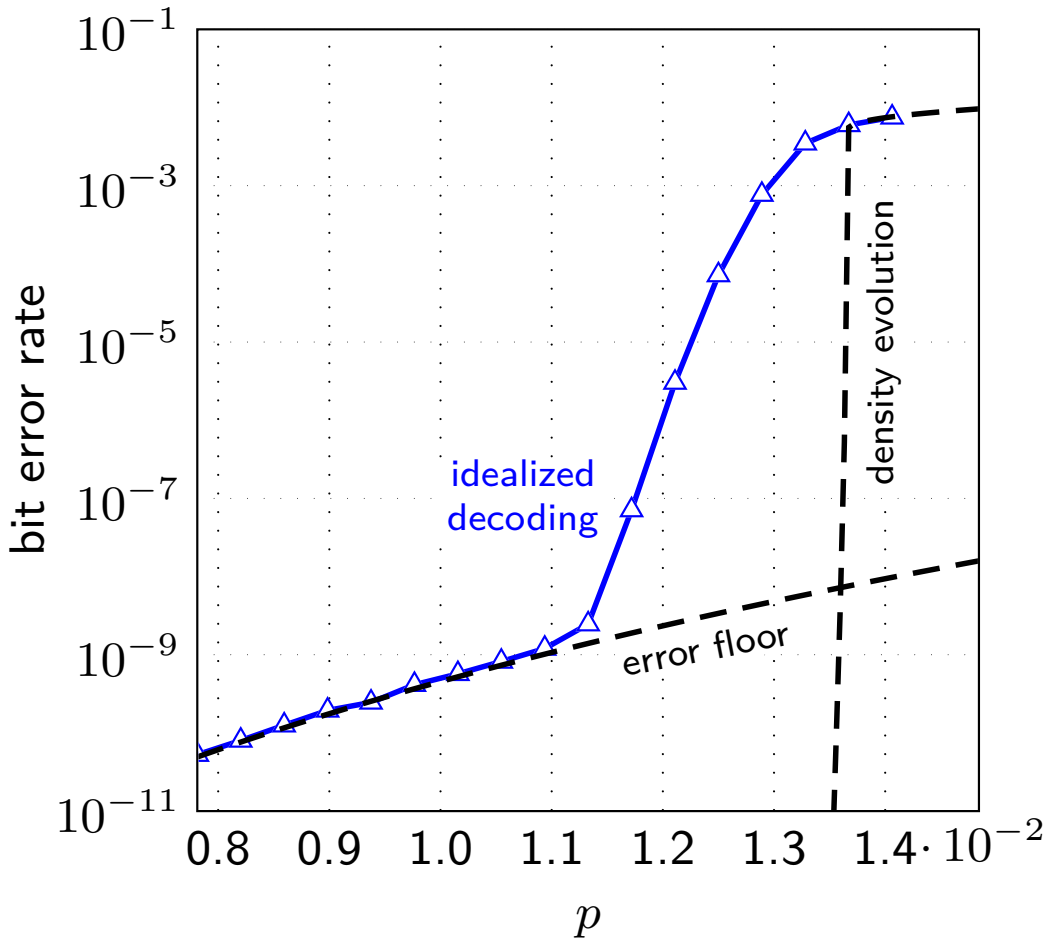
- Staircase code with $n = 256$ and $t = 2$



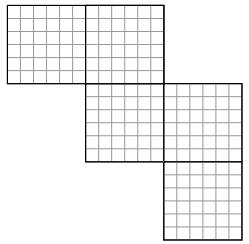
Performance Loss



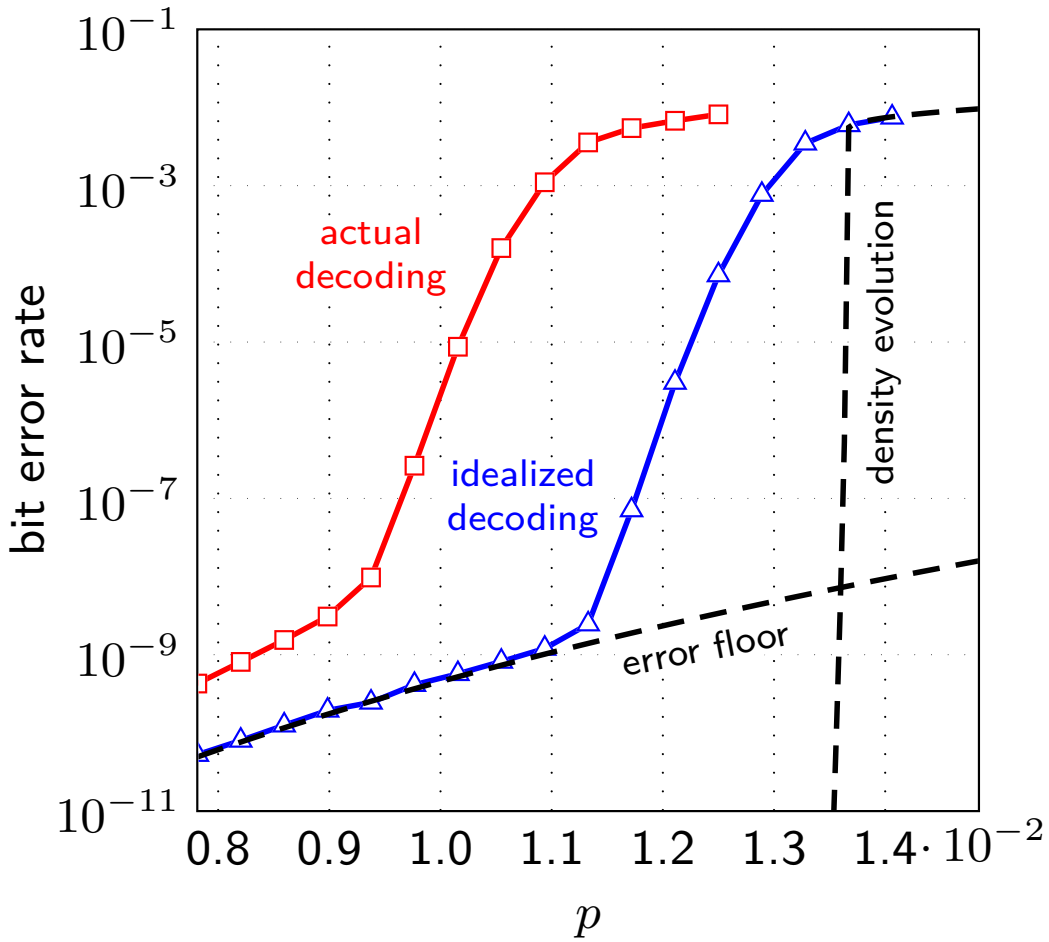
- Staircase code with $n = 256$ and $t = 2$



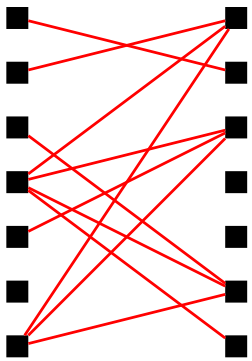
Performance Loss



- Staircase code with $n = 256$ and $t = 2$



Anchor Decoding



residual graph

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

Anchor Decoding

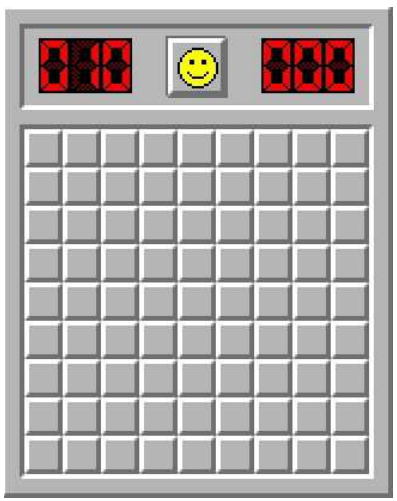
residual graph

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

Anchor Decoding

residual graph

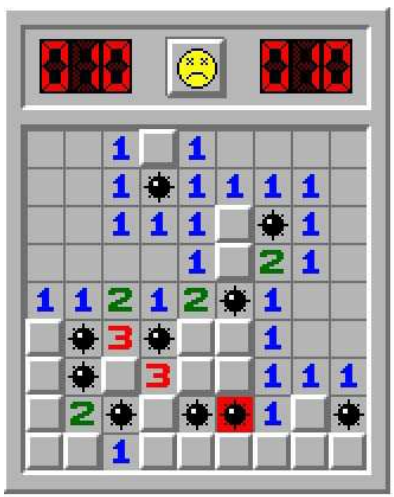
0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1



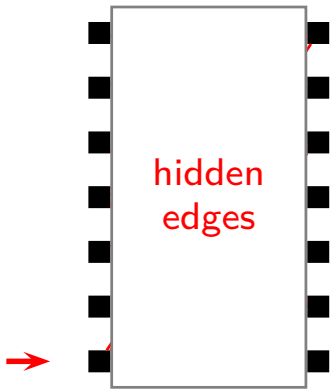
Anchor Decoding

residual graph

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

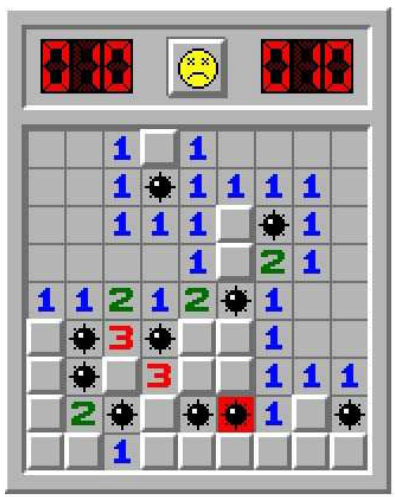


Anchor Decoding

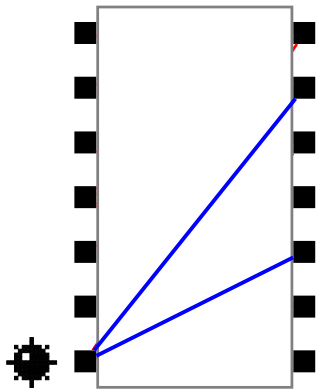


residual graph

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

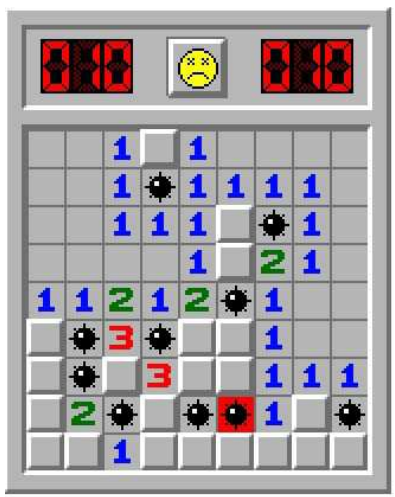


Anchor Decoding

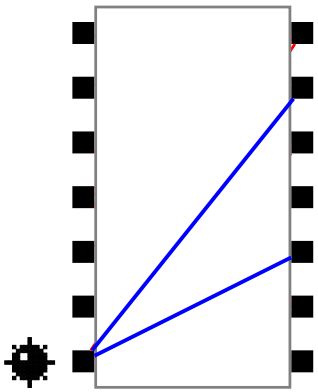


residual graph

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



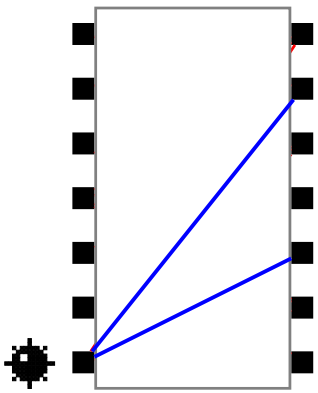
Anchor Decoding



residual graph

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

Anchor Decoding

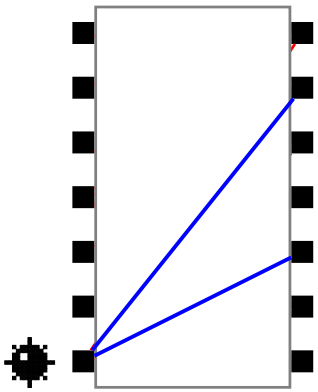


residual graph

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

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit

Anchor Decoding

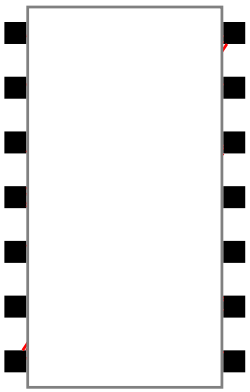


residual graph

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

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor Decoding

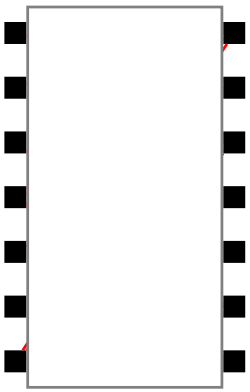


residual graph

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

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor Decoding

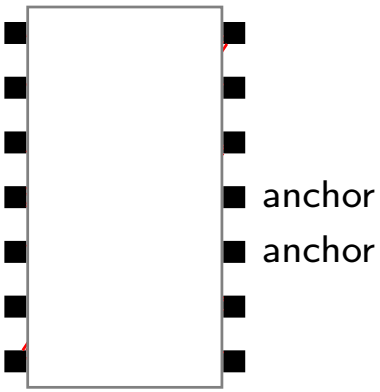


residual graph

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

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

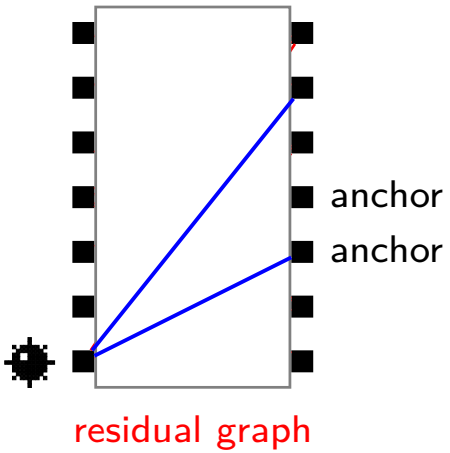
Anchor Decoding



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

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

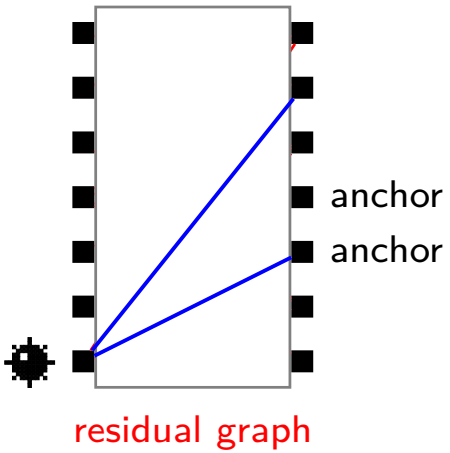
Anchor Decoding



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

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor Decoding

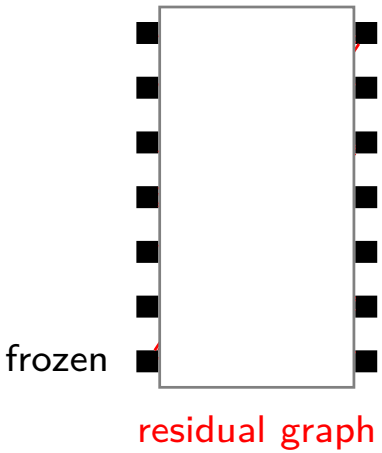


0	0	0	0	0	1	1
1	1	0	1	1	0	0
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	0
1	0	1	1	1	1	0
0	1	0	1	0	1	1

conflict with anchor
$\Longrightarrow$ reject bit flips

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor Decoding

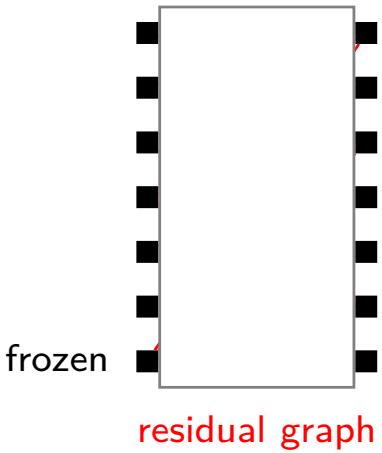


0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

conflict with anchor
$\Longrightarrow$ reject bit flips

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)

Anchor Decoding

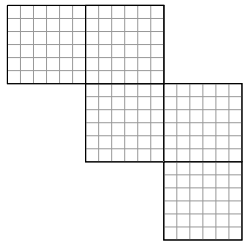


0	0	0	0	0	1	1
1	1	0	1	1	0	1
0	1	0	0	0	0	1
1	1	1	0	1	1	0
0	0	1	0	0	0	1
1	0	1	1	1	1	0
0	1	0	1	0	1	1

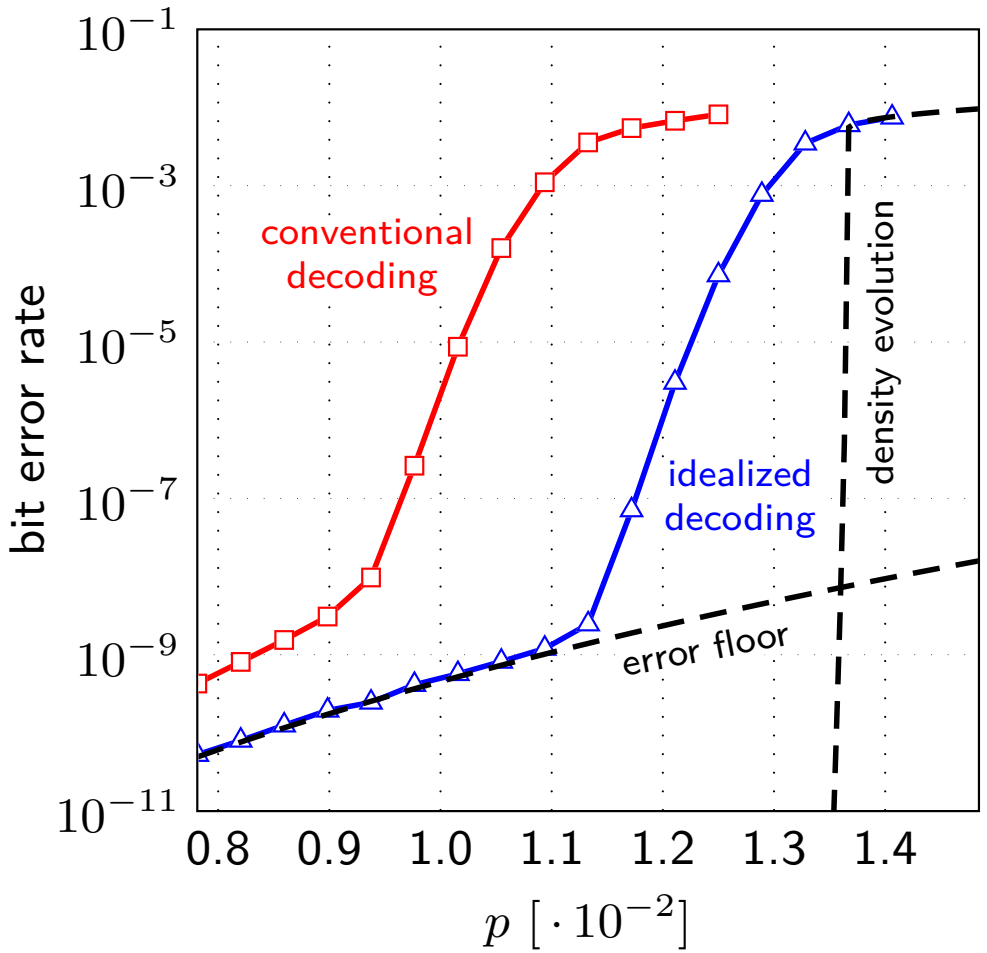
conflict with anchor
$\Longrightarrow$ reject bit flips

- Miscorrections lead to inconsistencies/conflicts: two component codewords may disagree on the value of a bit
- Idea: make correctly decoded codewords anchors and trust their decisions (requires status information for each component codeword)
- If any anchor has too many conflicts, backtrack its bit flips

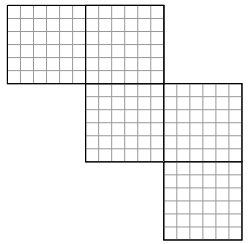
Simulation Results



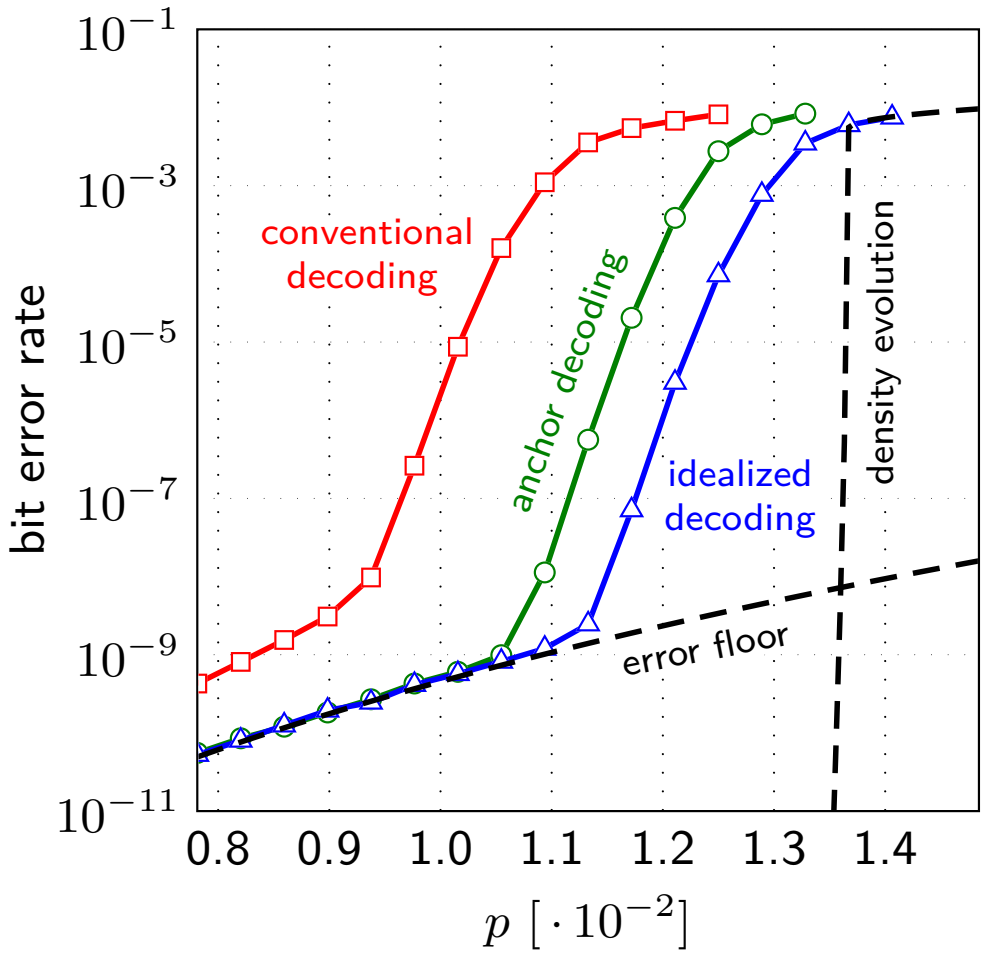
- Staircase code with $n = 256$ and $t = 2$



Simulation Results

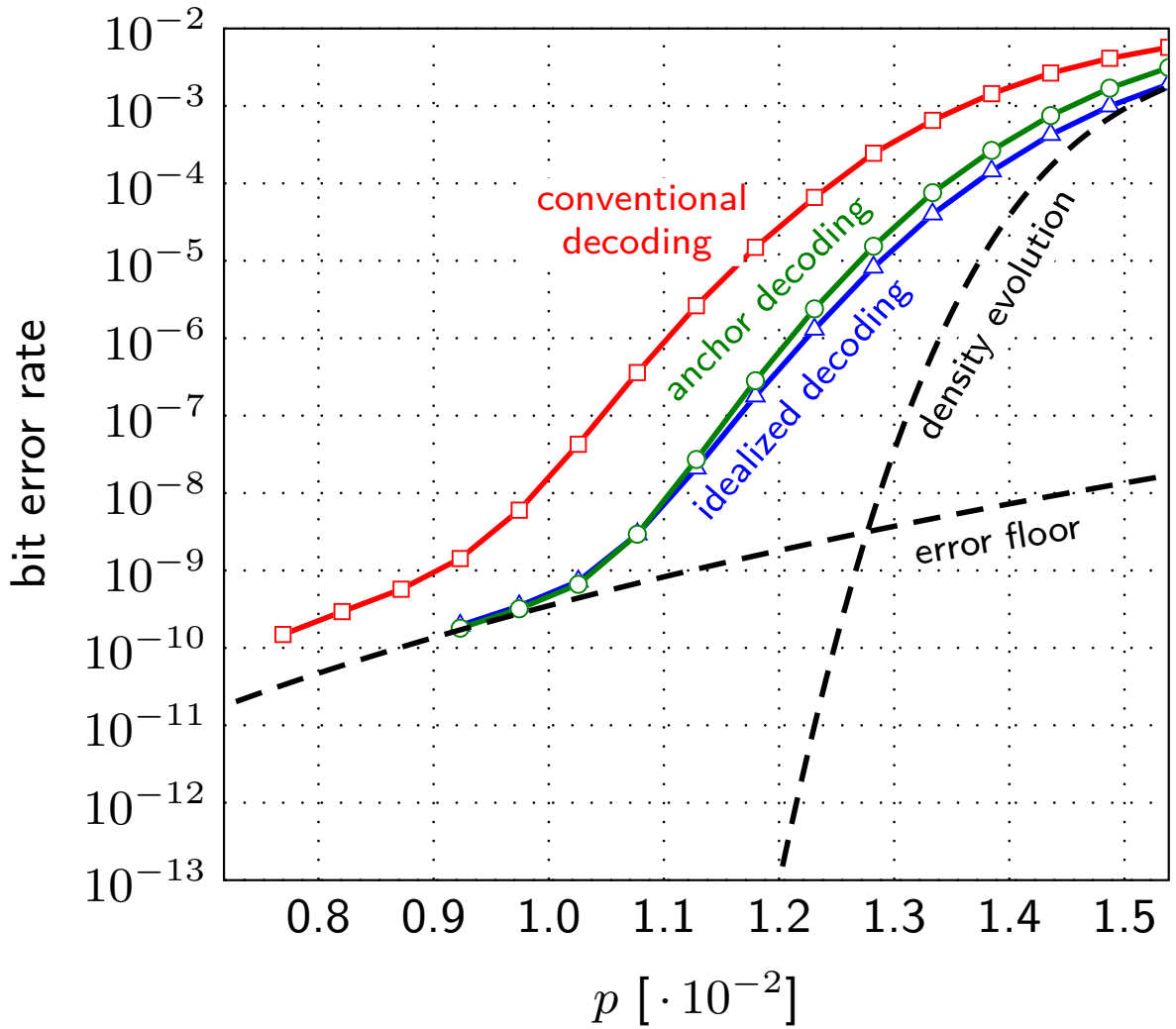


- Staircase code with $n = 256$ and $t = 2$



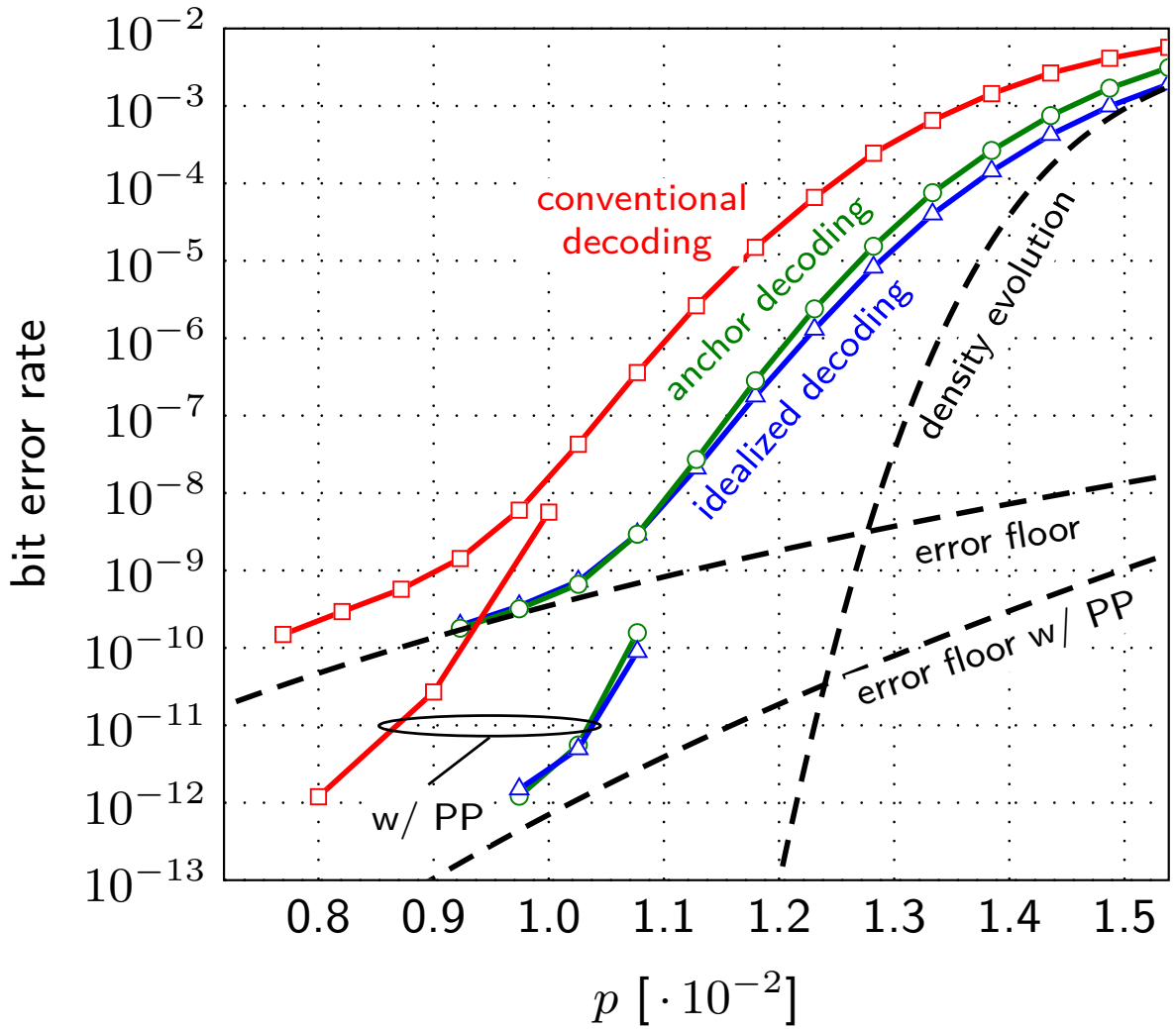
Simulation Results (cont.)

- Product code with $n = 195$ and $t = 2$, see [Condo et al., 2018]



Simulation Results (cont.)

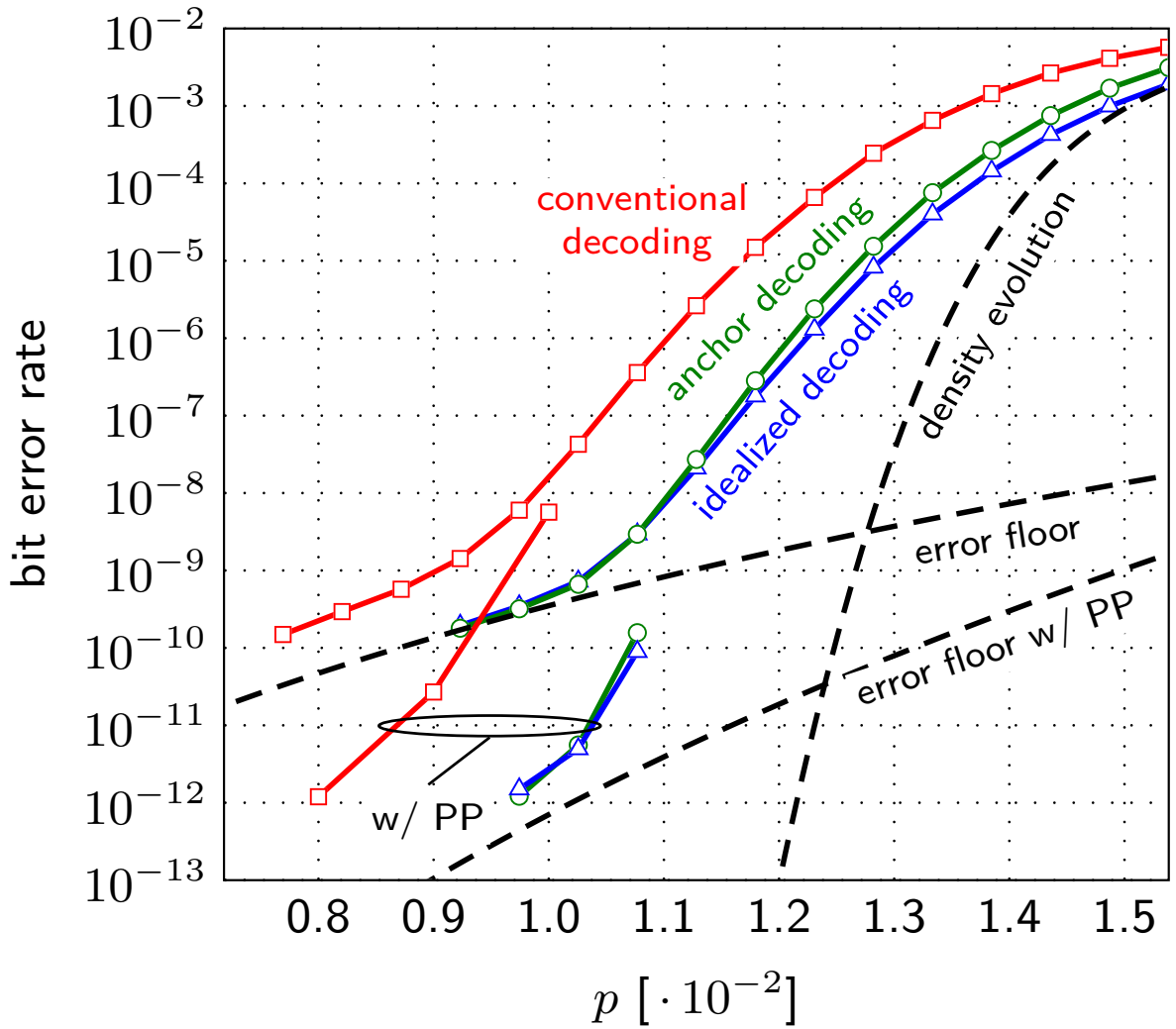
- Product code with $n = 195$ and $t = 2$, see [Condo et al., 2018]



post-processing (PP):
[Jian et al., 2014]
[Mittelholzer et al., 2016]
[Holzbaur et al., 2017]

Simulation Results (cont.)

- Product code with $n = 195$ and $t = 2$, see [Condo et al., 2018]



post-processing (PP):
[Jian et al., 2014]
[Mittelholzer et al., 2016]
[Holzbaur et al., 2017]

Future work: PP for staircase codes, complexity impact on product decoder architecture, ...

Conclusions

- Density evolution can be applied for deterministic generalized product codes over the binary erasure channel.
- In practice, miscorrection-free performance over the binary symmetric channel can be approached with anchor decoding.

Conclusions

- Density evolution can be applied for deterministic generalized product codes over the binary erasure channel.
- In practice, miscorrection-free performance over the binary symmetric channel can be approached with anchor decoding.

Conclusions

- Density evolution can be applied for deterministic generalized product codes over the binary erasure channel.
- In practice, miscorrection-free performance over the binary symmetric channel can be approached with anchor decoding.

Thank you!

FORCE
FIBER-OPTIC COMMUNICATIONS RESEARCH CENTER

Duke UNIVERSITY

References I

Bollobás, B., Janson, S., and Riordan, O. (2007).
The phase transition in inhomogeneous random graphs.
Random Structures and Algorithms, 31(1):3–122.

Condo, C., Giard, P., Leduc-Primeau, F., Sarkis, G., and Gross, W. J. (2018).
A 9.96 dB NCG FEC scheme and 164 bits/cycle low-complexity product decoder architecture.
IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications, 65(4):1420–1431.

Elias, P. (1954).
Error-free coding.
IRE Trans. Inf. Theory, 4(4):29–37.

Häger, C., Pfister, H. D., Graell i Amat, A., and Brännström, F. (2017).
Density evolution for deterministic generalized product codes on the binary erasure channel at high rates.
IEEE Trans. Inf. Theory, 63(7):4357–4378.

Holzbaur, L., Bartz, H., and Wachter-Zeh, A. (2017).
Improved decoding and error floor analysis of staircase codes.
In *Proc. Int. Workshop on Coding and Cryptography (WCC)*, Saint Petersburg, Russia.

Jian, Y.-Y., Pfister, H. D., and Narayanan, K. R. (2012).
Approaching capacity at high rates with iterative hard-decision decoding.
In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Cambridge, MA.

Jian, Y.-Y., Pfister, H. D., Narayanan, K. R., Rao, R., and Mazahreh, R. (2014).
Iterative hard-decision decoding of braided BCH codes for high-speed optical communication.
In *Proc. IEEE Glob. Communication Conf. (GLOBECOM)*, Atlanta, GA.

References II

Justesen, J. and Høholdt, T. (2007).
Analysis of iterated hard decision decoding of product codes with Reed-Solomon component codes.
In *Proc. IEEE Information Theory Workshop (ITW)*, Tahoe City, CA.

Luby, M. G., Mitzenmacher, M., and Shokrollahi, M. A. (1998).
Analysis of random processes via and-or tree evaluation.
In *Proc. 9th Annual ACM-SIAM Symp. Discrete Algorithms*, pages 364–373, San Franscisco, CA.

Mittelholzer, T., Parnell, T., Papandreou, N., and Pozidis, H. (2016).
Improving the error-floor performance of binary half-product codes.
In *Proc. Int. Symp. Information Theory and its Applications (ISITA)*, Montenery, CA.

Richardson, T. J. and Urbanke, R. L. (2001).
The capacity of low-density parity-check codes under message-passing decoding.
IEEE Trans. Inf. Theory, 47(2):599–618.

Schwartz, M., Siegel, P., and Vardy, A. (2005).
On the asymptotic performance of iterative decoders for product codes.
In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Adelaide, SA.

Smith, B. P., Farhood, A., Hunt, A., Kschischang, F. R., and Lodge, J. (2012).
Staircase codes: FEC for 100 Gb/s OTN.
J. Lightw. Technol., 30(1):110–117.

Zhang, L. M., Truhachev, D., and Kschischang, F. R. (2015).
Spatially-coupled split-component codes with bounded-distance component decoding.
In *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Hong Kong.