On Parameter Optimization for Staircase Codes

Christian Häger[1] Alexandre Graell i Amat[1] Henry D. Pfister[2]
Alex Alvarado[3] Fredrik Brännström[1] Erik Agrell[1]

[1]Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden

[2]Department of Electrical Engineering, Duke University, Durham, North Carolina

[3]Department of Electronic & Electrical Engineering, University College London, UK

{christian.haeger, alexandre.graell, fredrik.brannstrom, agrell}@chalmers.se
henry.pfister@duke.edu, alex.alvarado@ieee.org

Optical Fiber Communications Conference and Exhibition (OFC)
Los Angeles, March 26, 2015

FORCE
FIBER-OPTIC COMMUNICATIONS
RESEARCH CENTER

CHALMERS

Outline

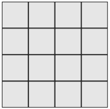
Staircase Codes (and Product Codes)

Staircase Codes (and Product Codes)

- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954]

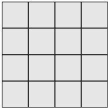


Example: $n = 4$

- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954]

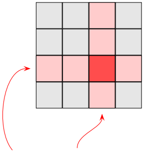


each row/column is a codeword in $\mathcal{C}$
($2n$ code constraints in total)

- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954]



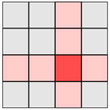
Example: $n = 4$

each row/column is a codeword in $\mathcal{C}$
($2n$ code constraints in total)

- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

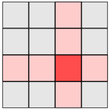
rectangular array [Elias, 1954]



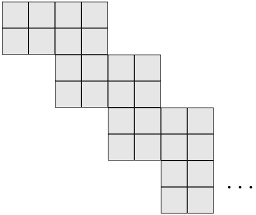
- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954]



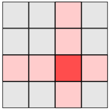
staircase array [Smith et al., JLT, 2012]

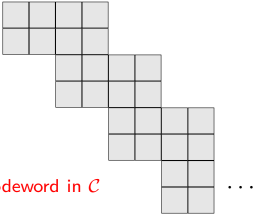


- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954] staircase array [Smith et al., JLT, 2012]

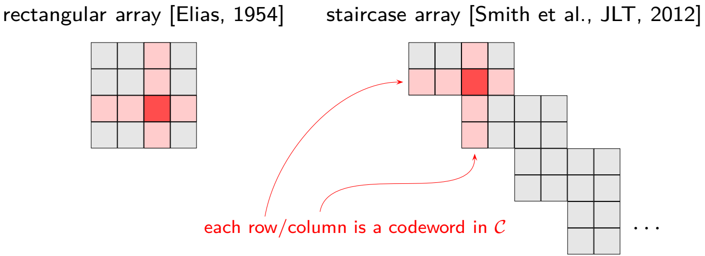




each row/column is a codeword in $\mathcal{C}$

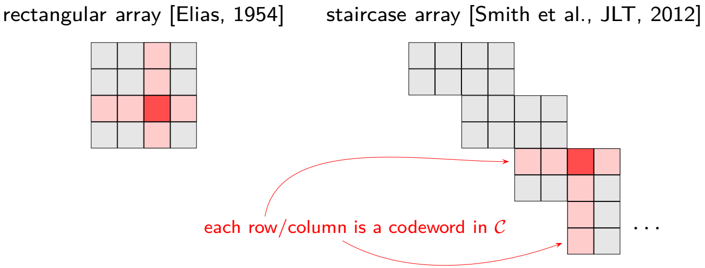
- Start with a binary linear code $\mathcal{C}(n, k, d_{\mathsf{min}})$ as a "building block"

Staircase Codes (and Product Codes)



- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

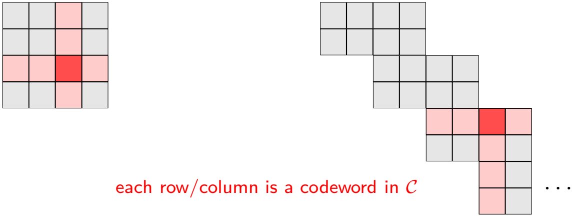
Staircase Codes (and Product Codes)



- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

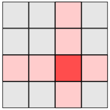
rectangular array [Elias, 1954] staircase array [Smith et al., JLT, 2012]



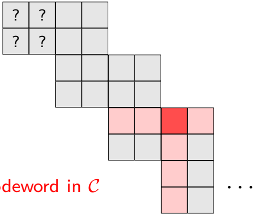
- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954]



staircase array [Smith et al., JLT, 2012]

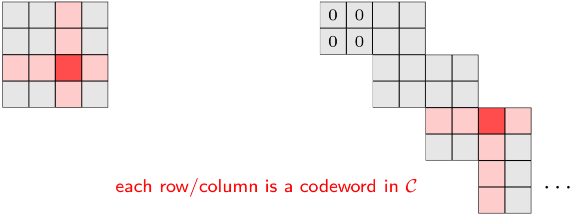


each row/column is a codeword in $\mathcal{C}$

- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

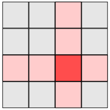
rectangular array [Elias, 1954] staircase array [Smith et al., JLT, 2012]



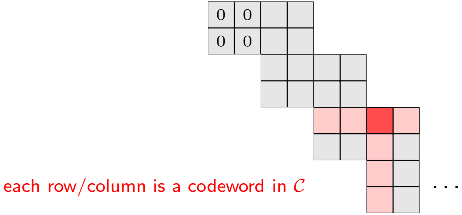
- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954]



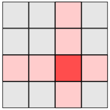
staircase array [Smith et al., JLT, 2012]

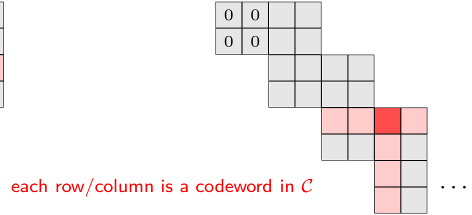


- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"
- $\mathcal{C}$: BCH code defined by (ν, t, s), where
  - ν: Galois-field extension degree
  - t: error-correction capability
  - s: shortening parameter

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954] staircase array [Smith et al., JLT, 2012]

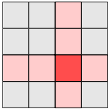




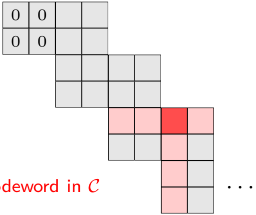
- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"
- $\mathcal{C}$: BCH code defined by (ν, t, s), where
  - ν: Galois-field extension degree
  - t: error-correction capability
  - s: shortening parameter
- $\Rightarrow$ length $n = 2^\nu - 1 - s$, dimension $k = 2^\nu - \nu t - 1 - s$

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954]



staircase array [Smith et al., JLT, 2012]

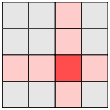


each row/column is a codeword in $\mathcal{C}$

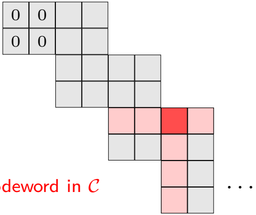
- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"
- $\mathcal{C}$: BCH code defined by (ν, t, s), where
  - ν: Galois-field extension degree
  - t: error-correction capability
  - s: shortening parameter
- $\Rightarrow$ length $n = 2^\nu - 1 - s$, dimension $k = 2^\nu - \nu t - 1 - s$
- Staircase code rate $R = 2k/n - 1$ and FEC overhead OH $= 1/R - 1$

Staircase Codes (and Product Codes)

rectangular array [Elias, 1954]



staircase array [Smith et al., JLT, 2012]



each row/column is a codeword in $\mathcal{C}$

- Start with a binary linear code $\mathcal{C}(n, k, d_{\text{min}})$ as a "building block"
- $\mathcal{C}$: BCH code defined by (ν, t, s), where
  - ν: Galois-field extension degree
  - t: error-correction capability
  - s: shortening parameter
- $\Rightarrow$ length $n = 2^\nu - 1 - s$, dimension $k = 2^\nu - \nu t - 1 - s$
- Staircase code rate $R = 2k/n - 1$ and FEC overhead $\text{OH} = 1/R - 1$

Problem Formulation

For fixed OH, find a "good" triple (ν, t, s).

Decoding Algorithm and Previous Work

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window
- Iterative intrinsic message-passing (IMP) with "hard" (binary) messages

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window
- Iterative intrinsic message-passing (IMP) with "hard" (binary) messages
- Significant decoder data flow reduction compared to LDPC codes [Smith et al., JLT, 2012] $\rightarrow$ high-speed optical communications

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window
- Iterative intrinsic message-passing (IMP) with “hard” (binary) messages
- Significant decoder data flow reduction compared to LDPC codes [Smith et al., JLT, 2012] $\rightarrow$ high-speed optical communications

Previous work [Zhang and Kschischang, JLT, 2014]

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window
- Iterative intrinsic message-passing (IMP) with "hard" (binary) messages
- Significant decoder data flow reduction compared to LDPC codes [Smith et al., JLT, 2012] $\rightarrow$ high-speed optical communications

Previous work [Zhang and Kschischang, JLT, 2014]

- Parameter space based on practical consideration: product set of OH $\in \{1/i : i = 3, 4, \ldots, 16\}$, $\nu \in \{8, 9, 10, 11, 12\}$, $t \in \{2, 3, 4, 5, 6\}$.

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window
- Iterative intrinsic message-passing (IMP) with "hard" (binary) messages
- Significant decoder data flow reduction compared to LDPC codes [Smith et al., JLT, 2012] $\rightarrow$ high-speed optical communications

Previous work [Zhang and Kschischang, JLT, 2014]

- Parameter space based on practical consideration: product set of OH $\in \{1/i : i = 3, 4, \ldots, 16\}$, $\nu \in \{8, 9, 10, 11, 12\}$, $t \in \{2, 3, 4, 5, 6\}$.
- Software simulations to predict staircase code performance

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window
- Iterative intrinsic message-passing (IMP) with "hard" (binary) messages
- Significant decoder data flow reduction compared to LDPC codes [Smith et al., JLT, 2012] $\rightarrow$ high-speed optical communications

Previous work [Zhang and Kschischang, JLT, 2014]

- Parameter space based on practical consideration: product set of OH $\in \{1/i : i = 3, 4, \ldots, 16\}$, $\nu \in \{8, 9, 10, 11, 12\}$, $t \in \{2, 3, 4, 5, 6\}$.
- Software simulations to predict staircase code performance
- Computationally intensive: use simplified BCH decoders, which do not account for miscorrections

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window
- Iterative intrinsic message-passing (IMP) with "hard" (binary) messages
- Significant decoder data flow reduction compared to LDPC codes [Smith et al., JLT, 2012] $\rightarrow$ high-speed optical communications

Previous work [Zhang and Kschischang, JLT, 2014]

- Parameter space based on practical consideration: product set of OH $\in \{1/i : i = 3, 4, \ldots, 16\}$, $\nu \in \{8, 9, 10, 11, 12\}$, $t \in \{2, 3, 4, 5, 6\}$.
- Software simulations to predict staircase code performance
- Computationally intensive: use simplified BCH decoders, which do not account for miscorrections

Our Approach

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window
- Iterative intrinsic message-passing (IMP) with "hard" (binary) messages
- Significant decoder data flow reduction compared to LDPC codes [Smith et al., JLT, 2012] $\rightarrow$ high-speed optical communications

Previous work [Zhang and Kschischang, JLT, 2014]

- Parameter space based on practical consideration: product set of OH $\in \{1/i : i = 3, 4, \dots, 16\}$, $\nu \in \{8, 9, 10, 11, 12\}$, $t \in \{2, 3, 4, 5, 6\}$.
- Software simulations to predict staircase code performance
- Computationally intensive: use simplified BCH decoders, which do not account for miscorrections

Our Approach

- Connect staircase codes to spatially-coupled generalized LDPC (SC-GLDPC) code ensemble in [Jian et al., ISIT, 2012]

Decoding Algorithm and Previous Work

- Iterate between BCH decoders for all rows/columns in a sliding window
- Iterative intrinsic message-passing (IMP) with “hard” (binary) messages
- Significant decoder data flow reduction compared to LDPC codes [Smith et al., JLT, 2012] $\rightarrow$ high-speed optical communications

Previous work [Zhang and Kschischang, JLT, 2014]

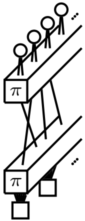
- Parameter space based on practical consideration: product set of OH $\in \{1/i : i = 3, 4, \dots, 16\}$, $\nu \in \{8, 9, 10, 11, 12\}$, $t \in \{2, 3, 4, 5, 6\}$.
- Software simulations to predict staircase code performance
- Computationally intensive: use simplified BCH decoders, which do not account for miscorrections

Our Approach

- Connect staircase codes to spatially-coupled generalized LDPC (SC-GLDPC) code ensemble in [Jian et al., ISIT, 2012]
- Use density evolution and ensemble thresholds to optimize parameters, can account for miscorrections assuming extrinsic message passing (EMP) [Jian et al., ISIT, 2012]

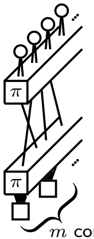
Spatially-Coupled Generalized LDPC Codes

Spatially-Coupled Generalized LDPC Codes



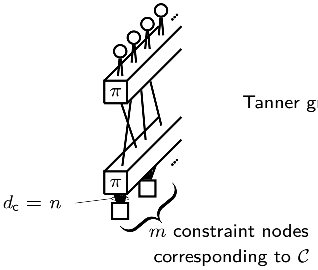
Tanner graph

Spatially-Coupled Generalized LDPC Codes



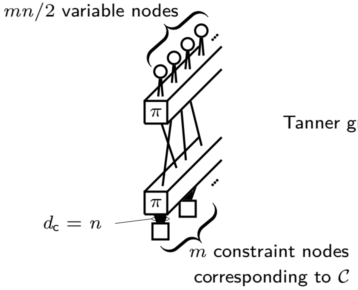
Tanner graph

Spatially-Coupled Generalized LDPC Codes



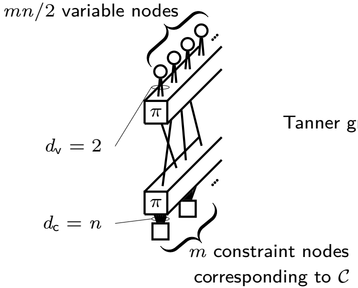
Tanner graph

Spatially-Coupled Generalized LDPC Codes



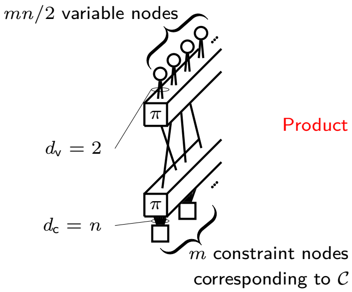
Tanner graph

Spatially-Coupled Generalized LDPC Codes



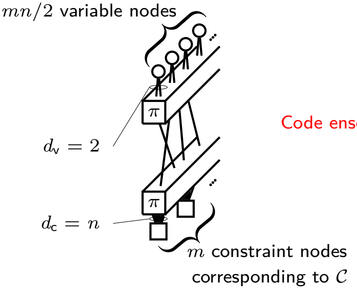
Tanner graph

Spatially-Coupled Generalized LDPC Codes



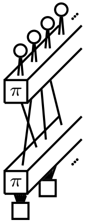
Product code: $m = 2n$, "structured" permutations π

Spatially-Coupled Generalized LDPC Codes

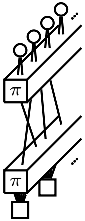


Code ensemble: set of codes defined by all possible π

Spatially-Coupled Generalized LDPC Codes

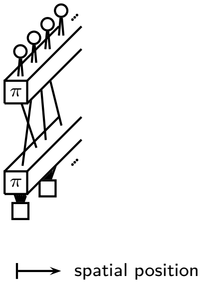


Spatially-Coupled Generalized LDPC Codes



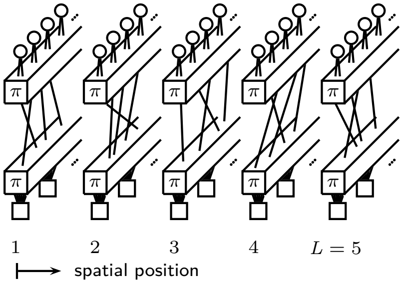
- SC-GLDPC code ensemble $(\mathcal{C}, m, L, w)$ [Jian et al., ISIT, 2012]
  - m: number of constraint nodes per spatial position
  - L: total number of spatial positions
  - w: coupling width

Spatially-Coupled Generalized LDPC Codes



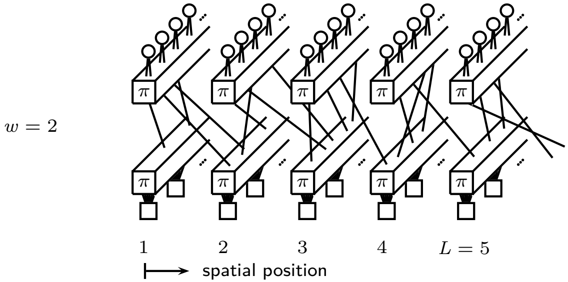
- SC-GLDPC code ensemble $(\mathcal{C}, m, L, w)$ [Jian et al., ISIT, 2012]
  - m: number of constraint nodes per spatial position
  - L: total number of spatial positions
  - w: coupling width

Spatially-Coupled Generalized LDPC Codes



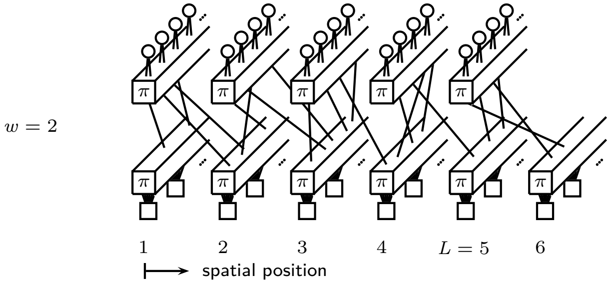
- SC-GLDPC code ensemble $(\mathcal{C}, m, L, w)$ [Jian et al., ISIT, 2012]
    - m: number of constraint nodes per spatial position
    - L: total number of spatial positions
    - w: coupling width

Spatially-Coupled Generalized LDPC Codes



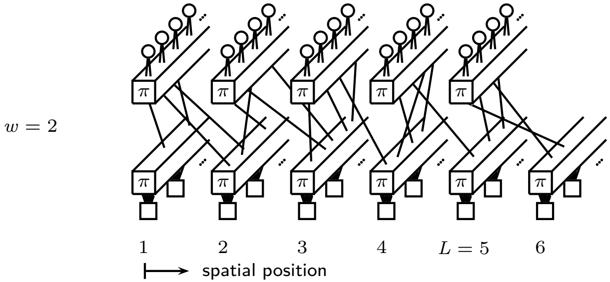
- SC-GLDPC code ensemble $(\mathcal{C}, m, L, w)$ [Jian et al., ISIT, 2012]
  - m: number of constraint nodes per spatial position
  - L: total number of spatial positions
  - w: coupling width

Spatially-Coupled Generalized LDPC Codes



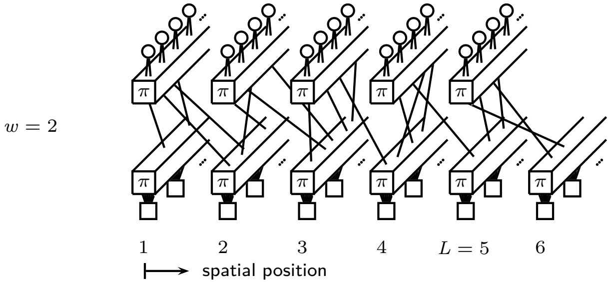
- SC-GLDPC code ensemble $(\mathcal{C}, m, L, w)$ [Jian et al., ISIT, 2012]
  - m: number of constraint nodes per spatial position
  - L: total number of spatial positions
  - w: coupling width

Spatially-Coupled Generalized LDPC Codes



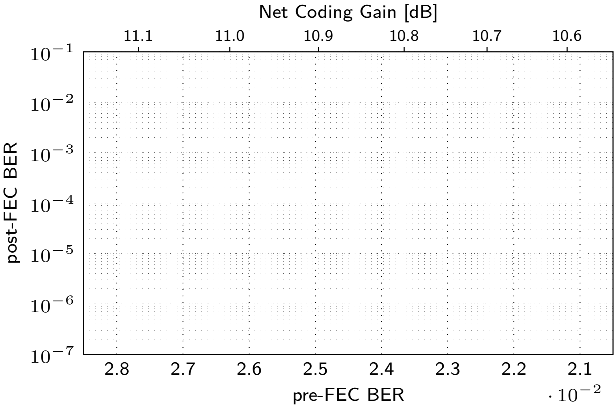
- SC-GLDPC code ensemble $(\mathcal{C}, m, L, w)$ [Jian et al., ISIT, 2012]
  - m: number of constraint nodes per spatial position
  - L: total number of spatial positions
  - w: coupling width
- Key observation: staircase code contained in the ensemble for $m = n/2$, $L \to \infty$ and $w = 2$ assuming "structured" permutations π

Spatially-Coupled Generalized LDPC Codes

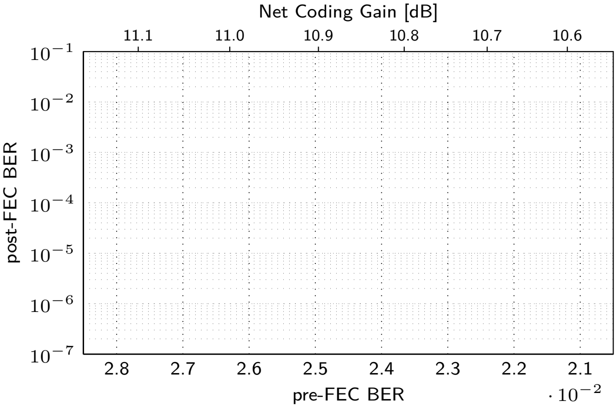


- SC-GLDPC code ensemble $(\mathcal{C}, m, L, w)$ [Jian et al., ISIT, 2012]
  - m: number of constraint nodes per spatial position
  - L: total number of spatial positions
  - w: coupling width
- Key observation: staircase code contained in the ensemble for $m = n/2$, $L \to \infty$ and $w = 2$ assuming "structured" permutations π
- Asymptotic $(m \to \infty)$ ensemble behavior can be analyzed via density evolution (DE) assuming extrinsic message passing (EMP)

Example (OH $= 33.33\%$): Density Evolution and Thresholds

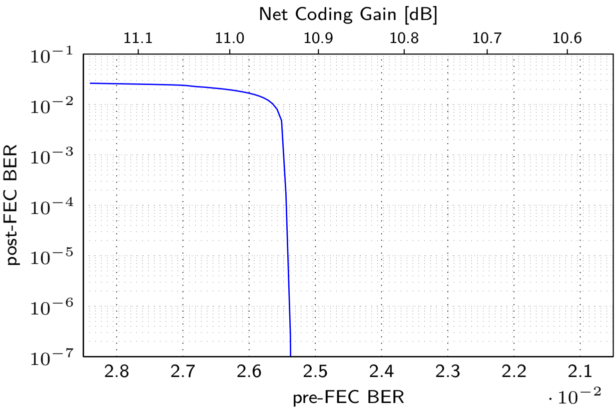


Example (OH $= 33.33\%$): Density Evolution and Thresholds



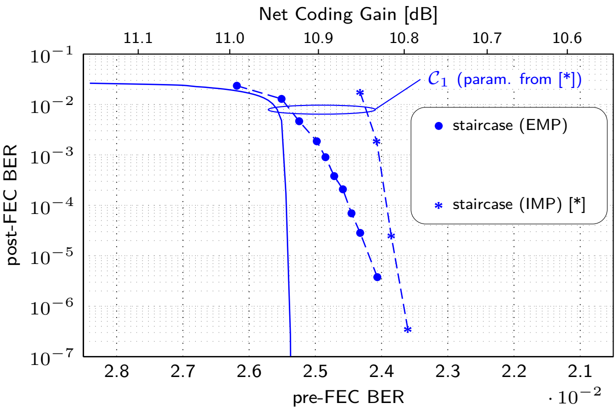
- $\mathcal{C}_1$ with $(\nu, t, s) = (9, 5, 151)$ *[Zhang and Kschischang, JLT, 2014]

Example (OH $= 33.33\%$): Density Evolution and Thresholds



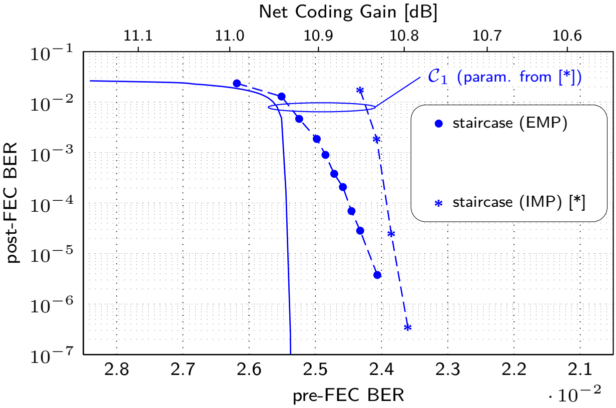
- $\mathcal{C}_1$ with $(\nu, t, s) = (9, 5, 151)$ *[Zhang and Kschischang, JLT, 2014]
- DE for $(\mathcal{C}_1, \infty, 30, 2)$ SC-GLDPC, adapted to sliding-window decoding

Example (OH $= 33.33\%$): Density Evolution and Thresholds



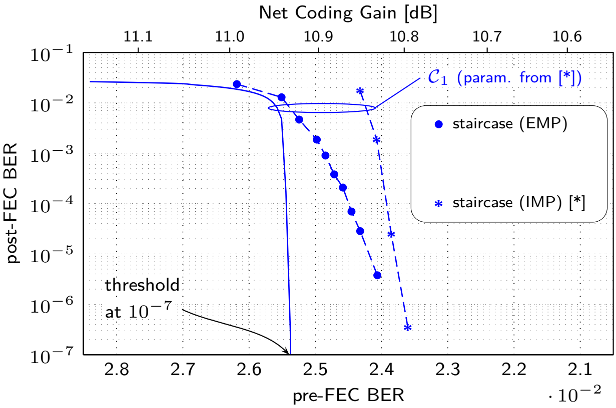
- $\mathcal{C}_1$ with $(\nu, t, s) = (9, 5, 151)$ *[Zhang and Kschischang, JLT, 2014]
- DE for $(\mathcal{C}_1, \infty, 30, 2)$ SC-GLDPC, adapted to sliding-window decoding

Example (OH $= 33.33\%$): Density Evolution and Thresholds



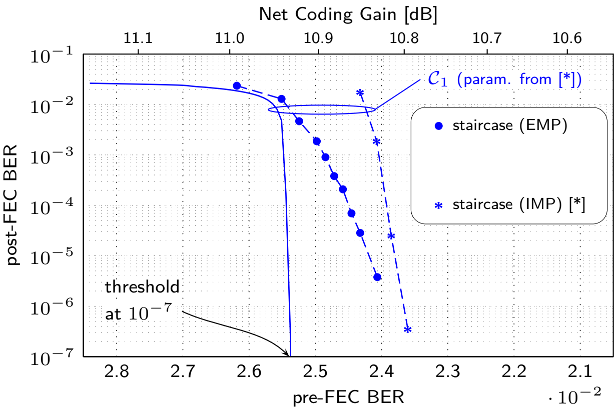
- $\mathcal{C}_1$ with $(\nu, t, s) = (9, 5, 151)$ *[Zhang and Kschischang, JLT, 2014]
- DE for $(\mathcal{C}_1, \infty, 30, 2)$ SC-GLDPC, adapted to sliding-window decoding
- DE accurately predicts pre-FEC BER region where staircase performance curve "bends" into waterfall behavior

Example (OH $= 33.33\%$): Density Evolution and Thresholds

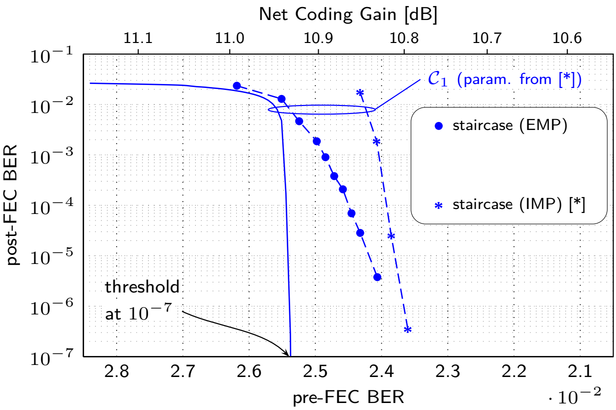


- $\mathcal{C}_1$ with $(\nu, t, s) = (9, 5, 151)$ *[Zhang and Kschischang, JLT, 2014]
- DE for $(\mathcal{C}_1, \infty, 30, 2)$ SC-GLDPC, adapted to sliding-window decoding
- DE accurately predicts pre-FEC BER region where staircase performance curve "bends" into waterfall behavior
- Use decoding thresholds for parameter optimization

Example (OH $= 33.33\%$): Density Evolution and Thresholds

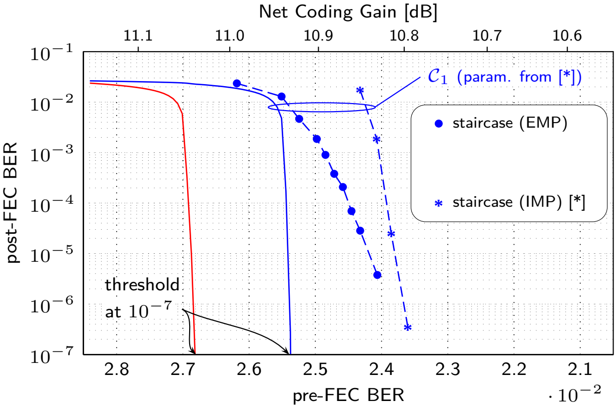


Example (OH $= 33.33\%$): Density Evolution and Thresholds



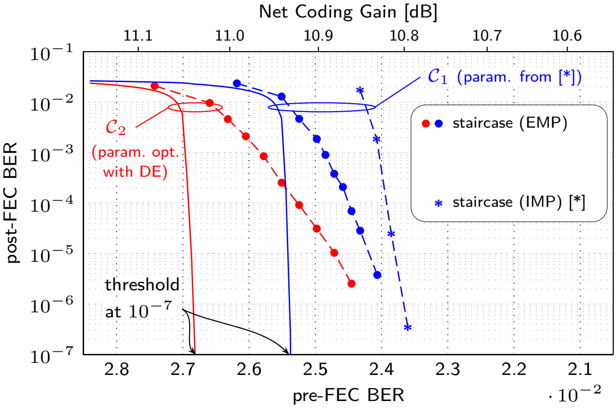
- Same parameter space as *[Zhang and Kschischang, JLT, 2014] → full table for all OHs in paper

Example (OH $= 33.33\%$): Density Evolution and Thresholds



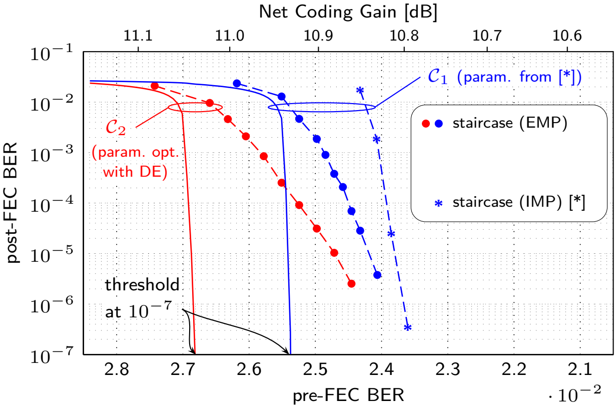
- Same parameter space as *[Zhang and Kschischang, JLT, 2014] → full table for all OHs in paper
- Result for OH $= 33.33\%$: $\mathcal{C}_2$ defined by $(\nu, t, s) = (8, 3, 63)$.

Example (OH $= 33.33\%$): Density Evolution and Thresholds



- Same parameter space as *[Zhang and Kschischang, JLT, 2014] → full table for all OHs in paper
- Result for OH $= 33.33\%$: $\mathcal{C}_2$ defined by $(\nu, t, s) = (8, 3, 63)$.

Example (OH $= 33.33\%$): Density Evolution and Thresholds



- Same parameter space as *[Zhang and Kschischang, JLT, 2014] → full table for all OHs in paper
- Result for OH $= 33.33\%$: $\mathcal{C}_2$ defined by $(\nu, t, s) = (8, 3, 63)$.
- Staircase codes with $\mathcal{C}_1$ and $\mathcal{C}_2$ have different slopes ⇒ DE gain prediction not preserved

Staircase Array with Multiple Row/Column Constraints

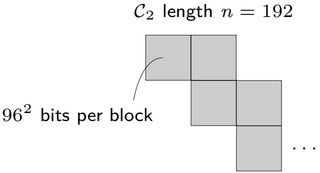
Staircase Array with Multiple Row/Column Constraints

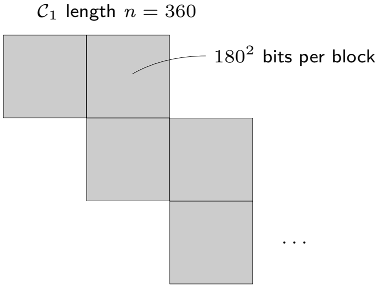
- “steepness” of BER curve determined by m, but fixed in the original construction

Staircase Array with Multiple Row/Column Constraints

- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$

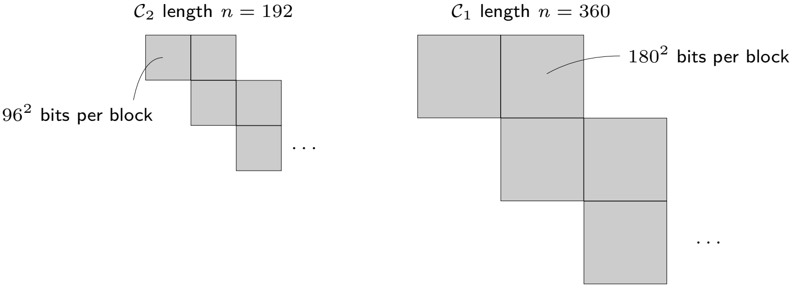
Staircase Array with Multiple Row/Column Constraints





- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$

Staircase Array with Multiple Row/Column Constraints



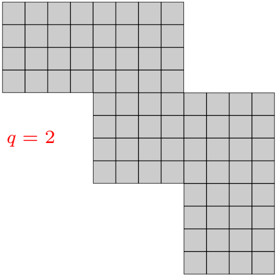
- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$
- Allow for $q > 1$ code constraints in each row/column of the staircase array

Staircase Array with Multiple Row/Column Constraints

- “steepness” of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$
- Allow for $q > 1$ code constraints in each row/column of the staircase array

Staircase Array with Multiple Row/Column Constraints

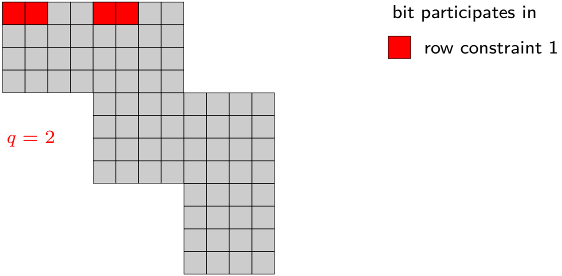
Example: $n = 4$, $q = 2$



- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$
- Allow for $q > 1$ code constraints in each row/column of the staircase array

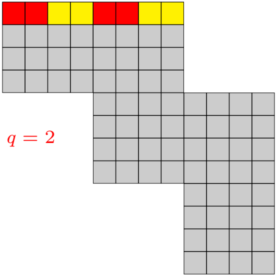
Staircase Array with Multiple Row/Column Constraints

Example: $n = 4$, $q = 2$



- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$
- Allow for $q > 1$ code constraints in each row/column of the staircase array

Staircase Array with Multiple Row/Column Constraints



Example: $n = 4, q = 2$

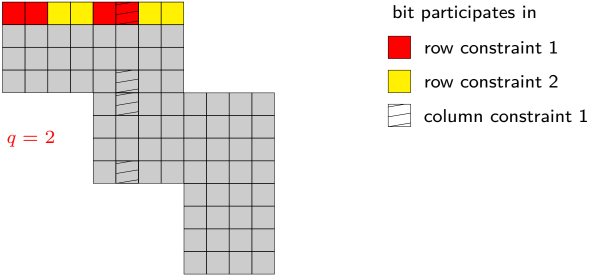
bit participates in

row constraint 1

row constraint 2

- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$
- Allow for $q > 1$ code constraints in each row/column of the staircase array

Staircase Array with Multiple Row/Column Constraints

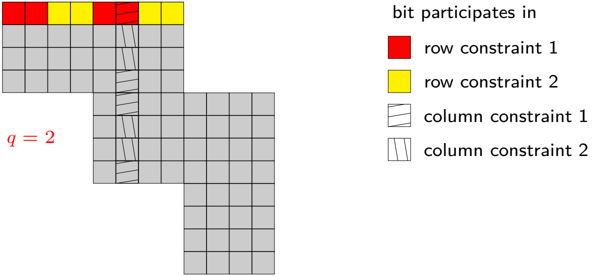


Example: $n = 4$, $q = 2$

- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$
- Allow for $q > 1$ code constraints in each row/column of the staircase array

Staircase Array with Multiple Row/Column Constraints

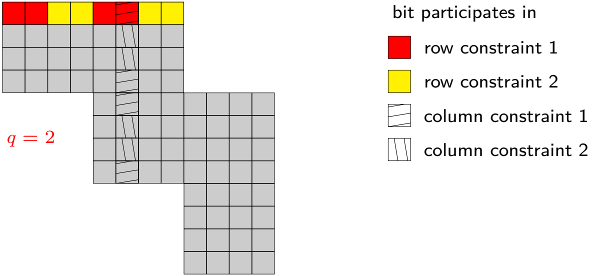
Example: $n = 4$, $q = 2$



- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$
- Allow for $q > 1$ code constraints in each row/column of the staircase array

Staircase Array with Multiple Row/Column Constraints

Example: $n = 4, q = 2$

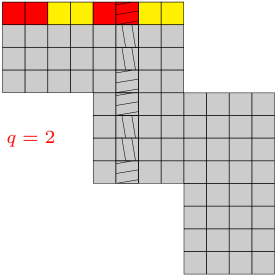


- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$
- Allow for $q > 1$ code constraints in each row/column of the staircase array
- Protograph lifting (copy-and-permute) of the Tanner graph describing the staircase code

CHALMERS

Staircase Array with Multiple Row/Column Constraints

Example: $n = 4$, $q = 2$

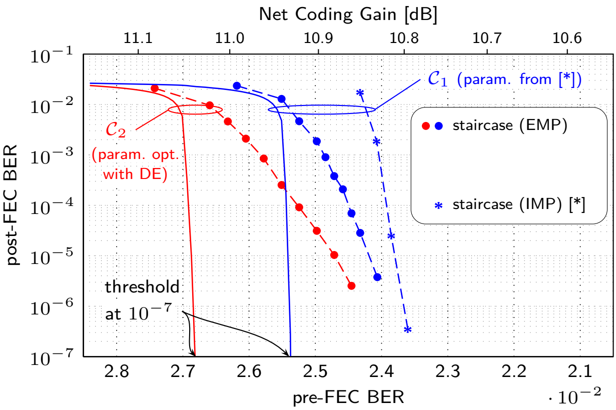


bit participates in

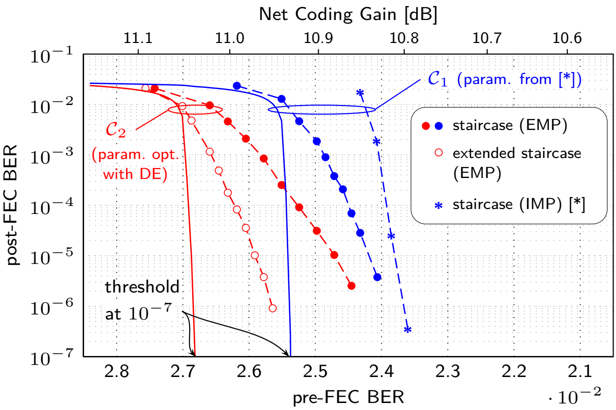
- row constraint 1
- row constraint 2
- column constraint 1
- column constraint 2

- "steepness" of BER curve determined by m, but fixed in the original construction
- Ensemble analysis: $m \to \infty$, staircase: $m = n/2$
- Allow for $q > 1$ code constraints in each row/column of the staircase array
- Protograph lifting (copy-and-permute) of the Tanner graph describing the staircase code
- The type of lifting preserves the staircase array structure and time-invariant encoding/decoding operations

Example (OH $= 33.33\%$): Extended Code Construction

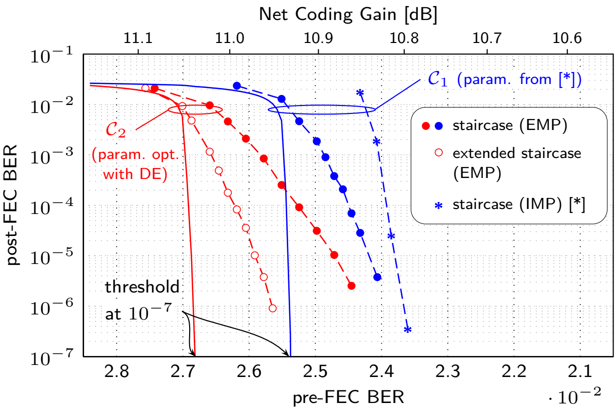


Example (OH $= 33.33\%$): Extended Code Construction



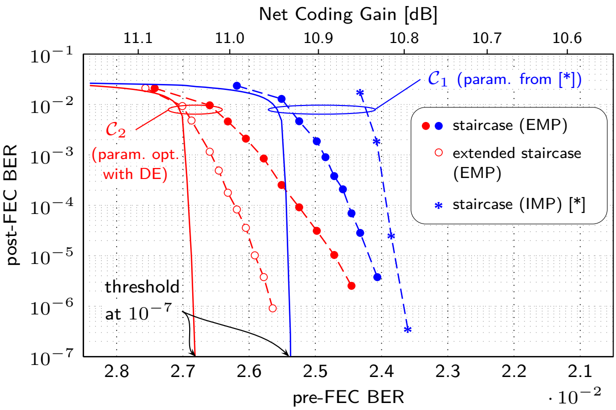
- Extended staircase code based on $\mathcal{C}_2$ for $q = 2$

Example (OH $= 33.33\%$): Extended Code Construction



- Extended staircase code based on $\mathcal{C}_2$ for $q = 2$
- Steeper waterfall performance at the expense of a larger staircase block size $2 \cdot n/2 = 192$

Example (OH $= 33.33\%$): Extended Code Construction



- Extended staircase code based on $\mathcal{C}_2$ for $q = 2$
- Steeper waterfall performance at the expense of a larger staircase block size $2 \cdot n/2 = 192$
- Staircase code with $\mathcal{C}_1$ has block size $n/2 = 180$

Conclusions

Conclusions

1. Density evolution can be used as an effective tool for finding good staircase code parameters.

Conclusions

1. Density evolution can be used as an effective tool for finding good staircase code parameters.
2. Extended staircase code construction can provide steeper waterfall performance at the expense of a larger staircase block size.

Conclusions

1. Density evolution can be used as an effective tool for finding good staircase code parameters.
2. Extended staircase code construction can provide steeper waterfall performance at the expense of a larger staircase block size.

Thank you!

FORCE
FIBER-OPTIC COMMUNICATIONS
RESEARCH CENTER

References

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

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.

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. and Kschischang, F. R. (2014).
Staircase codes with 6% to 33% overhead.
J. Lightw. Technol., 32(10):1999–2002.