Capacity Achieving Codes: There and Back Again

Henry D. Pfister

Electrical and Computer Engineering
Information Initiative (iiD)
Duke University

2016 European School of Information Theory
Chalmers University, Gothenburg, Sweden
April 6th, 2016

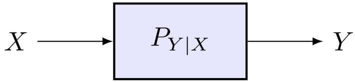
Acknowledgments

- Thanks to my coauthors involved in this work
  - Krishna Narayanan
  - Phong Nguyen
  - Arvind Yedla
  - Yung-Yih Jian
  - Santhosh Kumar
  - Shrinivas Kudekar, Marco Mondelli,
    Eren Sasoglu, Ruediger Urbanke
- Thanks to the organizers!
  - Alexandre Graell i Amat
  - Fredrik Brännström
  - Giuseppe Durisi

Outline

Outline

Capacity of Point-to-Point Communication



- Coding for Discrete-Time Memoryless Channels
  - Transition probability: $P_{Y|X}(y|x)$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$
  - Transmit a length-n codeword $\underline{x} \in \mathcal{C} \subset \mathcal{X}^n$
  - Decode to most likely codeword given received $\underline{y}$

Capacity of Point-to-Point Communication

$$X \longrightarrow \boxed{P_{Y|X}} \longrightarrow Y$$

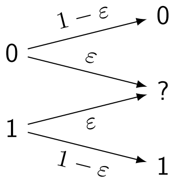
- Coding for Discrete-Time Memoryless Channels
  - Transition probability: $P_{Y|X}(y|x)$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$
  - Transmit a length-n codeword $\underline{x} \in \mathcal{C} \subset \mathcal{X}^n$
  - Decode to most likely codeword given received $\underline{y}$

- Channel Capacity introduced by Shannon in 1948
  - Random code of rate $R \triangleq \frac{1}{n} \log_2 |\mathcal{C}|$ (bits per channel use)
  - As $n \to \infty$, reliable transmission possible if $R < C$ with

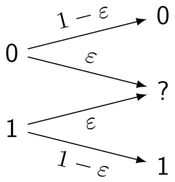
$$C \triangleq \max_{p(x)} I(X;Y)$$

The Binary Erasure Channel (BEC)

- Denoted BEC(ε) when erasure probability is ε
- $C = 1 - \varepsilon =$ expected fraction bits not erased

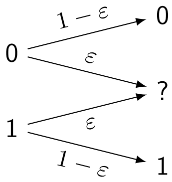


The Binary Erasure Channel (BEC)



- Denoted BEC(ε) when erasure probability is ε
- $C = 1 - \varepsilon =$ expected fraction bits not erased
- Coding with a binary linear code
  - Parity-check matrix $H \in \{0,1\}^{m \times n}$ with $m = (1-R)n$
  - Codebook $\mathcal{C} \triangleq \{\underline{x} \in \{0,1\}^n \mid H\underline{x} = \underline{0}\}$ has 2^{Rn} codewords

The Binary Erasure Channel (BEC)



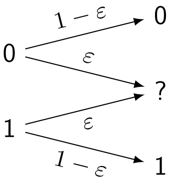
- Denoted BEC(ε) when erasure probability is ε
- $C = 1 - \varepsilon =$ expected fraction bits not erased

- Coding with a binary linear code
  - Parity-check matrix $H \in \{0,1\}^{m \times n}$ with $m = (1-R)n$
  - Codebook $\mathcal{C} \triangleq \{\underline{x} \in \{0,1\}^n \mid H\underline{x} = \underline{0}\}$ has 2^{Rn} codewords
  - Let $\mathcal{E}$ denote the index set of erased positions so that

$$H\underline{x} = \begin{bmatrix} H_{\mathcal{E}} & H_{\mathcal{E}^c} \end{bmatrix} \begin{bmatrix} \underline{x}_{\mathcal{E}} \\ \underline{y}_{\mathcal{E}^c} \end{bmatrix} = \underline{0} \quad \Leftrightarrow \quad H_{\mathcal{E}}\underline{x}_{\mathcal{E}} = -H_{\mathcal{E}^c}\underline{y}_{\mathcal{E}^c}$$

  - Decoding fails iff: $H_{\mathcal{E}}$ singular $\Leftrightarrow$ cw exists with 1's only in $\mathcal{E}$

The Binary Erasure Channel (BEC)



- Denoted BEC(ε) when erasure probability is ε
- $C = 1 - \varepsilon =$ expected fraction bits not erased
- Coding with a binary linear code
  - Parity-check matrix $H \in \{0,1\}^{m \times n}$ with $m = (1-R)n$
  - Codebook $\mathcal{C} \triangleq \{\underline{x} \in \{0,1\}^n \mid H\underline{x} = \underline{0}\}$ has 2^{Rn} codewords
  - Let $\mathcal{E}$ denote the index set of erased positions so that

$$H\underline{x} = \begin{bmatrix} H_{\mathcal{E}} & H_{\mathcal{E}^c} \end{bmatrix} \begin{bmatrix} \underline{x}_{\mathcal{E}} \\ \underline{y}_{\mathcal{E}^c} \end{bmatrix} = \underline{0} \quad \Leftrightarrow \quad H_{\mathcal{E}} \underline{x}_{\mathcal{E}} = -H_{\mathcal{E}^c} \underline{y}_{\mathcal{E}^c}$$

  - Decoding fails iff: $H_{\mathcal{E}}$ singular $\Leftrightarrow$ cw exists with 1's only in $\mathcal{E}$
  - One can achieve capacity by drawing H uniformly at random!

Some Early Milestones in Coding

- 1948: Shannon defines channel capacity and random codes

Some Early Milestones in Coding

- 1948: Shannon defines channel capacity and random codes
- 1950: Hamming formalizes linear codes and Hamming distance

Some Early Milestones in Coding

- 1948: Shannon defines channel capacity and random codes
- 1950: Hamming formalizes linear codes and Hamming distance
- 1954: Reed-Muller codes (Muller gives codes, Reed the decoder)

Some Early Milestones in Coding

- 1948: Shannon defines channel capacity and random codes
- 1950: Hamming formalizes linear codes and Hamming distance
- 1954: Reed-Muller codes (Muller gives codes, Reed the decoder)
- 1955: Elias introduces the erasure channel and convolutional codes; also shows random parity-check codes achieve capacity on the BEC

Some Early Milestones in Coding

- 1948: Shannon defines channel capacity and random codes
- 1950: Hamming formalizes linear codes and Hamming distance
- 1954: Reed-Muller codes (Muller gives codes, Reed the decoder)
- 1955: Elias introduces the erasure channel and convolutional codes; also shows random parity-check codes achieve capacity on the BEC
- 1959: BCH Codes (Hocquenghem'59 and Bose-Ray-Chaudhuri'60)

Some Early Milestones in Coding

- 1948: Shannon defines channel capacity and random codes
- 1950: Hamming formalizes linear codes and Hamming distance
- 1954: Reed-Muller codes (Muller gives codes, Reed the decoder)
- 1955: Elias introduces the erasure channel and convolutional codes; also shows random parity-check codes achieve capacity on the BEC
- 1959: BCH Codes (Hocquenghem'59 and Bose-Ray-Chaudhuri'60)
- 1960: Gallager introduces low-density parity-check (LDPC) codes and iterative decoding

Some Early Milestones in Coding

- 1948: Shannon defines channel capacity and random codes
- 1950: Hamming formalizes linear codes and Hamming distance
- 1954: Reed-Muller codes (Muller gives codes, Reed the decoder)
- 1955: Elias introduces the erasure channel and convolutional codes; also shows random parity-check codes achieve capacity on the BEC
- 1959: BCH Codes (Hocquenghem'59 and Bose-Ray-Chaudhuri'60)
- 1960: Gallager introduces low-density parity-check (LDPC) codes and iterative decoding
- 1960: Reed-Solomon codes

Achieving Capacity in Practice

But, more than 35 years passed before we could:

- Achieve capacity in practice
- Provably achieve capacity with determinstic constructions

Achieving Capacity in Practice

But, more than 35 years passed before we could:

- Achieve capacity in practice
- Provably achieve capacity with determinstic constructions

Modern Milestones:

- 1993: Turbo Codes (Berrou, Glavieux, Thitimajshima)

Achieving Capacity in Practice

But, more than 35 years passed before we could:

- Achieve capacity in practice
- Provably achieve capacity with determinstic constructions

Modern Milestones:

- 1993: Turbo Codes (Berrou, Glavieux, Thitimajshima)
- 1995: Rediscovery of LDPC codes (MacKay-Neal,Spielman)

Achieving Capacity in Practice

But, more than 35 years passed before we could:

- Achieve capacity in practice
- Provably achieve capacity with determinstic constructions

Modern Milestones:

- 1993: Turbo Codes (Berrou, Glavieux, Thitimajshima)
- 1995: Rediscovery of LDPC codes (MacKay-Neal,Spielman)
- 1997: Optimized irregular LDPC codes for the BEC (LMSSS)

Achieving Capacity in Practice

But, more than 35 years passed before we could:

- Achieve capacity in practice
- Provably achieve capacity with determinstic constructions

Modern Milestones:

- 1993: Turbo Codes (Berrou, Glavieux, Thitimajshima)
- 1995: Rediscovery of LDPC codes (MacKay-Neal,Spielman)
- 1997: Optimized irregular LDPC codes for the BEC (LMSSS)
- 2001: Optimized irregular LDPC codes for BMS channels (RSU)

Achieving Capacity in Practice

But, more than 35 years passed before we could:

- Achieve capacity in practice
- Provably achieve capacity with determinstic constructions

Modern Milestones:

- 1993: Turbo Codes (Berrou, Glavieux, Thitimajshima)
- 1995: Rediscovery of LDPC codes (MacKay-Neal,Spielman)
- 1997: Optimized irregular LDPC codes for the BEC (LMSSS)
- 2001: Optimized irregular LDPC codes for BMS channels (RSU)
- 2008: Polar codes provable, low-complexity, deterministic (Arikan)

Achieving Capacity in Practice

But, more than 35 years passed before we could:

- Achieve capacity in practice
- Provably achieve capacity with determinstic constructions

Modern Milestones:

- 1993: Turbo Codes (Berrou, Glavieux, Thitimajshima)
- 1995: Rediscovery of LDPC codes (MacKay-Neal,Spielman)
- 1997: Optimized irregular LDPC codes for the BEC (LMSSS)
- 2001: Optimized irregular LDPC codes for BMS channels (RSU)
- 2008: Polar codes provable, low-complexity, deterministic (Arikan)
- 1999-2011: Understanding LDPC convolutional codes and coupling

Key Tools That Made the Difference

- Factor Graph (FG)

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables
  - Natural setup for inference problems with partial observations

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables
  - Natural setup for inference problems with partial observations
- Belief-Propagation (BP)

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables
  - Natural setup for inference problems with partial observations
- Belief-Propagation (BP)
  - Message-passing algorithm for inference on a FG

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables
  - Natural setup for inference problems with partial observations
- Belief-Propagation (BP)
  - Message-passing algorithm for inference on a FG
  - Probability estimates are passed along edges in the factor graph

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables
  - Natural setup for inference problems with partial observations
- Belief-Propagation (BP)
  - Message-passing algorithm for inference on a FG
  - Probability estimates are passed along edges in the factor graph
  - Provides exact marginals if factor graph is a tree

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables
  - Natural setup for inference problems with partial observations
- Belief-Propagation (BP)
  - Message-passing algorithm for inference on a FG
  - Probability estimates are passed along edges in the factor graph
  - Provides exact marginals if factor graph is a tree
- Density Evolution (DE)

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables
  - Natural setup for inference problems with partial observations
- Belief-Propagation (BP)
  - Message-passing algorithm for inference on a FG
  - Probability estimates are passed along edges in the factor graph
  - Provides exact marginals if factor graph is a tree
- Density Evolution (DE)
  - Tracks distribution of messages passed by belief propagation

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables
  - Natural setup for inference problems with partial observations
- Belief-Propagation (BP)
  - Message-passing algorithm for inference on a FG
  - Probability estimates are passed along edges in the factor graph
  - Provides exact marginals if factor graph is a tree
- Density Evolution (DE)
  - Tracks distribution of messages passed by belief propagation
  - In some cases, allows rigorous analysis of BP-based inference

Key Tools That Made the Difference

- Factor Graph (FG)
  - Compact description of joint distribution for random variables
  - Natural setup for inference problems with partial observations
- Belief-Propagation (BP)
  - Message-passing algorithm for inference on a FG
  - Probability estimates are passed along edges in the factor graph
  - Provides exact marginals if factor graph is a tree
- Density Evolution (DE)
  - Tracks distribution of messages passed by belief propagation
  - In some cases, allows rigorous analysis of BP-based inference
- EXtrinsic Information Transfer (EXIT) Curves

Applications of These Tools

- Error-Correcting Codes
  - Random code defined by random factor graph
  - Low-complexity decoding via belief propagation
  - Analysis of belief-propagation decoding via density evolution
  - Provides code constructions that provably achieve capacity!

Applications of These Tools

- Error-Correcting Codes
  - Random code defined by random factor graph
  - Low-complexity decoding via belief propagation
  - Analysis of belief-propagation decoding via density evolution
  - Provides code constructions that provably achieve capacity!
- Boolean Satisfiability: K-SAT
  - Random instance of K-SAT defined by random factor graph
  - Non-rigorous analysis via the cavity method
  - Predicted thresholds later proved exact!

Applications of These Tools

- Error-Correcting Codes
  - Random code defined by random factor graph
  - Low-complexity decoding via belief propagation
  - Analysis of belief-propagation decoding via density evolution
  - Provides code constructions that provably achieve capacity!
- Boolean Satisfiability: K-SAT
  - Random instance of K-SAT defined by random factor graph
  - Non-rigorous analysis via the cavity method
  - Predicted thresholds later proved exact!
- Compressed Sensing
  - Random measurement matrix defined by random factor graph
  - Low-complexity reconstruction via message passing
  - Schemes provably achieve the information-theoretic limit!

Polya's Dictum

If you can't solve a problem, then it probably contains an easier problem that you can't solve: find it.

Polya's Dictum

If you can't solve a problem, then it probably contains an easier problem that you can't solve: find it.

- The solution of the simpler problem often provides insight that allows one to crack the harder problem.

Polya's Dictum

If you can't solve a problem, then it probably contains an easier problem that you can't solve: find it.

- The solution of the simpler problem often provides insight that allows one to crack the harder problem.
- To achieve channel capacity in practice, we now know that a good "easy" problem would have been:
  - "Design a code that achieves capacity on the BEC and is encodable and decodable in quasi-linear time"

Outline

Factor Graphs

- A factor graph provides a graphical representation of the local dependence structure for a set of random variables
  - Bipartite graph with variables $x_1, \ldots, x_n$ and factors $f_1, \ldots, f_m$

Factor Graphs

- A factor graph provides a graphical representation of the local dependence structure for a set of random variables
  - Bipartite graph with variables $x_1, \ldots, x_n$ and factors $f_1, \ldots, f_m$
- Consider random variables $(X_1, X_2, \ldots, X_4) \in \mathcal{X}^4$ and Y where:

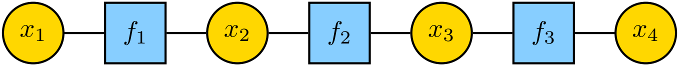
$$
\begin{aligned}
P(x_1, x_2, x_3, x_4) &\triangleq \mathbb{P}(X_1 = x_1, X_2 = x_2, \ldots, X_4 = x_4 | Y = y) \\
&\propto f(x_1, x_2, x_3, x_4) \\
&\triangleq f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_3, x_4)
\end{aligned}
$$

Factor Graphs

- A factor graph provides a graphical representation of the local dependence structure for a set of random variables
  - Bipartite graph with variables $x_1, \ldots, x_n$ and factors $f_1, \ldots, f_m$
- Consider random variables $(X_1, X_2, \ldots, X_4) \in \mathcal{X}^4$ and Y where:

$$
\begin{aligned}
P(x_1, x_2, x_3, x_4) &\triangleq \mathbb{P}(X_1 = x_1, X_2 = x_2, \ldots, X_4 = x_4 | Y = y) \\
&\propto f(x_1, x_2, x_3, x_4) \\
&\triangleq f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_3, x_4)
\end{aligned}
$$

- Given $Y = y$, this describes a Markov chain whose factor graph is



Conditional Independence for Factor Graphs

- Let $A, B, S \subset [n]$ be disjoint subsets of VNs in factor graph G
  - If S separates A from B (i.e., there is no path in G from A to B that avoids S), then we have $X_A \perp\!\!\!\perp X_B \mid X_S$

$$P(x_A, x_B | x_S) = P(x_A | x_S) P(x_B | x_S)$$

Conditional Independence for Factor Graphs

- Let $A, B, S \subset [n]$ be disjoint subsets of VNs in factor graph G
  - If S separates A from B (i.e., there is no path in G from A to B that avoids S), then we have $X_A \perp\!\!\!\perp X_B \mid X_S$

$$P(x_A, x_B|x_S) = P(x_A|x_S)P(x_B|x_S)$$

  - Markov chain example: $A = \{x_1, x_2\}$, $B = \{x_4\}$, $S = \{x_3\}$

Conditional Independence for Factor Graphs

- Let $A, B, S \subset [n]$ be disjoint subsets of VNs in factor graph G
  - If S separates A from B (i.e., there is no path in G from A to B that avoids S), then we have $X_A \perp\!\!\!\perp X_B \mid X_S$

$$P(x_A, x_B | x_S) = P(x_A | x_S) P(x_B | x_S)$$

  - Markov chain example: $A = \{x_1, x_2\}$, $B = \{x_4\}$, $S = \{x_3\}$

- Sketch of Proof:
  - Fixing $X_S = x_S$ separates the FG into disjoint components

Conditional Independence for Factor Graphs

- Let $A, B, S \subset [n]$ be disjoint subsets of VNs in factor graph G
  - If S separates A from B (i.e., there is no path in G from A to B that avoids S), then we have $X_A \perp\!\!\!\perp X_B \mid X_S$

$$P(x_A, x_B | x_S) = P(x_A | x_S) P(x_B | x_S)$$

  - Markov chain example: $A = \{x_1, x_2\}$, $B = \{x_4\}$, $S = \{x_3\}$

- Sketch of Proof:
  - Fixing $X_S = x_S$ separates the FG into disjoint components
  - Groups of VNs in different components are independent
  - $X_A \perp\!\!\!\perp X_B$ because A and B are in different components

Inference via Marginalization

- Marginalizing out all variables except X_1 gives

$$\mathbb{P}(X_1 = x_1 | Y = y) \propto g_1(x_1) \triangleq \sum_{(x_2,\ldots,x_4)\in\mathcal{X}^3} f(x_1, x_2, x_3, x_4)$$

- Thus, the maximum a posteriori decision for X_1 given $Y = y$ is

$$\hat{x}_1 = \arg\max_{x_1\in\mathcal{X}} \sum_{(x_2,\ldots,x_4)\in\mathcal{X}^3} f(x_1, x_2, x_3, x_4)$$

  - For a general function, this requires roughly $|\mathcal{X}|^4$ operations

Inference via Marginalization

- Marginalizing out all variables except X_1 gives

$$\mathbb{P}(X_1 = x_1 | Y = y) \propto g_1(x_1) \triangleq \sum_{(x_2,\ldots,x_4)\in\mathcal{X}^3} f(x_1, x_2, x_3, x_4)$$

- Thus, the maximum a posteriori decision for X_1 given $Y = y$ is

$$\hat{x}_1 = \arg\max_{x_1\in\mathcal{X}} \sum_{(x_2,\ldots,x_4)\in\mathcal{X}^3} f(x_1, x_2, x_3, x_4)$$

  - For a general function, this requires roughly $|\mathcal{X}|^4$ operations

- Marginalization is efficient for tree-structured factor graphs
  - For the Markov chain, roughly $5\,|\mathcal{X}|^2$ operations required

$$g_1(x_1) = \sum_{x_2\in\mathcal{X}} f_1(x_1, x_2) \sum_{x_3\in\mathcal{X}} f_2(x_2, x_3) \sum_{x_4\in\mathcal{X}} f_3(x_3, x_4)$$

The Importance of Factorization (1)

- Consider a random vector $(X_1, X_2, \ldots, X_6) \in \mathcal{X}^6$ where

$$\mathbb{P}(X_1 = x_1, \ldots, X_6 = x_6 | Y = y) \propto f(x_1, x_2, x_3, x_4, x_5, x_6)$$

The Importance of Factorization (1)

- Consider a random vector $(X_1, X_2, \ldots, X_6) \in \mathcal{X}^6$ where

$$\mathbb{P}(X_1 = x_1, \ldots, X_6 = x_6 | Y = y) \propto f(x_1, x_2, x_3, x_4, x_5, x_6)$$

- Brute force marginal requires $|\mathcal{X}|^5$ operations for each $x_1 \in \mathcal{X}$:

$$g_1(x_1) \triangleq \sum_{x_2^6 \in \mathcal{X}^5} f(x_1, x_2, x_3, x_4, x_5, x_6)$$

  - Thus, we need $|\mathcal{X}|^6$ operations

The Importance of Factorization (1)

- Consider a random vector $(X_1, X_2, \ldots, X_6) \in \mathcal{X}^6$ where

$$\mathbb{P}(X_1 = x_1, \ldots, X_6 = x_6 | Y = y) \propto f(x_1, x_2, x_3, x_4, x_5, x_6)$$

- Brute force marginal requires $|\mathcal{X}|^5$ operations for each $x_1 \in \mathcal{X}$:

$$g_1(x_1) \triangleq \sum_{x_2^6 \in \mathcal{X}^5} f(x_1, x_2, x_3, x_4, x_5, x_6)$$

  - Thus, we need $|\mathcal{X}|^6$ operations

- If f factors as follows, then the marginalization can be simplified:

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$$

The Importance of Factorization (2)

For example, we can write $g_1(x_1)$ as:

$$= \sum_{x_2^6} f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$$

The Importance of Factorization (2)

For example, we can write $g_1(x_1)$ as:

$$= \sum_{x_2^6} f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$$

$$= \sum_{x_2^5} f_1(x_1, x_2, x_3) f_3(x_4) f_4(x_4, x_5) \left[\sum_{x_6} f_2(x_1, x_4, x_6) \right]$$

The Importance of Factorization (2)

For example, we can write $g_1(x_1)$ as:

$$= \sum_{x_2^6} f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$$

$$= \sum_{x_2^5} f_1(x_1, x_2, x_3) f_3(x_4) f_4(x_4, x_5) \left[\sum_{x_6} f_2(x_1, x_4, x_6) \right]$$

$$= \sum_{x_2^4} f_1(x_1, x_2, x_3) f_3(x_4) \left[\sum_{x_5} f_4(x_4, x_5) \right] \left[\sum_{x_6} f_2(x_1, x_4, x_6) \right]$$

The Importance of Factorization (2)

For example, we can write $g_1(x_1)$ as:

$$
\begin{aligned}
&= \sum_{x_2^6} f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5) \\
&= \sum_{x_2^5} f_1(x_1, x_2, x_3) f_3(x_4) f_4(x_4, x_5) \left[\sum_{x_6} f_2(x_1, x_4, x_6) \right] \\
&= \sum_{x_2^4} f_1(x_1, x_2, x_3) f_3(x_4) \left[\sum_{x_5} f_4(x_4, x_5) \right] \left[\sum_{x_6} f_2(x_1, x_4, x_6) \right] \\
&= \sum_{x_2^3} f_1(x_1, x_2, x_3) \left[\sum_{x_4} f_3(x_4) \left[\sum_{x_5} f_4(x_4, x_5) \right] \left[\sum_{x_6} f_2(x_1, x_4, x_6) \right] \right]
\end{aligned}
$$

The Importance of Factorization (2)

For example, we can write $g_1(x_1)$ as:

$$=\sum_{x_2^6} f_1(x_1,x_2,x_3)f_2(x_1,x_4,x_6)f_3(x_4)f_4(x_4,x_5)$$

$$=\sum_{x_2^5} f_1(x_1,x_2,x_3)f_3(x_4)f_4(x_4,x_5)\left[\sum_{x_6} f_2(x_1,x_4,x_6)\right]$$

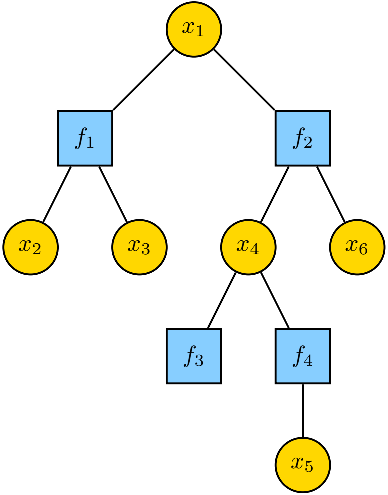
$$=\sum_{x_2^4} f_1(x_1,x_2,x_3)f_3(x_4)\left[\sum_{x_5} f_4(x_4,x_5)\right]\left[\sum_{x_6} f_2(x_1,x_4,x_6)\right]$$

$$=\sum_{x_2^3} f_1(x_1,x_2,x_3)\left[\sum_{x_4} f_3(x_4)\left[\sum_{x_5} f_4(x_4,x_5)\right]\left[\sum_{x_6} f_2(x_1,x_4,x_6)\right]\right]$$

$$=\sum_{x_2}\left[\sum_{x_3} f_1(x_1,x_2,x_3)\right]\left[\sum_{x_4} f_3(x_4)\left[\sum_{x_5} f_4(x_4,x_5)\right]\left[\sum_{x_6} f_2(x_1,x_4,x_6)\right]\right]$$

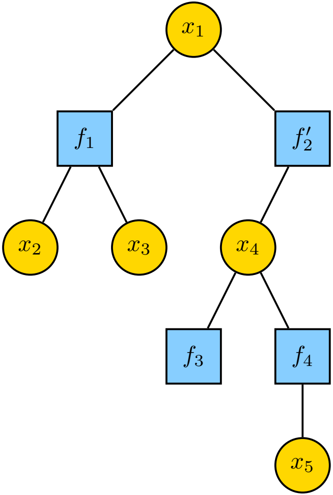
This implementation requires roughly $2\,|\mathcal{X}|^3+5\,|\mathcal{X}|^2$ operations

The Factor Graph and Leaf Removal



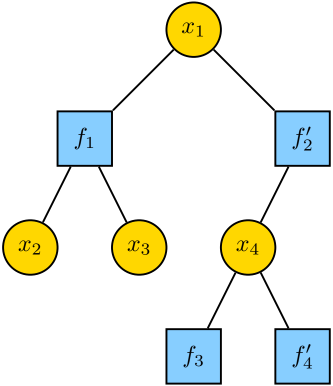
$$g_1(x_1) = \sum_{x_2^5} f_1(x_1, x_2, x_3) f_3(x_4) f_4(x_4, x_5) \sum_{x_6} f_2(x_1, x_4, x_6)$$

The Factor Graph and Leaf Removal



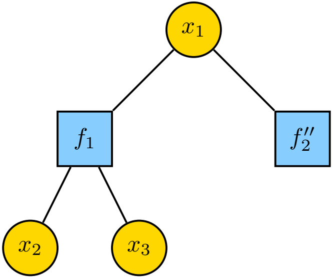
$$g_1(x_1) = \sum_{x_2^4} f_1(x_1, x_2, x_3) f_3(x_4) \left[\sum_{x_5} f_4(x_4, x_5) \right] f_2'(x_1, x_4)$$

The Factor Graph and Leaf Removal



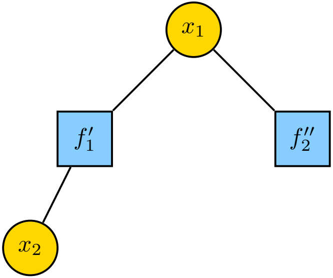
$$g_1(x_1) = \sum_{x_2^3} f_1(x_1, x_2, x_3) \left[\sum_{x_4} f_3(x_4) f_4'(x_4) f_2'(x_1, x_4) \right]$$

The Factor Graph and Leaf Removal



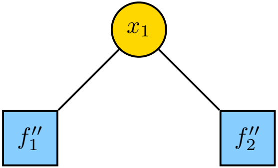
$$g_1(x_1) = \sum_{x_2} \left[\sum_{x_3} f_1(x_1, x_2, x_3) \right] f_2''(x_1)$$

The Factor Graph and Leaf Removal



$$g_1(x_1) = \left[\sum_{x_2} f_1'(x_1, x_2)\right] f_2''(x_1)$$

The Factor Graph and Leaf Removal



$$g_1(x_1) = f_1''(x_1) f_2''(x_1)$$

Constraint Satisfaction and Zero-One Factors

- A non-negative function $f\colon \mathcal{X}^n \to \mathbb{R}$ defines a distribution on $\mathcal{X}^n$:

$$
\begin{aligned}
P(\underline{x}) &\triangleq \mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) \\
&= \frac{1}{Z} f(\underline{x}) \triangleq \frac{1}{Z} \prod_{a=1}^{m} f_a(\underline{x}_{\partial a}),
\end{aligned}
$$

  - where $\underline{x}_{\partial a}$ is the subvector of variables involved in factor a
  - and $Z \triangleq \sum_{\underline{x}} f(\underline{x})$ is called the partition function

Constraint Satisfaction and Zero-One Factors

- A non-negative function $f\colon \mathcal{X}^n \to \mathbb{R}$ defines a distribution on $\mathcal{X}^n$:

$$P(\underline{x}) \triangleq \mathbb{P}(X_1 = x_1, \ldots, X_n = x_n)$$
$$= \frac{1}{Z} f(\underline{x}) \triangleq \frac{1}{Z} \prod_{a=1}^{m} f_a(\underline{x}_{\partial a}),$$

  - where $\underline{x}_{\partial a}$ is the subvector of variables involved in factor a
  - and $Z \triangleq \sum_{\underline{x}} f(\underline{x})$ is called the partition function

- For Constraint Satisfaction Problems (CSPs)
  - All factors $f_a(\underline{x}_{\partial a})$ take values in $\{0, 1\}$
  - The set of valid configurations is $\{\underline{x} \in \mathcal{X}^n | f(\underline{x}) = 1\}$
  - Thus, Z equals the number of valid configurations
  - $P(\underline{x})$ is uniform over the set of valid configurations

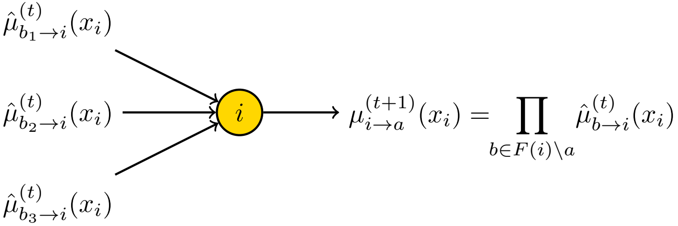
Outline

Marginalization via Belief Propagation

- Factor Graph $G = (V \cup F, E)$
  - Variable nodes V, Factor nodes F
  - Edges: $(i, a) \in E \subseteq V \times F$
  - $F(i)/V(a) =$ set of neighbors for node-i/a
  - Messages: $\mu_{i \to a}^{(t)}(x_i)$ and $\hat{\mu}_{a \to i}^{(t)}(x_i)$

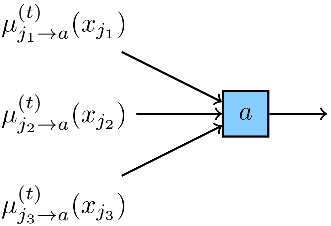
Marginalization via Belief Propagation

- Factor Graph $G = (V \cup F, E)$
  - Variable nodes V, Factor nodes F
  - Edges: $(i, a) \in E \subseteq V \times F$
  - $F(i)/V(a) =$ set of neighbors for node-i/a
  - Messages: $\mu_{i \to a}^{(t)}(x_i)$ and $\hat{\mu}_{a \to i}^{(t)}(x_i)$
- variable-i to factor-a message



Marginalization via Belief Propagation

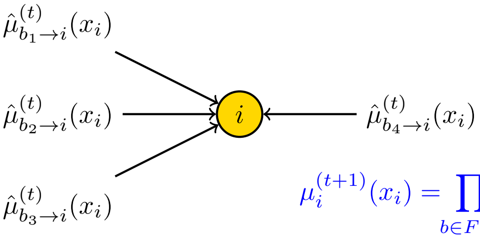
- Factor Graph $G = (V \cup F, E)$
  - Variable nodes V, Factor nodes F
  - Edges: $(i, a) \in E \subseteq V \times F$
  - $F(i)/V(a) =$ set of neighbors for node-i/a
  - Messages: $\mu^{(t)}_{i \to a}(x_i)$ and $\hat{\mu}^{(t)}_{a \to i}(x_i)$
- factor-a to variable-i message



$$\hat{\mu}^{(t)}_{a \to i}(x_i) = \sum_{\underline{x}_{V(a)\setminus i}} f_a(\underline{x}_{V(a)}) \prod_{j \in V(a)\setminus i} \mu^{(t)}_{j \to a}(x_j)$$

Marginalization via Belief Propagation

- Factor Graph $G = (V \cup F, E)$
  - Variable nodes V, Factor nodes F
  - Edges: $(i, a) \in E \subseteq V \times F$
  - $F(i)/V(a) =$ set of neighbors for node-i/a
  - Messages: $\mu^{(t)}_{i \to a}(x_i)$ and $\hat{\mu}^{(t)}_{a \to i}(x_i)$
- variable-i marginal

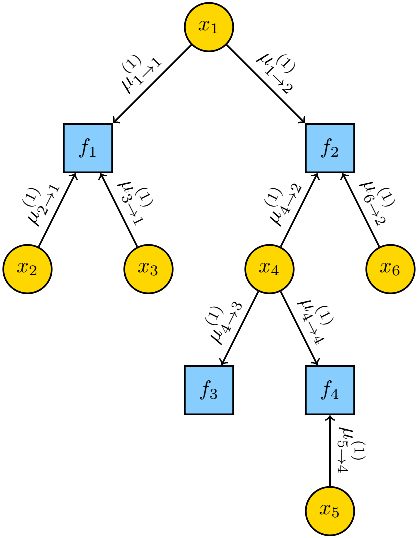


$$\mu^{(t+1)}_{i}(x_i) = \prod_{b \in F(i)} \hat{\mu}^{(t)}_{b \to i}(x_i)$$

Marginalization via Belief Propagation: Example

iteration 1: variable to factor

$$\mu_{i\to a}^{(1)}(x_i) = 1$$



Marginalization via Belief Propagation: Example

iteration 1: variable to factor

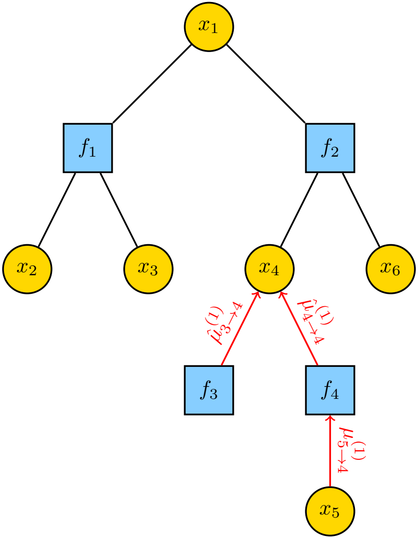
$$\mu_{i\to a}^{(1)}(x_i) = 1$$

iteration 1: factor to variable

$$\hat{\mu}_{4\to 4}^{(1)}(x_4) = \sum_{x_5} f_4(x_4, x_5)\mu_{5\to 4}^{(1)}(x_i)$$

$$= \sum_{x_5} f_4(x_4, x_5)$$

$$\hat{\mu}_{3\to 4}^{(1)}(x_4) = f_3(x_4)$$



Marginalization via Belief Propagation: Example

iteration 1: factor to variable

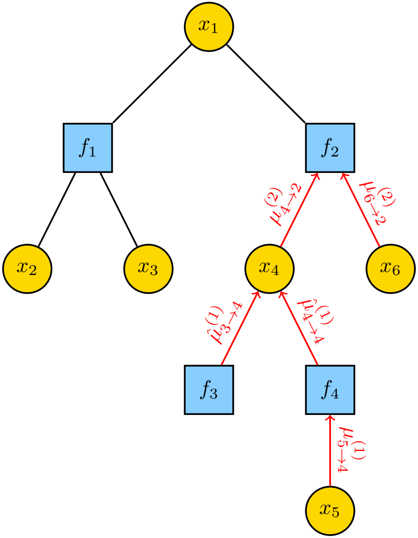
$$\hat{\mu}^{(1)}_{4\to 4}(x_4) = \sum_{x_5} f_4(x_4, x_5)\mu^{(1)}_{5\to 4}(x_i)$$
$$= \sum_{x_5} f_4(x_4, x_5)$$

$$\hat{\mu}^{(1)}_{3\to 4}(x_4) = f_3(x_4)$$

iteration 2: variable to factor

$$\mu^{(2)}_{4\to 2}(x_4) = \hat{\mu}^{(1)}_{4\to 4}(x_4)\hat{\mu}^{(1)}_{3\to 4}(x_4)$$
$$= f_3(x_4)\sum_{x_5} f_4(x_4, x_5)$$

$$\mu^{(2)}_{6\to 2}(x_6) = 1$$



Marginalization via Belief Propagation: Example

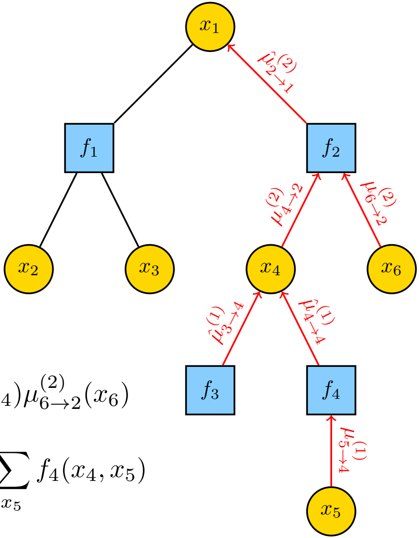
iteration 2: variable to factor

$$\mu^{(2)}_{4\to 2}(x_4) = \hat{\mu}^{(1)}_{4\to 4}(x_4)\hat{\mu}^{(1)}_{3\to 4}(x_4)$$
$$= f_3(x_4)\sum_{x_5} f_4(x_4, x_5)$$

$$\mu^{(2)}_{6\to 2}(x_6) = 1$$

iteration 2: factor to variable

$$\hat{\mu}^{(2)}_{2\to 1}(x_1) = \sum_{x_4, x_6} f_2(x_1, x_4, x_6)\mu^{(2)}_{4\to 2}(x_4)\mu^{(2)}_{6\to 2}(x_6)$$
$$= \sum_{x_4, x_6} f_2(x_1, x_4, x_6) f_3(x_4)\sum_{x_5} f_4(x_4, x_5)$$
$$= f''_2(x_1)$$

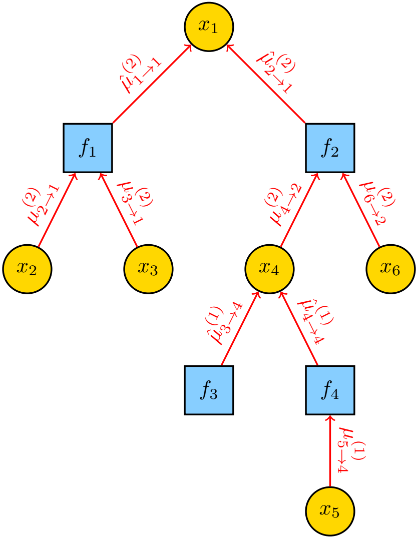


Marginalization via Belief Propagation: Example

iteration 2: variable marginal

$$\begin{aligned} \mu_1^{(3)}(x_1) &= \hat{\mu}_{1\to 1}^{(2)}(x_1)\hat{\mu}_{2\to 1}^{(2)}(x_1) \\ &= f_1''(x_1) f_2''(x_2) \end{aligned}$$

Same answer as peeling but from a distributed parallel algorithm



Outline

Sudoku: A Factor Graph for the Masses

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

rows are permutations of $\{1, 2, \ldots, 9\}$

Sudoku: A Factor Graph for the Masses

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

rows are permutations of $\{1, 2, \ldots, 9\}$
columns are permutations of $\{1, 2, \ldots, 9\}$

Sudoku: A Factor Graph for the Masses

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

rows are permutations of $\{1, 2, \ldots, 9\}$
columns are permutations of $\{1, 2, \ldots, 9\}$
subblocks are permutations of $\{1, 2, \ldots, 9\}$

Sudoku: A Factor Graph for the Masses

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

rows are permutations of $\{1, 2, \ldots, 9\}$
columns are permutations of $\{1, 2, \ldots, 9\}$
subblocks are permutations of $\{1, 2, \ldots, 9\}$

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}	x_{38}	x_{39}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}	x_{48}	x_{49}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	x_{57}	x_{58}	x_{59}
x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}	x_{67}	x_{68}	x_{69}
x_{71}	x_{72}	x_{73}	x_{74}	x_{75}	x_{76}	x_{77}	x_{78}	x_{79}
x_{81}	x_{82}	x_{83}	x_{84}	x_{85}	x_{86}	x_{87}	x_{88}	x_{89}
x_{91}	x_{92}	x_{93}	x_{94}	x_{95}	x_{96}	x_{97}	x_{98}	x_{99}

implied factor graph has
81 variable and 27 factor nodes

$$f(\underline{x}) = \left(\prod_{i=1}^{9} f_\sigma(x_{i*})\right) \left(\prod_{j=1}^{9} f_\sigma(x_{*j})\right) \left(\prod_{k=1}^{9} f_\sigma(x_{B(k)})\right) \prod_{(i,j)\in O} \mathbb{I}(x_{ij} = y_{ij})$$

Solving Sudoku with a Factor Graph

- Consider any constraint satisfaction problem with observed entries
  - One can write $f(\underline{x})$ as the product of indicator functions
  - Some factors force $\underline{x}$ to be valid (i.e., satisfy constraints)
  - Other factors force $\underline{x}$ to be compatible with observed values
  - Summing over $\underline{x}$ counts the $\#$ of valid compatible sequences

Solving Sudoku with a Factor Graph

- Consider any constraint satisfaction problem with observed entries
  - One can write $f(\underline{x})$ as the product of indicator functions
  - Some factors force $\underline{x}$ to be valid (i.e., satisfy constraints)
  - Other factors force $\underline{x}$ to be compatible with observed values
  - Summing over $\underline{x}$ counts the $\#$ of valid compatible sequences
- Low-complexity peeling solution
  - Set elements of $\underline{x}$ one at a time
  - Each step looks for $i \in [n]$ and $x' \in \mathcal{X}$ such that:
    - For currently set variables, $f(\underline{x}) = 0$ for all $x_i \in \mathcal{X} \setminus x'$
  - Sudoku's unique solution implies that $x_i = x'$ correct
  - Fix $x_i = x'$ and repeat until all values fixed

Boolean Satisfiability: K-SAT

- One instance of 3-SAT is given, for example, by

$$f(\underline{x}) = (\overline{x}_1 \vee \overline{x}_3 \vee x_7) \wedge (x_1 \vee \overline{x}_2 \vee x_5) \wedge (x_2 \vee \overline{x}_4 \vee x_6)\,.$$

  - In the FG, clause $a \in [m]$ is enforced by the function f_a

Boolean Satisfiability: K-SAT

- One instance of 3-SAT is given, for example, by

$$f(\underline{x}) = (\overline{x}_1 \vee \overline{x}_3 \vee x_7) \wedge (x_1 \vee \overline{x}_2 \vee x_5) \wedge (x_2 \vee \overline{x}_4 \vee x_6)\,.$$

  - In the FG, clause $a \in [m]$ is enforced by the function f_a

- Marginalization allows uniform sampling from valid set
  - For $i = 1, 2, \ldots, n$, fix x_j for $j < i$ and compute marginal

$$g_i(x_i) = \frac{1}{Z_i} \sum_{x_{i+1},\ldots,x_n} f(\underline{x}) = \mathbb{P}\left(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}\right)$$

  - Then, sample $x_i \sim g_i(\cdot)$ and repeat

Boolean Satisfiability: K-SAT

- One instance of 3-SAT is given, for example, by

$$f(\underline{x}) = (\overline{x}_1 \vee \overline{x}_3 \vee x_7) \wedge (x_1 \vee \overline{x}_2 \vee x_5) \wedge (x_2 \vee \overline{x}_4 \vee x_6)\,.$$

  - In the FG, clause $a \in [m]$ is enforced by the function f_a

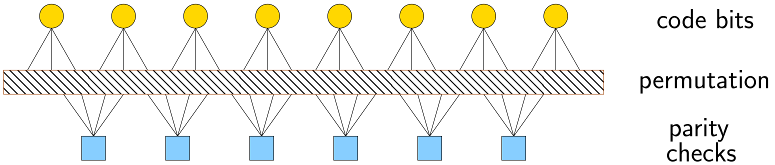
- Marginalization allows uniform sampling from valid set
  - For $i = 1, 2, \ldots, n$, fix x_j for $j < i$ and compute marginal

$$g_i(x_i) = \frac{1}{Z_i} \sum_{x_{i+1},\ldots,x_n} f(\underline{x}) = \mathbb{P}\left(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}\right)$$

  - Then, sample $x_i \sim g_i(\cdot)$ and repeat

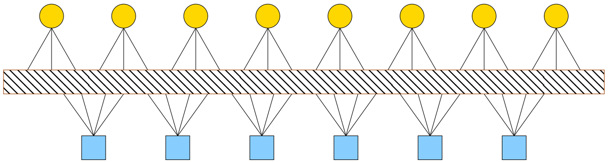
- This algorithm has low complexity if factor graph forms a tree
  - If not a tree, use approximate marginal from belief propagation
  - This is related to BP-guided decimation [MM09]

Low-Density Parity-Check (LDPC) Codes



- Linear codes defined by $\underline{x}H^T = \underline{0}$ for all c.w. $\underline{x} \in \mathcal{C} \subset \{0,1\}^n$
  - H is an $m \times n$ sparse parity-check matrix for the code
  - Code bits and parity checks associated with cols/rows of H

Low-Density Parity-Check (LDPC) Codes



- Linear codes defined by $\underline{x}H^T = \underline{0}$ for all c.w. $\underline{x} \in \mathcal{C} \subset \{0,1\}^n$
  - H is an $m \times n$ sparse parity-check matrix for the code
  - Code bits and parity checks associated with cols/rows of H
- Factor graph: H is the biadjacency matrix for variable/factor nodes
  - Ensemble defined by configuration model for random graphs
  - Checks define factors: $f_{\text{even}}(x_1^d) = \mathbb{I}(x_1 \oplus \cdots \oplus x_d = 0)$
  - Let $\underline{x}_{\partial a}$ be the subvector of variables in the a-th check and

$$f(x_1, \ldots, x_n) = \left(\prod_{a=1}^{m} f_{\text{even}}(\underline{x}_{\partial a}) \right) \left(\prod_{i=1}^{n} P_{Y|X}(y_i|x_i) \right)$$

A Little History

Robert Gallager

introduced LDPC codes in 1962 paper

1962 *IRE TRANSACTIONS ON INFORMATION THEORY* 21

Low-Density Parity-Check Codes*

R. G. GALLAGER†

Summary—A low-density parity-check code is a code specified by a parity-check matrix with the following properties: each column contains a small fixed number $j \geq 3$ of 1's and each row contains a small fixed number $k > j$ of 1's. The typical minimum distance of these codes increases linearly with block length for a fixed rate and fixed j. When used with maximum likelihood decoding on a sufficiently quiet binary-input symmetric channel, the typical probability of decoding error decreases exponentially with block length for a fixed rate and fixed j.

A simple but nonoptimum decoding scheme operating directly from the channel a posteriori probabilities is described. Both the

equations. We call the set of digits contained in a parity-check equation a parity-check set. For example, the first parity-check set in Fig. 1 is the set of digits (1, 2, 3, 5).

The use of parity-check codes makes coding (as distinguished from decoding) relatively simple to implement. Also, as Elias [3] has shown, if a typical parity-check code of long block length is used on a binary symmetric channel, and if the code rate is between *critical rate* and channel capacity, then the probability of decoding error

Judea Pearl

defined general belief-propagation in 1986 paper

Fusion, Propagation, and Structuring in Belief Networks*

Judea Pearl
Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, CA 90024, U.S.A.

Recommended by Patrick Hayes

ABSTRACT

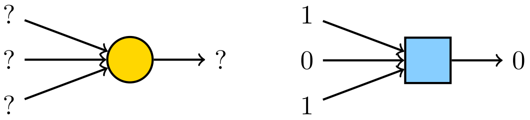
Belief networks are directed acyclic graphs in which the nodes represent propositions (or variables), the arcs signify direct dependencies between the linked propositions, and the strengths of these dependencies are quantified by conditional probabilities. A network of this sort can be used to represent the generic knowledge of a domain expert, and it turns into a computational architecture if the links are used not merely for storing factual knowledge but also for directing and activating the data flow in the computations which manipulate this knowledge.

Simple Message-Passing Decoding for the BEC

- Constraint nodes define the valid patterns
  - Circles represent a single value shared by factors
  - Squares assert attached variables sum to 0 mod 2
- Iterative decoding on the binary erasure channel (BEC)
  - Messages passed in phases: bit-to-check and check-to-bit
  - Each output message depends on other input messages
  - Each message is either the correct value or an erasure

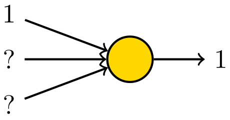
Simple Message-Passing Decoding for the BEC

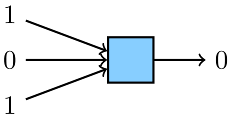
- Constraint nodes define the valid patterns
  - Circles represent a single value shared by factors
  - Squares assert attached variables sum to 0 mod 2
- Iterative decoding on the binary erasure channel (BEC)
  - Messages passed in phases: bit-to-check and check-to-bit
  - Each output message depends on other input messages
  - Each message is either the correct value or an erasure
- Message passing rules for the BEC
  - Bits pass an erasure only if all other inputs are erased
  - Checks pass the correct value only if all other inputs are correct



Simple Message-Passing Decoding for the BEC

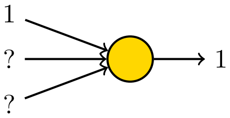
- Constraint nodes define the valid patterns
  - Circles represent a single value shared by factors
  - Squares assert attached variables sum to 0 mod 2
- Iterative decoding on the binary erasure channel (BEC)
  - Messages passed in phases: bit-to-check and check-to-bit
  - Each output message depends on other input messages
  - Each message is either the correct value or an erasure
- Message passing rules for the BEC
  - Bits pass an erasure only if all other inputs are erased
  - Checks pass the correct value only if all other inputs are correct

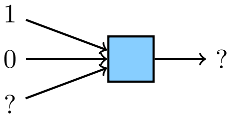




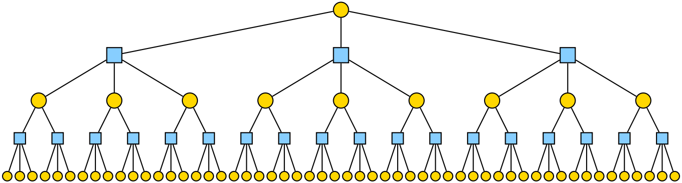
Simple Message-Passing Decoding for the BEC

- Constraint nodes define the valid patterns
  - Circles represent a single value shared by factors
  - Squares assert attached variables sum to 0 mod 2
- Iterative decoding on the binary erasure channel (BEC)
  - Messages passed in phases: bit-to-check and check-to-bit
  - Each output message depends on other input messages
  - Each message is either the correct value or an erasure
- Message passing rules for the BEC
  - Bits pass an erasure only if all other inputs are erased
  - Checks pass the correct value only if all other inputs are correct





Computation Graph and Density Evolution



$$\tilde{x}_3 = \varepsilon y_2^3$$

$$y_2 = 1-(1-x_2)^3$$

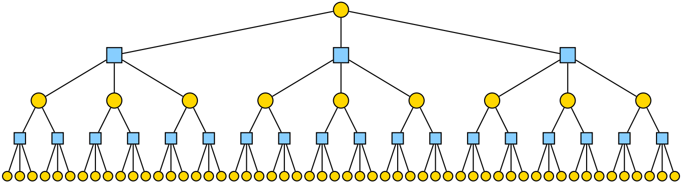
$$x_2 = \varepsilon y_1^2$$

$$y_1 = 1-(1-x_1)^3$$

$$x_1 = \varepsilon$$

- Computation graph for a (3,4)-regular LDPC code
  - Illustrates decoding from the perspective of a single bit-node
  - For long random LDPC codes, the graph is typically a tree
  - Allows density evolution to track message erasure probability

Computation Graph and Density Evolution



$\tilde{x}_3 = \varepsilon y_2^3$

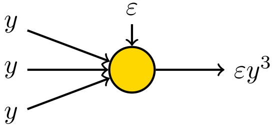
$y_2 = 1-(1-x_2)^3$

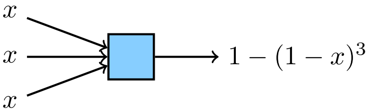
$x_2 = \varepsilon y_1^2$

$y_1 = 1-(1-x_1)^3$

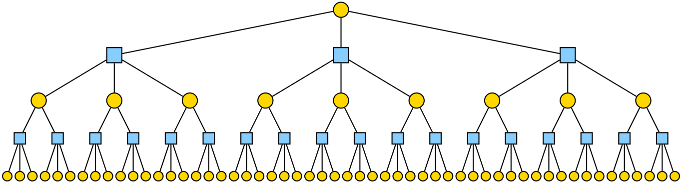
$x_1 = \varepsilon$

- Computation graph for a (3,4)-regular LDPC code
  - Illustrates decoding from the perspective of a single bit-node
  - For long random LDPC codes, the graph is typically a tree
  - Allows density evolution to track message erasure probability
  - If x/y are erasure prob. of bit/check output messages, then





Computation Graph and Density Evolution



$\tilde{x}_3 = \varepsilon y_2^3$

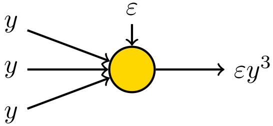
$y_2 = 1 - (1 - x_2)^3$

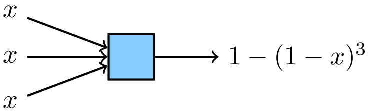
$x_2 = \varepsilon y_1^2$

$y_1 = 1 - (1 - x_1)^3$

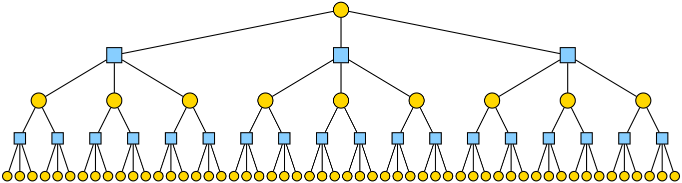
$x_1 = 0.600$

- Computation graph for a (3,4)-regular LDPC code
  - Illustrates decoding from the perspective of a single bit-node
  - For long random LDPC codes, the graph is typically a tree
  - Allows density evolution to track message erasure probability
  - If x/y are erasure prob. of bit/check output messages, then





Computation Graph and Density Evolution



$\tilde{x}_3 = \varepsilon y_2^3$

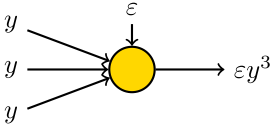
$y_2 = 1-(1-x_2)^3$

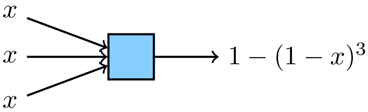
$x_2 = \varepsilon y_1^2$

$y_1 = 0.936$

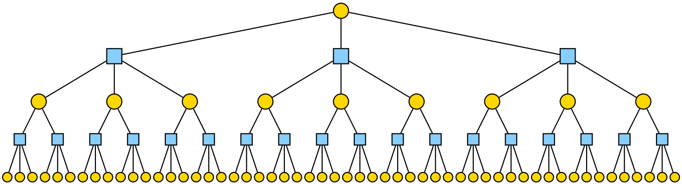
$x_1 = 0.600$

- Computation graph for a (3,4)-regular LDPC code
  - Illustrates decoding from the perspective of a single bit-node
  - For long random LDPC codes, the graph is typically a tree
  - Allows density evolution to track message erasure probability
  - If x/y are erasure prob. of bit/check output messages, then





Computation Graph and Density Evolution



$\tilde{x}_3 = \varepsilon y_2^3$

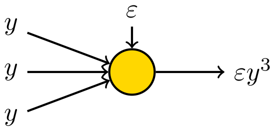
$y_2 = 1-(1-x_2)^3$

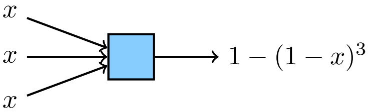
$x_2 = 0.526$

$y_1 = 0.936$

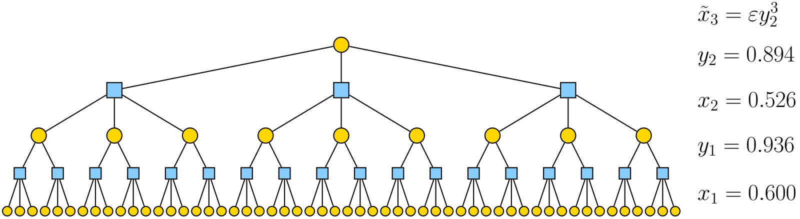
$x_1 = 0.600$

- Computation graph for a (3,4)-regular LDPC code
  - Illustrates decoding from the perspective of a single bit-node
  - For long random LDPC codes, the graph is typically a tree
  - Allows density evolution to track message erasure probability
  - If x/y are erasure prob. of bit/check output messages, then

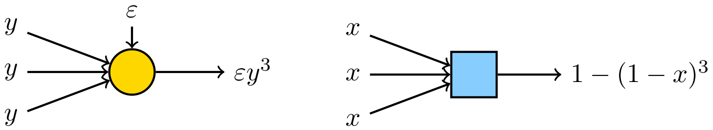




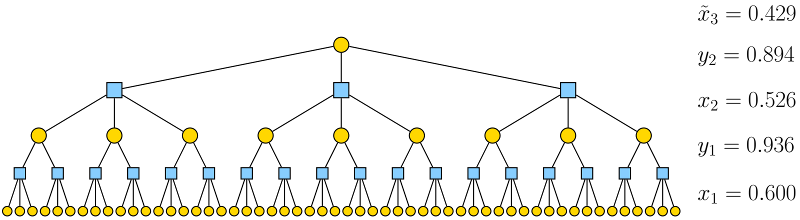
Computation Graph and Density Evolution



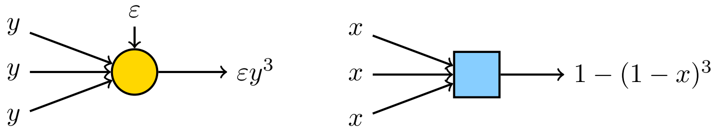
- Computation graph for a (3,4)-regular LDPC code
  - Illustrates decoding from the perspective of a single bit-node
  - For long random LDPC codes, the graph is typically a tree
  - Allows density evolution to track message erasure probability
  - If x/y are erasure prob. of bit/check output messages, then



Computation Graph and Density Evolution

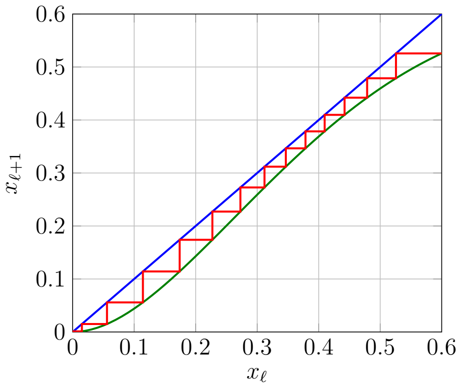


- Computation graph for a (3,4)-regular LDPC code
  - Illustrates decoding from the perspective of a single bit-node
  - For long random LDPC codes, the graph is typically a tree
  - Allows density evolution to track message erasure probability
  - If x/y are erasure prob. of bit/check output messages, then



Density Evolution (DE) for LDPC Codes

(3,4) LDPC Code with $\varepsilon = 0.6$



Density evolution for a $(3,4)$-regular LDPC code:

$$x_{\ell+1} = \varepsilon \left(1 - (1 - x_\ell)^3\right)^2$$

Decoding Thresholds:

$$\varepsilon^{\mathrm{BP}} \approx 0.647$$
$$\varepsilon^{\mathrm{MAP}} \approx 0.746$$
$$\varepsilon^{\mathrm{Sh}} = 0.750$$

- Binary erasure channel (BEC) with erasure prob. ε
- DE tracks bit-to-check msg erasure rate x_ℓ after ℓ iterations
- Defines noise threshold $\varepsilon^{\mathrm{BP}}$ for the large system limit
  - Easily computed numerically for given code ensemble

EXtrinsic Information Transfer (EXIT) Curves

- Introduced by ten Brink in 1999 to understand iterative decoding
  - For the BEC, the MAP EXIT curve is

$$h^{\mathrm{MAP}}(\varepsilon) \triangleq \frac{1}{n}\sum_{i=1}^{n} H(X_i|\underline{Y}_{\sim i}(\varepsilon))$$

EXtrinsic Information Transfer (EXIT) Curves

- Introduced by ten Brink in 1999 to understand iterative decoding
  - For the BEC, the MAP EXIT curve is

$$h^{\mathrm{MAP}}(\varepsilon) \triangleq \frac{1}{n} \sum_{i=1}^{n} H(X_i | \underline{Y}_{\sim i}(\varepsilon))$$

  - EXIT Area Theorem [ABK04]

$$\frac{1}{n} H(\underline{X} | \underline{Y}(\varepsilon)) = \int_0^{\varepsilon} h^{\mathrm{MAP}}(\delta) \mathrm{d}\delta$$

EXtrinsic Information Transfer (EXIT) Curves

- Introduced by ten Brink in 1999 to understand iterative decoding
  - For the BEC, the MAP EXIT curve is

$$h^{\mathrm{MAP}}(\varepsilon) \triangleq \frac{1}{n}\sum_{i=1}^{n} H(X_i|\underline{Y}_{\sim i}(\varepsilon))$$

  - EXIT Area Theorem [ABK04]

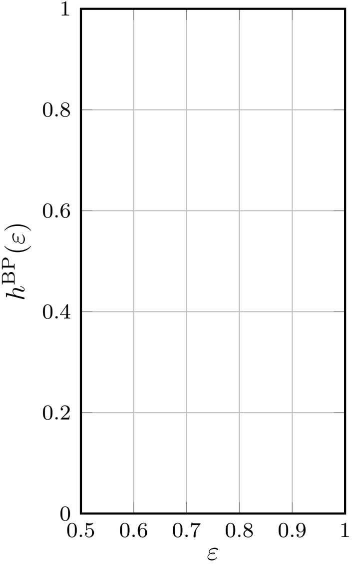
$$\frac{1}{n}H(\underline{X}|\underline{Y}(\varepsilon)) = \int_0^{\varepsilon} h^{\mathrm{MAP}}(\delta)\mathrm{d}\delta$$

  - BP EXIT curve

$$h^{\mathrm{BP}}(\varepsilon) \triangleq \frac{1}{n}\sum_{i=1}^{n} H\big(X_i|\Phi_i^{\mathrm{BP}}(\underline{Y}_{\sim i}(\varepsilon))\big)$$

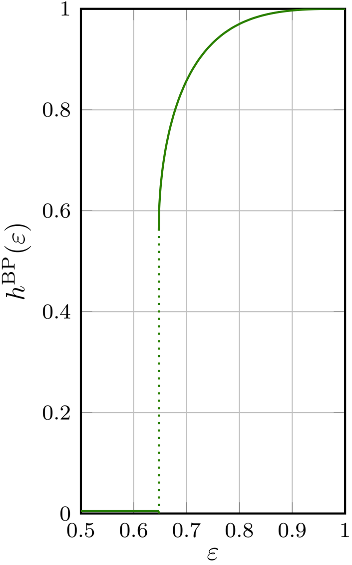
    - where $\Phi_i^{\mathrm{BP}}(Z)$ is the BP estimate of X_i given Z
    - Data processing inequality: $h^{\mathrm{BP}}(\varepsilon) \geq h^{\mathrm{MAP}}(\varepsilon)$

EXtrinsic Information Transfer (EXIT) Curves



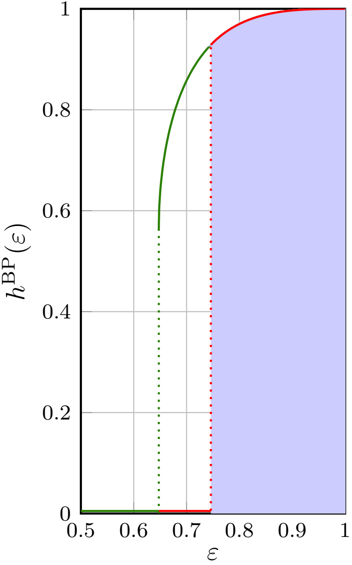
- (3,4)-regular LDPC code
  - Codeword $(X_1, \ldots, X_n)$
  - Received $(Y_1, \ldots, Y_n)$

EXtrinsic Information Transfer (EXIT) Curves



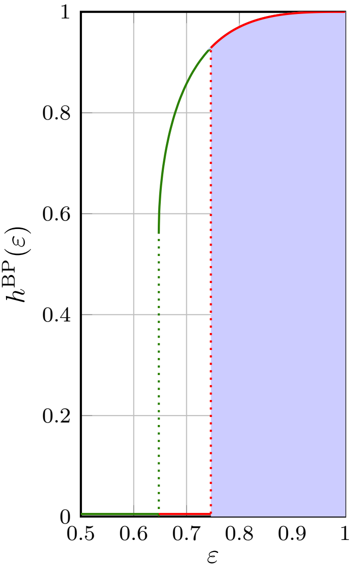
- (3,4)-regular LDPC code
  - Codeword $(X_1, \ldots, X_n)$
  - Received $(Y_1, \ldots, Y_n)$
- BP EXIT curve via DE
  - This code: $h^{\mathrm{BP}}(\varepsilon) = (x_\infty(\varepsilon))^3$
  - 0 below BP threshold 0.647

EXtrinsic Information Transfer (EXIT) Curves



- (3,4)-regular LDPC code
  - Codeword $(X_1, \ldots, X_n)$
  - Received $(Y_1, \ldots, Y_n)$
- BP EXIT curve via DE
  - This code: $h^{\mathrm{BP}}(\varepsilon) = (x_\infty(\varepsilon))^3$
  - 0 below BP threshold 0.647
- MAP EXIT curve is extrinsic entropy $H(X_i|\underline{Y}_{\sim i})$ vs. channel ε
  - 0 below MAP threshold 0.746
  - Area under curve equals rate R
  - Upper bounded by BP EXIT

EXtrinsic Information Transfer (EXIT) Curves



- (3,4)-regular LDPC code
  - Codeword $(X_1, \ldots, X_n)$
  - Received $(Y_1, \ldots, Y_n)$
- BP EXIT curve via DE
  - This code: $h^{\mathrm{BP}}(\varepsilon) = (x_\infty(\varepsilon))^3$
  - 0 below BP threshold 0.647
- MAP EXIT curve is extrinsic entropy $H(X_i|\underline{Y}_{\sim i})$ vs. channel ε
  - 0 below MAP threshold 0.746
  - Area under curve equals rate R
  - Upper bounded by BP EXIT
- MAP threshold upper bound $\overline{\varepsilon}^{\mathrm{MAP}}$
  - ε s.t. area under BP EXIT is R

Outline

Introduction

Factor Graphs

Message Passing

Applications of Factor Graphs

Applications of EXIT Curves

Spatially-Coupled Factor Graphs

Universality for Multiuser Scenarios

Abstract Formulation of Threshold Saturation

Properties of the MAP EXIT Curve

- For linear codes, the recovery of X_i from $\underline{Y} = \underline{y}$
  - is independent of the transmitted codeword $\underline{X}$
  - only depends on erasure indicator $z_i = \mathbf{1}_{\{?\}}(y_i)$
  - is determined by whether $H(X_i|\underline{Z} = \underline{z})$ is 0 or 1

Properties of the MAP EXIT Curve

- For linear codes, the recovery of X_i from $\underline{Y} = \underline{y}$
  - is independent of the transmitted codeword $\underline{X}$
  - only depends on erasure indicator $z_i = \mathbf{1}_{\{?\}}(y_i)$
  - is determined by whether $H(X_i|\underline{Z} = \underline{z})$ is 0 or 1
- The MAP bit-erasure rate $P_b(\varepsilon)$ satisfies

$$P_b(\varepsilon) = \mathbb{P}(Y_i = ?)H(X_i|\underline{Y}, Y_i = ?) = \varepsilon h^{\mathrm{MAP}}(\varepsilon)$$

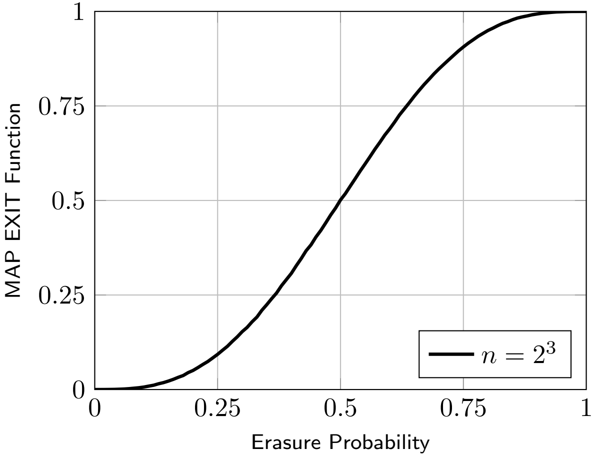
Properties of the MAP EXIT Curve

- For linear codes, the recovery of X_i from $\underline{Y} = \underline{y}$
  - is independent of the transmitted codeword $\underline{X}$
  - only depends on erasure indicator $z_i = \mathbf{1}_{\{?\}}(y_i)$
  - is determined by whether $H(X_i|\underline{Z} = \underline{z})$ is 0 or 1
- The MAP bit-erasure rate $P_b(\varepsilon)$ satisfies

$$P_b(\varepsilon) = \mathbb{P}(Y_i = ?)H(X_i|\underline{Y}, Y_i = ?) = \varepsilon h^{\mathrm{MAP}}(\varepsilon)$$

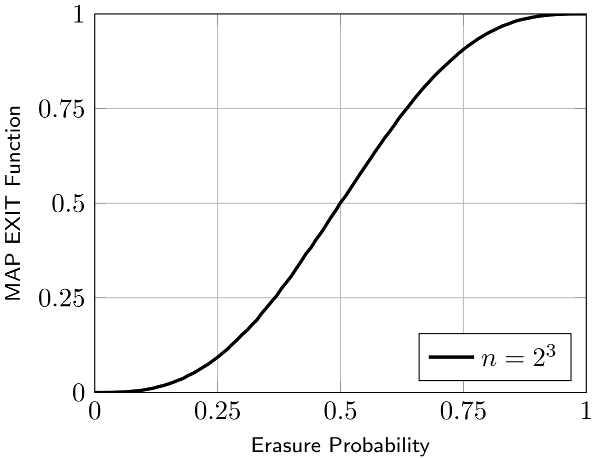
- A sequence of rate-R codes achieves capacity iff
  - $P_b(\varepsilon) \to 0$ for all $\varepsilon < 1 - R$
  - $h^{\mathrm{MAP}}(\varepsilon) \to 0$ for all $\varepsilon < 1 - R$
  - $h^{\mathrm{MAP}}(\varepsilon)$ transitions sharply from 0 to 1

The MAP EXIT Curve of a Capacity-Achieving Code



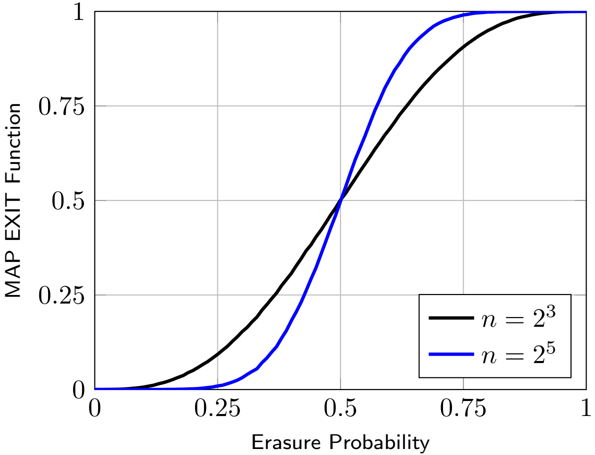
- For $\delta > 0$, transition width is ε-range over which $\delta \le h^{\mathrm{MAP}}(\varepsilon) \le 1 - \delta$

The MAP EXIT Curve of a Capacity-Achieving Code



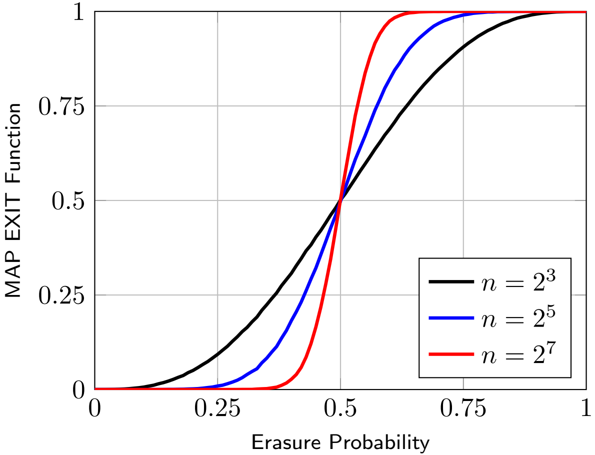
- For $\delta > 0$, transition width is ε-range over which $\delta \leq h^{\mathrm{MAP}}(\varepsilon) \leq 1 - \delta$
- Area Theorem implies sharp transition iff capacity achieving

The MAP EXIT Curve of a Capacity-Achieving Code



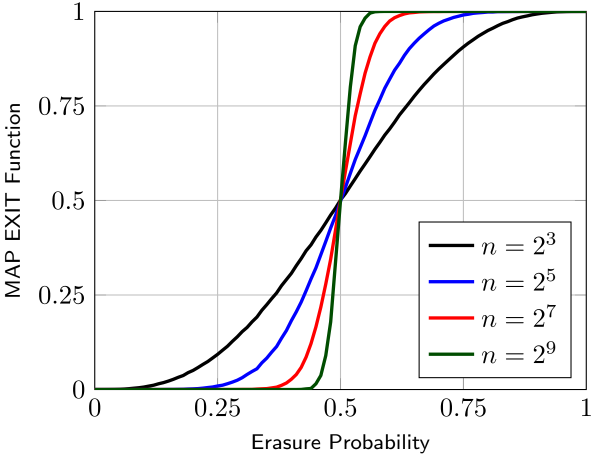
- For $\delta > 0$, transition width is ε-range over which $\delta \leq h^{\mathrm{MAP}}(\varepsilon) \leq 1 - \delta$
- Area Theorem implies sharp transition iff capacity achieving

The MAP EXIT Curve of a Capacity-Achieving Code



- For $\delta > 0$, transition width is ε-range over which $\delta \leq h^{\mathrm{MAP}}(\varepsilon) \leq 1 - \delta$
- Area Theorem implies sharp transition iff capacity achieving

The MAP EXIT Curve of a Capacity-Achieving Code



- For $\delta > 0$, transition width is ε-range over which $\delta \leq h^{\mathrm{MAP}}(\varepsilon) \leq 1 - \delta$
- Area Theorem implies sharp transition iff capacity achieving

EXIT Curves and Sharp Transitions

- Consider any monotone boolean function $f\colon \{0,1\}^{n-1} \to \{0,1\}$

EXIT Curves and Sharp Transitions

- Consider any monotone boolean function $f\colon \{0,1\}^{n-1} \to \{0,1\}$
  - Define its symmetry group $\mathcal{G}$ to be

$$\mathcal{G} = \left\{ \pi \in S_{n-1} \,\middle|\, f(\pi(\underline{z})) = f(\underline{z}) \,\forall \underline{z} \in \{0,1\}^{n-1} \right\}$$

EXIT Curves and Sharp Transitions

- Consider any monotone boolean function $f\colon \{0,1\}^{n-1} \to \{0,1\}$
  - Define its symmetry group $\mathcal{G}$ to be

$$\mathcal{G} = \left\{ \pi \in S_{n-1} \,|\, f(\pi(\underline{z})) = f(\underline{z})\, \forall \underline{z} \in \{0,1\}^{n-1} \right\}$$

  - Let $Z_i \in \{0,1\}$ be i.i.d. with $\mathbb{P}(Z_i = 1) = \varepsilon$ and define

$$h(\varepsilon) \triangleq \mathbb{E}\left[f(Z_1, \ldots, Z_{n-1}) \right]$$

EXIT Curves and Sharp Transitions

- Consider any monotone boolean function $f\colon \{0,1\}^{n-1} \to \{0,1\}$
  - Define its symmetry group $\mathcal{G}$ to be

$$\mathcal{G} = \left\{ \pi \in S_{n-1} \,|\, f(\pi(\underline{z})) = f(\underline{z}) \, \forall \underline{z} \in \{0,1\}^{n-1} \right\}$$

  - Let $Z_i \in \{0,1\}$ be i.i.d. with $\mathbb{P}(Z_i = 1) = \varepsilon$ and define

$$h(\varepsilon) \triangleq \mathbb{E}\left[f(Z_1, \ldots, Z_{n-1}) \right]$$

  - If $\mathcal{G}$ is transitive, then $h(\varepsilon)$ has transition width $O\left(\frac{1}{\ln n}\right)^*$

$$\forall i, j \in \{1, 2, \ldots, n-1\}, \exists \pi \in \mathcal{G} \text{ s.t. } \pi(i) = j$$

* Friedgut-Kalai'96: "Every monotone graph property has a sharp threshold"

EXIT Curves and Sharp Transitions

- Consider any monotone boolean function $f\colon \{0,1\}^{n-1} \to \{0,1\}$
  - Define its symmetry group $\mathcal{G}$ to be

$$\mathcal{G} = \left\{\pi \in S_{n-1} \,|\, f(\pi(\underline{z})) = f(\underline{z}) \,\forall \underline{z} \in \{0,1\}^{n-1}\right\}$$

  - Let $Z_i \in \{0,1\}$ be i.i.d. with $\mathbb{P}(Z_i = 1) = \varepsilon$ and define

$$h(\varepsilon) \triangleq \mathbb{E}\left[f(Z_1, \ldots, Z_{n-1})\right]$$

  - If $\mathcal{G}$ is transitive, then $h(\varepsilon)$ has transition width $O\left(\frac{1}{\ln n}\right)^*$

$$\forall i, j \in \{1, 2, \ldots, n-1\}, \exists \pi \in \mathcal{G} \text{ s.t. } \pi(i) = j$$

- When do EXIT curves have a sharp transition? [KKMPSU15]
  - If the code's permutation group is doubly transitive!
  - For example, Reed-Muller and prim. narrow-sense BCH codes

---

* Friedgut-Kalai'96: "Every monotone graph property has a sharp threshold"

Summary and Open Problems

- Gallager's 1960 thesis already contains most of the tools necessary to achieve capacity in practice
  - But, he focuses mainly on the BSC
  - Had he attacked the BEC, practical capacity-achieving codes might have been introduced years earlier

Summary and Open Problems

- Gallager's 1960 thesis already contains most of the tools necessary to achieve capacity in practice
  - But, he focuses mainly on the BSC
  - Had he attacked the BEC, practical capacity-achieving codes might have been introduced years earlier
- The first deterministic sequence of capacity-achieving binary codes for the BEC (under MAP decoding) was defined in 1954!
  - Sequences of Reed-Muller codes achieve capacity on the BEC
  - But, we didn't know this until 2015!

Summary and Open Problems

- Gallager's 1960 thesis already contains most of the tools necessary to achieve capacity in practice
  - But, he focuses mainly on the BSC
  - Had he attacked the BEC, practical capacity-achieving codes might have been introduced years earlier
- The first deterministic sequence of capacity-achieving binary codes for the BEC (under MAP decoding) was defined in 1954!
  - Sequences of Reed-Muller codes achieve capacity on the BEC
  - But, we didn't know this until 2015!
- Open problems
  - Generalize the Reed-Muller result to have weaker conditions and/or apply to more general channels/problems
  - Find a purely information-theoretic proof of the Reed-Muller result for the BEC

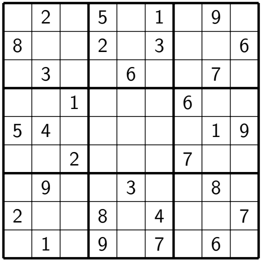
Outline

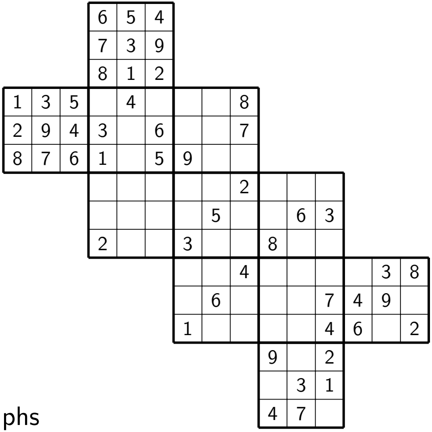
What is Spatial Coupling?

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

			6	5	4									
			7	3	9									
			8	1	2									
1	3	5		4				8						
2	9	4	3		6			7						
8	7	6	1		5	9								
								2						
							5			6	3			
			2			3			8					
								4					3	8
							6				7	4	9	
						1					4	6		2
									9		2			
										3	1			
									4	7				

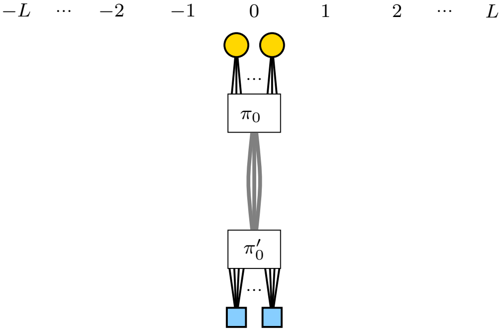
What is Spatial Coupling?



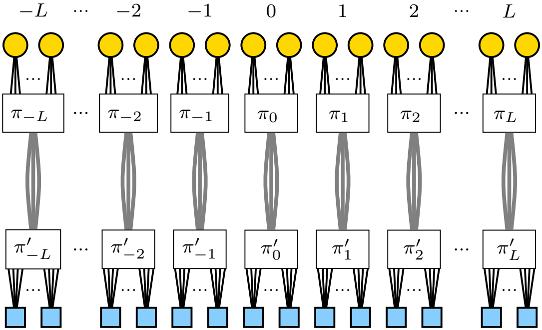


- Spatially-Coupled Factor Graphs
  - Variable nodes have a natural global orientation
  - Boundaries help variables to be recovered in an ordered fashion

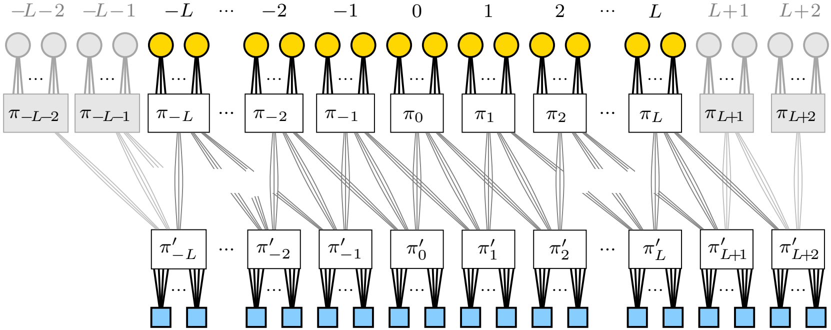
Spatially-Coupled LDPC Codes: (l, r, L, w) Ensemble



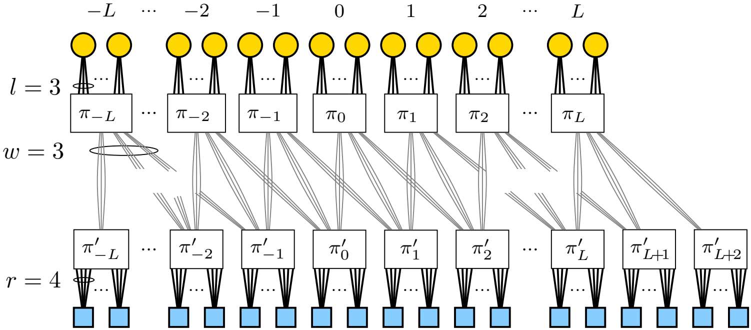
Spatially-Coupled LDPC Codes: (l, r, L, w) Ensemble



Spatially-Coupled LDPC Codes: (l, r, L, w) Ensemble



Spatially-Coupled LDPC Codes: (l, r, L, w) Ensemble



- Historical Notes
  - LDPC convolutional codes introduced by FZ in 1999
  - Shown to have near optimal noise thresholds by LSZC in 2005
  - (l, r, L, w) ensemble proven to achieve capacity by KRU in 2011

The LDPCC Gang

5274 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 56, NO. 10, OCTOBER 2010

Iterative Decoding Threshold Analysis for LDPC Convolutional Codes

Michael Lentmaier, *Member, IEEE*, Arvind Sridharan, *Member, IEEE*, Daniel J. Costello, Jr., *Life Fellow, IEEE*, and Kamil Sh. Zigangirov, *Fellow, IEEE*

The Spatial Coupling KRU

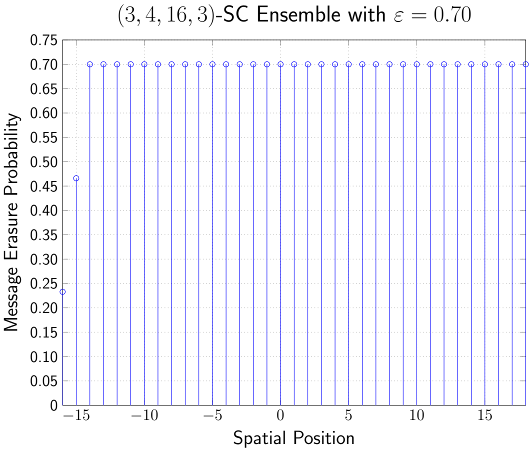
IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 57, NO. 2, FEBRUARY 2011 803

Threshold Saturation via Spatial Coupling: Why Convolutional LDPC Ensembles Perform So Well over the BEC

Shrinivas Kudekar, *Member, IEEE*, Thomas J. Richardson, *Fellow, IEEE*, and Rüdiger L. Urbanke

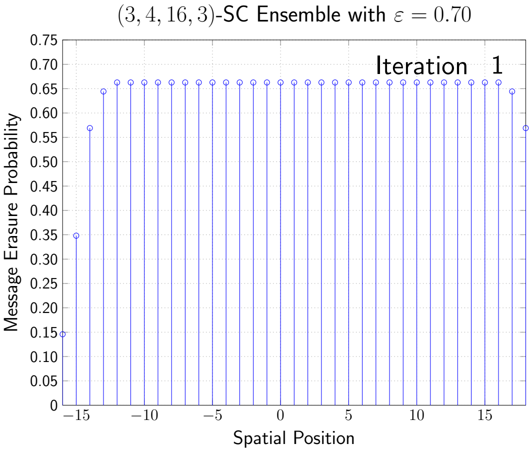
  

Density Evolution for the (l, r, L, w)-SC LDPC Ensemble



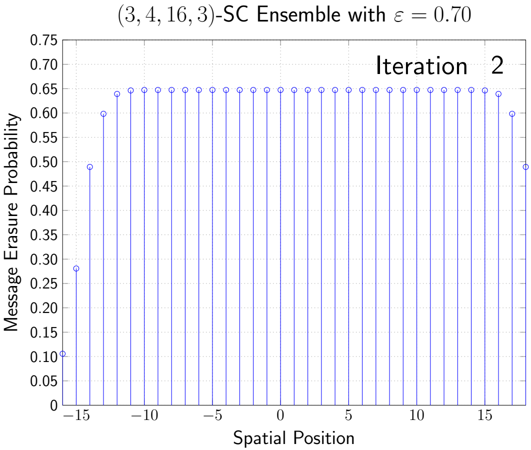
$$x_i^{(\ell+1)} = \frac{1}{w} \sum_{k=0}^{w-1} \varepsilon \left(\frac{1}{w} \sum_{j=0}^{w-1} \left(1 - (1 - x_{i+j-k}^{(\ell)})^{r-1}\right) \right)^{l-1} \mathbf{1}_{[-L, L+w-1]}(i-k)$$

Density Evolution for the (l, r, L, w)-SC LDPC Ensemble



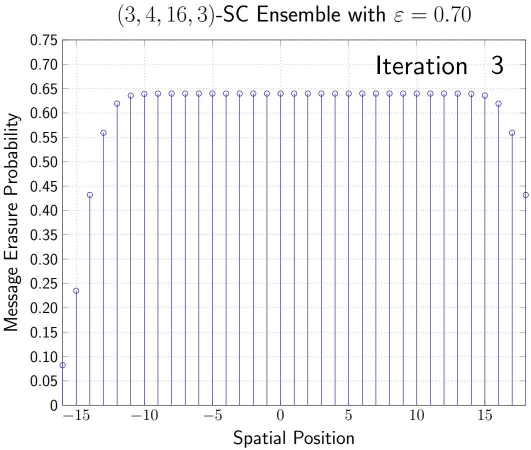
$$x_i^{(\ell+1)} = \frac{1}{w}\sum_{k=0}^{w-1} \varepsilon \left(\frac{1}{w}\sum_{j=0}^{w-1} \left(1 - (1 - x_{i+j-k}^{(\ell)})^{r-1}\right) \right)^{l-1} \mathbf{1}_{[-L,L+w-1]}(i-k)$$

Density Evolution for the (l, r, L, w)-SC LDPC Ensemble



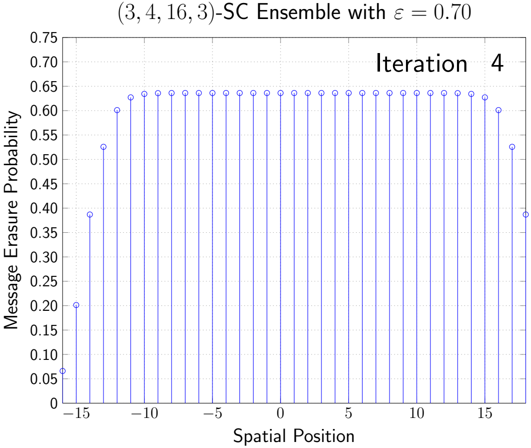
$$x_i^{(\ell+1)} = \frac{1}{w} \sum_{k=0}^{w-1} \varepsilon \left(\frac{1}{w} \sum_{j=0}^{w-1} \left(1 - (1 - x_{i+j-k}^{(\ell)})^{r-1} \right) \right)^{l-1} \mathbf{1}_{[-L, L+w-1]}(i-k)$$

Density Evolution for the (l, r, L, w)-SC LDPC Ensemble



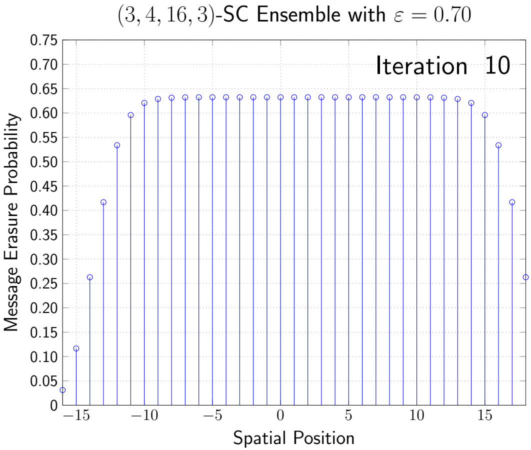
$$x_i^{(\ell+1)} = \frac{1}{w}\sum_{k=0}^{w-1} \varepsilon \left(\frac{1}{w}\sum_{j=0}^{w-1} \left(1 - (1 - x_{i+j-k}^{(\ell)})^{r-1}\right) \right)^{l-1} \mathbf{1}_{[-L, L+w-1]}(i-k)$$

Density Evolution for the (l, r, L, w)-SC LDPC Ensemble



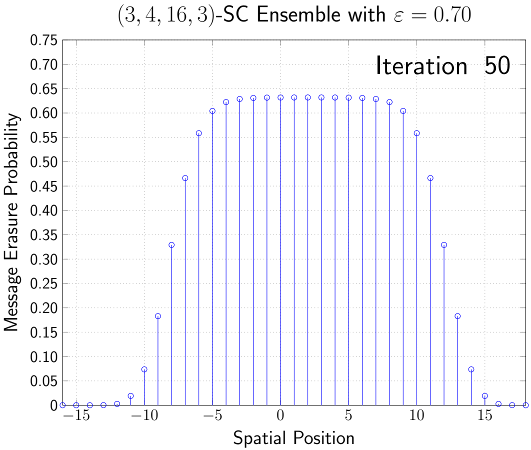
$$x_i^{(\ell+1)} = \frac{1}{w} \sum_{k=0}^{w-1} \varepsilon \left(\frac{1}{w} \sum_{j=0}^{w-1} \left(1 - (1 - x_{i+j-k}^{(\ell)})^{r-1} \right) \right)^{l-1} \mathbf{1}_{[-L, L+w-1]}(i-k)$$

Density Evolution for the (l, r, L, w)-SC LDPC Ensemble



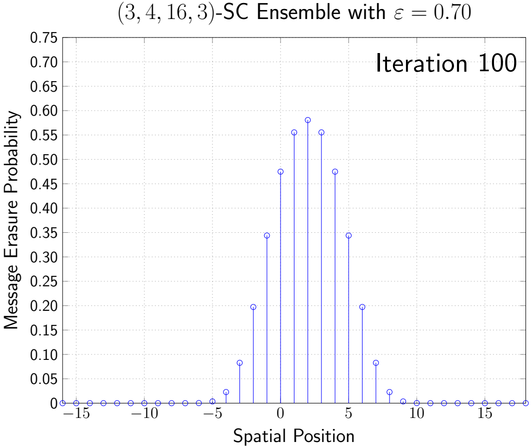
$$x_i^{(\ell+1)} = \frac{1}{w} \sum_{k=0}^{w-1} \varepsilon \left(\frac{1}{w} \sum_{j=0}^{w-1} \left(1 - (1 - x_{i+j-k}^{(\ell)})^{r-1} \right) \right)^{l-1} \mathbf{1}_{[-L, L+w-1]}(i-k)$$

Density Evolution for the (l, r, L, w)-SC LDPC Ensemble



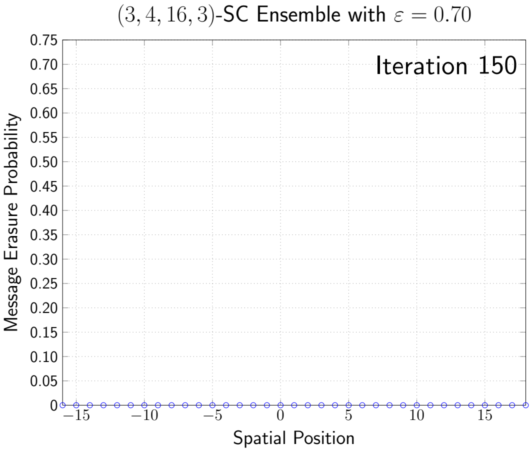
$$x_i^{(\ell+1)} = \frac{1}{w} \sum_{k=0}^{w-1} \varepsilon \left(\frac{1}{w} \sum_{j=0}^{w-1} \left(1 - (1 - x_{i+j-k}^{(\ell)})^{r-1}\right) \right)^{l-1} \mathbf{1}_{[-L, L+w-1]}(i-k)$$

Density Evolution for the (l, r, L, w)-SC LDPC Ensemble



$$x_i^{(\ell+1)} = \frac{1}{w} \sum_{k=0}^{w-1} \varepsilon \left(\frac{1}{w} \sum_{j=0}^{w-1} \left(1 - (1 - x_{i+j-k}^{(\ell)})^{r-1} \right) \right)^{l-1} \mathbf{1}_{[-L, L+w-1]}(i-k)$$

Density Evolution for the (l, r, L, w)-SC LDPC Ensemble



$$x_i^{(\ell+1)} = \frac{1}{w}\sum_{k=0}^{w-1}\varepsilon\left(\frac{1}{w}\sum_{j=0}^{w-1}\left(1-(1-x_{i+j-k}^{(\ell)})^{r-1}\right)\right)^{l-1}\mathbf{1}_{[-L,L+w-1]}(i-k)$$

Properties of Threshold Saturation

l	r	$\varepsilon^{\mathrm{BP}}$	$\varepsilon^{\mathrm{MAP}}$
3	6	0.4294	0.4882
4	8	0.3834	0.4977
5	10	0.3416	0.4995
6	12	0.3075	0.4999
7	14	0.2798	0.5000

- Spatial coupling achieves the MAP threshold as $w \to \infty$
  - BP threshold typically decreases after $l = 3$
  - MAP threshold is increasing in l, r for fixed rate
- Benefits and Drawbacks
  - For fixed L, minimum distance grows linearly with block length
  - Rate loss of $O(w/L)$ is a big obstacle in practice

Threshold Saturation via Spatial Coupling

- **General Phenomenon** (observed by Kudekar, Richardson, Urbanke)
  - BP threshold of the spatially-coupled system converges to the MAP threshold of the uncoupled system
  - Can be proven rigorously in many cases!

Threshold Saturation via Spatial Coupling

- **General Phenomenon** (observed by Kudekar, Richardson, Urbanke)
  - BP threshold of the spatially-coupled system converges to the MAP threshold of the uncoupled system
  - Can be proven rigorously in many cases!
- Connection to statistical physics
  - Factor graph defines system of coupled particles
  - Valid sequences are ordered crystalline structures

Threshold Saturation via Spatial Coupling

- **General Phenomenon** (observed by Kudekar, Richardson, Urbanke)
  - BP threshold of the spatially-coupled system converges to the MAP threshold of the uncoupled system
  - Can be proven rigorously in many cases!
- Connection to statistical physics
  - Factor graph defines system of coupled particles
  - Valid sequences are ordered crystalline structures
- Between BP and MAP threshold, system acts as supercooled liquid
  - Correct answer (crystalline state) has minimum energy
  - Crystallization (i.e., decoding) does not occur without a seed
  - Ex.: ice melts at $0\,^{\circ}$C but freezing w/o a seed requires $-48.3\,^{\circ}$C

http://www.youtube.com/watch?v=Xe8vJrIvDQM

Why is Spatial Coupling Interesting?

- Breakthroughs: first practical constructions of
  - universal codes for binary-input memoryless channels [KRU12]
  - information-theoretically optimal compressive sensing [DJM11]
  - universal codes for Slepian-Wolf and MAC problems [YJNP11]
  - codes $\to$ capacity with iterative hard-decision decoding [JNP12]
  - codes $\to$ rate-distortion limit with iterative decoding [AMUV12]

Why is Spatial Coupling Interesting?

- Breakthroughs: first practical constructions of
  - universal codes for binary-input memoryless channels [KRU12]
  - information-theoretically optimal compressive sensing [DJM11]
  - universal codes for Slepian-Wolf and MAC problems [YJNP11]
  - codes $\rightarrow$ capacity with iterative hard-decision decoding [JNP12]
  - codes $\rightarrow$ rate-distortion limit with iterative decoding [AMUV12]
- It allows rigorous proof in many cases
  - Original proofs [KRU11/12] quite specific to LDPC codes
  - Our proof for increasing scalar/vector recursions [YJNP12/13]

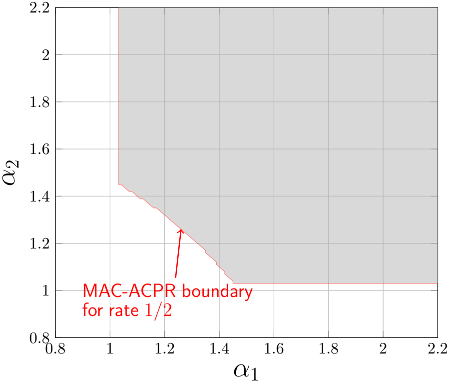
Why is Spatial Coupling Interesting?

- Breakthroughs: first practical constructions of
  - universal codes for binary-input memoryless channels [KRU12]
  - information-theoretically optimal compressive sensing [DJM11]
  - universal codes for Slepian-Wolf and MAC problems [YJNP11]
  - codes $\rightarrow$ capacity with iterative hard-decision decoding [JNP12]
  - codes $\rightarrow$ rate-distortion limit with iterative decoding [AMUV12]
- It allows rigorous proof in many cases
  - Original proofs [KRU11/12] quite specific to LDPC codes
  - Our proof for increasing scalar/vector recursions [YJNP12/13]
- Spatial coupling as a proof technique [GMU13]
  - For a large random factor graph, construct a coupled version
  - Use DE to analyze BP decoding of coupled system
  - Compare uncoupled MAP with coupled BP via interpolation

Outline

Universality over Unknown Parameters

- The Achievable Channel Parameter Region (ACPR)
  - For a sequence of coding schemes involving one or more parameters, the parameter region where decoding succeeds in the limit
  - In contrast, a capacity region is a rate region for fixed channels



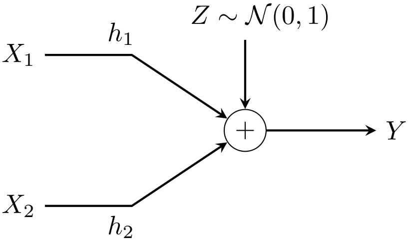
Universality over Unknown Parameters

- The Achievable Channel Parameter Region (ACPR)
  - For a sequence of coding schemes involving one or more parameters, the parameter region where decoding succeeds in the limit
  - In contrast, a capacity region is a rate region for fixed channels
- Properties
  - For fixed encoders, the ACPR depends on the decoders
  - For example, one has BP-ACPR $\subseteq$ MAP-ACPR
  - Often, $\exists$ unique maximal ACPR given by information theory

Universality over Unknown Parameters

- The Achievable Channel Parameter Region (ACPR)
  - For a sequence of coding schemes involving one or more parameters, the parameter region where decoding succeeds in the limit
  - In contrast, a capacity region is a rate region for fixed channels
- Properties
  - For fixed encoders, the ACPR depends on the decoders
  - For example, one has BP-ACPR $\subseteq$ MAP-ACPR
  - Often, $\exists$ unique maximal ACPR given by information theory
- Universality
  - A sequence of encoding/decoding schemes is called universal if:
    its ACPR equals the optimal ACPR
  - Channel parameters are assumed unknown at the transmitter
  - At the receiver, the channel parameters are easily estimated

2-User Binary-Input Gaussian Multiple Access Channel



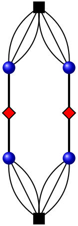
- Fixed noise variance
- Real channel gains h_1 and h_2 not known at transmitter
- Each code has rate R

- MAC-ACPR denotes the information-theoretic optimal region

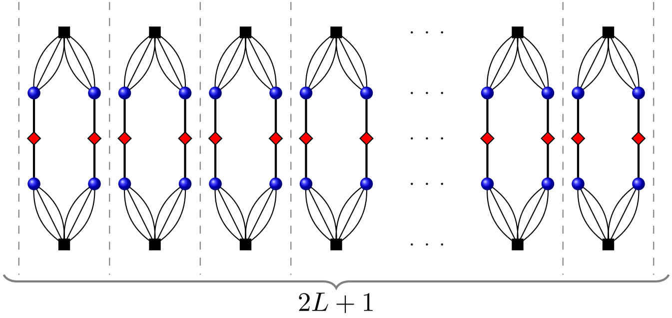
A Little History: SC for Multiple-Access (MAC) Channels

- KK consider a binary-adder erasure channel (ISIT 2011)
  - SC exhibits threshold saturation for the joint decoder
- YNPN consider the Gaussian MAC (ISIT/Allerton 2011)
  - SC exhibits threshold saturation for the joint decoder
  - For channel gains h_1, h_2 unknown at transmitter, SC provides universality
- Others consider CDMA systems without coding
  - TTK show SC improves BP demod of standard CDMA
  - ST prove saturation for a SC protograph-style CDMA

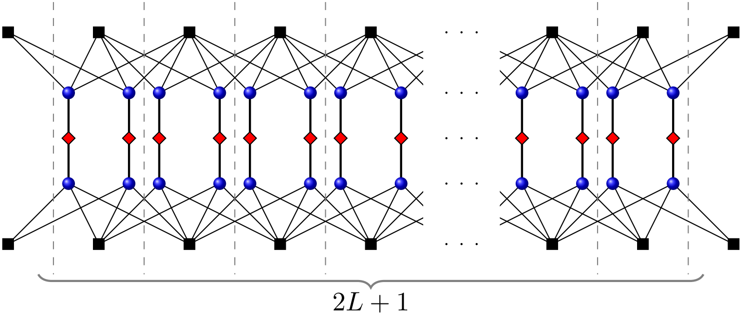
Spatially-Coupled Factor Graph for Joint Decoder



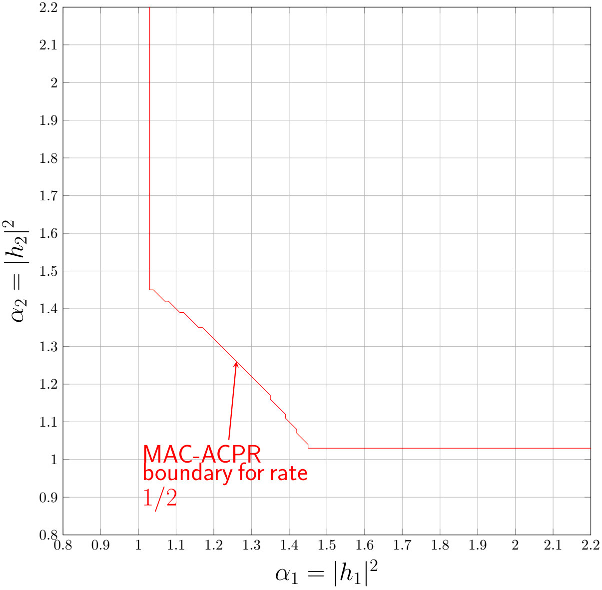
Spatially-Coupled Factor Graph for Joint Decoder



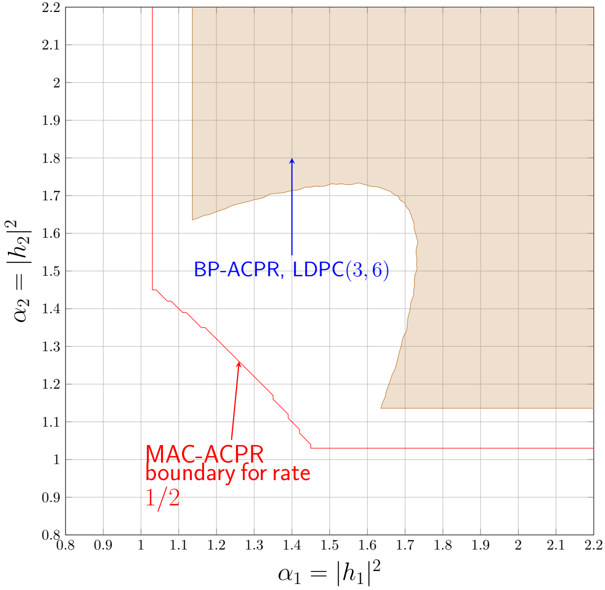
Spatially-Coupled Factor Graph for Joint Decoder



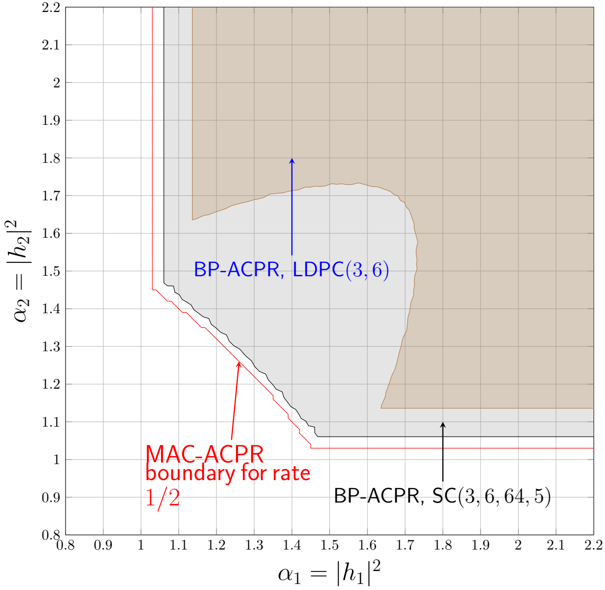
DE Performance of the Joint Decoder



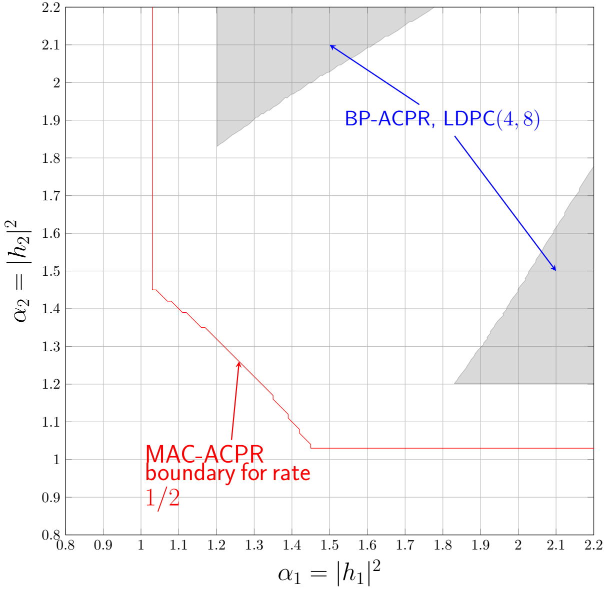
DE Performance of the Joint Decoder



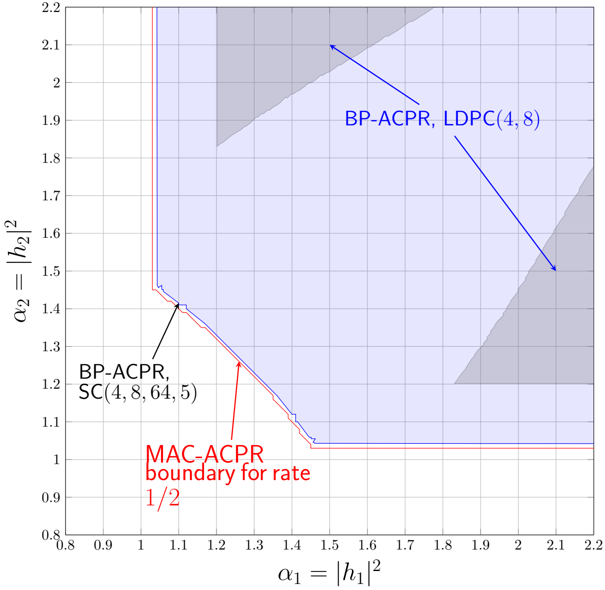
DE Performance of the Joint Decoder



DE Performance of the Joint Decoder



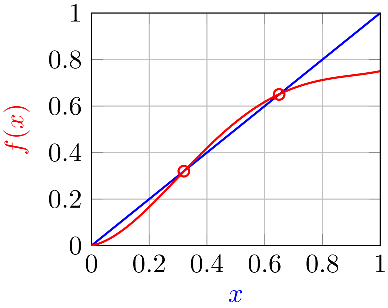
DE Performance of the Joint Decoder



Outline

Single Monotone Recursion

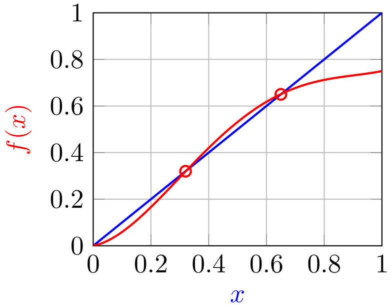
- Smooth increasing $f\colon [0,1] \to [0,1]$



Single Monotone Recursion

- Smooth increasing $f\colon [0,1] \to [0,1]$
- Discrete-time recursion

$$x^{(\ell+1)} = f(x^{(\ell)})$$



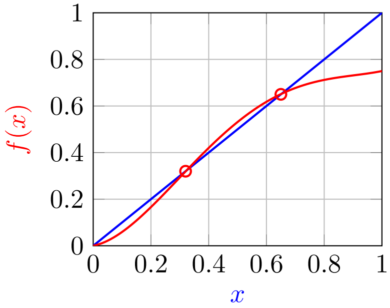
Single Monotone Recursion

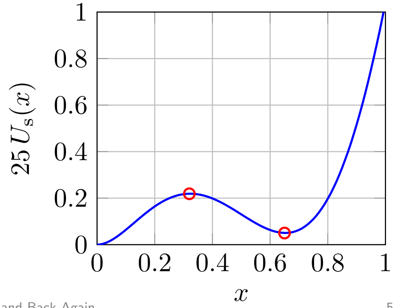
- Smooth increasing $f\colon [0,1] \to [0,1]$
- Discrete-time recursion

$$x^{(\ell+1)} = f(x^{(\ell)})$$

- "Potential energy" $U_{\mathrm{s}}(x)$

$$U_{\mathrm{s}}(x) = \int_0^x \big(z - f(z)\big)\mathrm{d}z = \frac{x^2}{2} - F(x)$$





Single Monotone Recursion

- Smooth increasing $f\colon [0,1] \to [0,1]$
- Discrete-time recursion

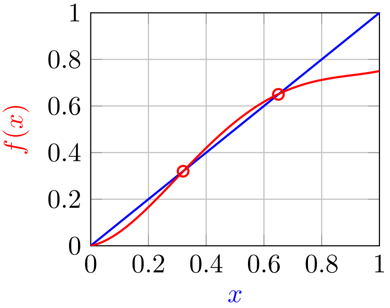
$$x^{(\ell+1)} = f(x^{(\ell)})$$

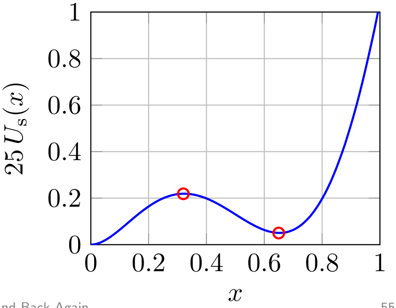
- "Potential energy" $U_{\mathrm{s}}(x)$

$$U_{\mathrm{s}}(x) = \int_0^x \big(z - f(z)\big)\mathrm{d}z = \frac{x^2}{2} - F(x)$$

- Continuous (small step) dynamics

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = f\big(x(t)\big) - x(t) = -\nabla U_{\mathrm{s}}\big(x(t)\big)$$





Single Monotone Recursion

- Smooth increasing $f\colon [0,1] \to [0,1]$
- Discrete-time recursion

$$x^{(\ell+1)} = f(x^{(\ell)})$$

- "Potential energy" $U_{\mathrm{s}}(x)$

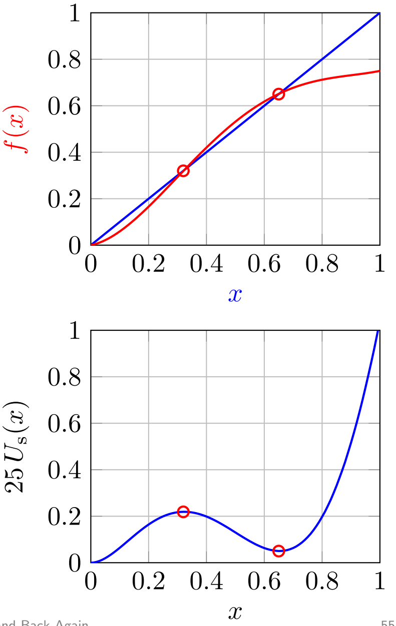
$$U_{\mathrm{s}}(x) = \int_0^x \big(z - f(z)\big)\mathrm{d}z = \frac{x^2}{2} - F(x)$$

- Continuous (small step) dynamics

$$\frac{\mathrm{d}}{\mathrm{d}t} x(t) = f\big(x(t)\big) - x(t) = -\nabla U_{\mathrm{s}}\big(x(t)\big)$$

- Lyapunov stability

$$\frac{\mathrm{d}}{\mathrm{d}t} U_{\mathrm{s}}\big(x(t)\big) = -\big(x(t) - f(x(t))\big)^2$$



Single Monotone Recursion

- Smooth increasing $f\colon [0,1] \to [0,1]$
- Discrete-time recursion

$$x^{(\ell+1)} = f(x^{(\ell)})$$

- "Potential energy" $U_{\mathrm{s}}(x)$

$$U_{\mathrm{s}}(x) = \int_0^x \big(z - f(z)\big)\mathrm{d}z = \frac{x^2}{2} - F(x)$$

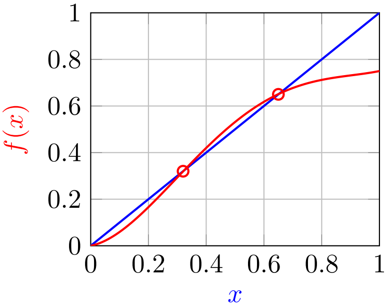
- Continuous (small step) dynamics

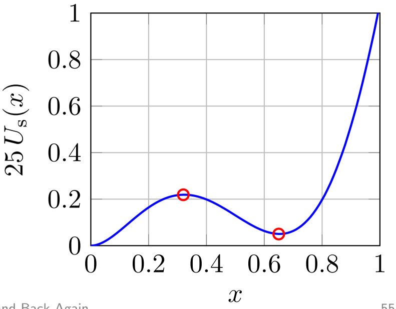
$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = f\big(x(t)\big) - x(t) = -\nabla U_{\mathrm{s}}\big(x(t)\big)$$

- Lyapunov stability

$$\frac{\mathrm{d}}{\mathrm{d}t}U_{\mathrm{s}}\big(x(t)\big) = -\big(x(t) - f\big(x(t)\big)\big)^2$$

Both $\downarrow 0$ iff no fixed points in $(0,1]$





Coupled Monotone Recursion (1)

- Coupled recursion $\underline{x}^{(\ell+1)} = T\underline{x}^{(\ell)}$ with $\underline{x}^{(\ell)} = \left(x_0^{(\ell)}, x_1^{(\ell)}, \ldots\right)$ and

$$T\underline{x} \triangleq A^\top \underline{f}\left(A\underline{x}\right),$$

where $[\underline{f}(\underline{x})]_i = f(x_i)$ and A averages w adjacent values

$$A = \frac{1}{w}\begin{bmatrix} 1 & 1 & \cdots & 1 & 0 & \cdots \\ 0 & 1 & 1 & \ddots & 1 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

  - i.e., avg right w positions, apply f, then avg left w positions

Coupled Monotone Recursion (1)

- Coupled recursion $\underline{x}^{(\ell+1)} = T\underline{x}^{(\ell)}$ with $\underline{x}^{(\ell)} = \left(x_0^{(\ell)}, x_1^{(\ell)}, \ldots\right)$ and

$$T\underline{x} \triangleq A^\top \underline{f}(A\underline{x}),$$

where $[\underline{f}(\underline{x})]_i = f(x_i)$ and A averages w adjacent values

$$A = \frac{1}{w}\begin{bmatrix} 1 & 1 & \cdots & 1 & 0 & \cdots \\ 0 & 1 & 1 & \ddots & 1 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

  - i.e., avg right w positions, apply f, then avg left w positions

- Coupled potential: $U_{\mathrm{c}}(\underline{x}) = \frac{1}{2}\sum_{i=0}^{\infty} x_i^2 - \sum_{i=0}^{\infty} F\left(\frac{1}{w}\sum_{j=0}^{w-1} x_{i+j}\right)$
  - Satisfies $\nabla U_{\mathrm{c}}(\underline{x}) = \underline{x} - A^\top \underline{f}(A\underline{x})$
  - Danger: there be ~~dragons~~ infinities

Coupled Monotone Recursion (2)

- Properties of T (note: $\underline{x} \preceq \underline{y} \Leftrightarrow x_i \leq y_i$ for all i)
  - T is monotone: $\underline{x} \preceq \underline{y}$ implies $T\underline{x} \preceq T\underline{y}$
  - T preserves spatial order: $x_{i+1} \geq x_i$ implies $[T\underline{x}]_{i+1} \geq [T\underline{x}]_i$

Coupled Monotone Recursion (2)

- Properties of T (note: $\underline{x} \preceq \underline{y} \Leftrightarrow x_i \leq y_i$ for all i)
  - T is monotone: $\underline{x} \preceq \underline{y}$ implies $T\underline{x} \preceq T\underline{y}$
  - T preserves spatial order: $x_{i+1} \geq x_i$ implies $[T\underline{x}]_{i+1} \geq [T\underline{x}]_i$
- For $\underline{x}^{(0)} = \underline{1}$, iterates $x_i^{(\ell)}$ are decreasing in ℓ and increasing in i
  - Spatial limit exists: $x_\infty^{(\ell)} = \lim_{i\to\infty} x_i^{(\ell)}$
  - Iteration limit exists: $x_i^{(\infty)} = \lim_{\ell\to\infty} x_i^{(\ell)}$
  - Iteration limit satisfies fixed point: $\underline{x}^{(\infty)} = T\underline{x}^{(\infty)}$
  - Double limit satisfies fixed point: $x_\infty^{(\infty)} = f\big(x_\infty^{(\infty)}\big)$

Intuition Behind Threshold Saturation

- Between the BP and MAP threshold
  - decoding trajectory looks like a right-moving wave

Intuition Behind Threshold Saturation

- Between the BP and MAP threshold
  - decoding trajectory looks like a right-moving wave
  - we know recursion converges pointwise to a limit

Intuition Behind Threshold Saturation

- Between the BP and MAP threshold
  - decoding trajectory looks like a right-moving wave
  - we know recursion converges pointwise to a limit
  - if limit not $\underline{0}$, then compute energy change due to right shift

Intuition Behind Threshold Saturation

- Between the BP and MAP threshold
  - decoding trajectory looks like a right-moving wave
  - we know recursion converges pointwise to a limit
  - if limit not $\underline{0}$, then compute energy change due to right shift
- Right-shift S satisfies $[S\underline{x}]_i = x_{i-1}$ with $x_{-1} = 0$

Intuition Behind Threshold Saturation

- Between the BP and MAP threshold
  - decoding trajectory looks like a right-moving wave
  - we know recursion converges pointwise to a limit
  - if limit not $\underline{0}$, then compute energy change due to right shift
- Right-shift S satisfies $[S\underline{x}]_i = x_{i-1}$ with $x_{-1} = 0$
- Relative potential: $V_{\underline{x}}(t) = U_{\mathrm{c}}\big((1-t)\underline{x} + tS\underline{x}\big) - U_{\mathrm{c}}(\underline{x})$

Intuition Behind Threshold Saturation

- Between the BP and MAP threshold
  - decoding trajectory looks like a right-moving wave
  - we know recursion converges pointwise to a limit
  - if limit not $\underline{0}$, then compute energy change due to right shift
- Right-shift S satisfies $[S\underline{x}]_i = x_{i-1}$ with $x_{-1} = 0$
- Relative potential: $V_{\underline{x}}(t) = U_c\big((1-t)\underline{x} + tS\underline{x}\big) - U_c(\underline{x})$
  - If $x_{i+1} \geq x_i$ for all i, then $V_{\underline{x}}(t)$ well-defined for $t \in [0,1]$

Intuition Behind Threshold Saturation

- Between the BP and MAP threshold
  - decoding trajectory looks like a right-moving wave
  - we know recursion converges pointwise to a limit
  - if limit not $\underline{0}$, then compute energy change due to right shift
- Right-shift S satisfies $[S\underline{x}]_i = x_{i-1}$ with $x_{-1} = 0$
- Relative potential: $V_{\underline{x}}(t) = U_{\mathrm{c}}\big((1-t)\underline{x} + tS\underline{x}\big) - U_{\mathrm{c}}(\underline{x})$
  - If $x_{i+1} \geq x_i$ for all i, then $V_{\underline{x}}(t)$ well-defined for $t \in [0,1]$
  - For $t = 1$, one gets a telescoping sum that shows

$$V_{\underline{x}}(1) \leq -U_{\mathrm{s}}(x_\infty)$$

Threshold Saturation

Theorem

If $f(0)=0$ and $f'(0)<1$ (0 is stable f.p.) with $U_{\mathrm{s}}(x)>0$ for $x\in(0,1]$, then $\exists w_0 < \infty$ such that $x_{\infty}^{(\infty)} = 0$ for all $w > w_0$.

Threshold Saturation

Theorem

If $f(0)=0$ and $f'(0)<1$ (0 is stable f.p.) with $U_{\mathrm{s}}(x)>0$ for $x\in(0,1]$, then $\exists w_0 < \infty$ such that $x_\infty^{(\infty)} = 0$ for all $w > w_0$.

- Define relative potential (with $x_i(t) \triangleq x_i + t(x_{i-1} - x_i)$)

$$V_{\underline{x}}(t) \triangleq \frac{1}{2}\sum_{i=0}^{\infty}\left(x_i(t)^2-(x_i)^2\right)-\sum_{i=0}^{\infty}\left[F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i(t)\right)-F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i\right)\right]$$

- Sketch of Proof:

Threshold Saturation

Theorem

If $f(0)=0$ and $f'(0)<1$ (0 is stable f.p.) with $U_{\mathrm{s}}(x)>0$ for $x\in(0,1]$, then $\exists w_0<\infty$ such that $x_\infty^{(\infty)}=0$ for all $w>w_0$.

- Define relative potential (with $x_i(t) \triangleq x_i + t(x_{i-1}-x_i)$)

$$V_{\underline{x}}(t) \triangleq \frac{1}{2}\sum_{i=0}^{\infty}\left(x_i(t)^2-(x_i)^2\right)-\sum_{i=0}^{\infty}\left[F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i(t)\right)-F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i\right)\right]$$

- Sketch of Proof:
  - For $\underline{x}^{(0)}=\underline{1}$, let $\underline{z}=\underline{x}^{(\infty)}$ be limiting fixed-point of recursion

Threshold Saturation

Theorem

If $f(0)=0$ and $f'(0)<1$ (0 is stable f.p.) with $U_{\mathrm{s}}(x)>0$ for $x\in(0,1]$, then $\exists w_0<\infty$ such that $x_\infty^{(\infty)}=0$ for all $w>w_0$.

- Define relative potential (with $x_i(t)\triangleq x_i+t(x_{i-1}-x_i)$)

$$V_{\underline{x}}(t)\triangleq\frac{1}{2}\sum_{i=0}^{\infty}\left(x_i(t)^2-(x_i)^2\right)-\sum_{i=0}^{\infty}\left[F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i(t)\right)-F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i\right)\right]$$

- Sketch of Proof:
  - For $\underline{x}^{(0)}=\underline{1}$, let $\underline{z}=\underline{x}^{(\infty)}$ be limiting fixed-point of recursion
  - If $z_\infty=0$, then we're done. Suppose $z_\infty>0$

Threshold Saturation

Theorem

If $f(0)=0$ and $f'(0)<1$ (0 is stable f.p.) with $U_s(x)>0$ for $x \in (0,1]$, then $\exists w_0 < \infty$ such that $x_\infty^{(\infty)} = 0$ for all $w > w_0$.

- Define relative potential (with $x_i(t) \triangleq x_i + t(x_{i-1} - x_i)$)

$$V_{\underline{x}}(t) \triangleq \frac{1}{2}\sum_{i=0}^{\infty}\left(x_i(t)^2 - (x_i)^2\right) - \sum_{i=0}^{\infty}\left[F\left(\frac{1}{w}\sum_{j=0}^{w-1} x_i(t)\right) - F\left(\frac{1}{w}\sum_{j=0}^{w-1} x_i\right)\right]$$

- Sketch of Proof:
  - For $\underline{x}^{(0)} = \underline{1}$, let $\underline{z} = \underline{x}^{(\infty)}$ be limiting fixed-point of recursion
  - If $z_\infty = 0$, then we're done. Suppose $z_\infty > 0$
  - Then, $z_\infty = f(z_\infty) \geq$ smallest non-zero f.p. > 0 (ind. of w)

Threshold Saturation

Theorem

If $f(0)=0$ and $f'(0)<1$ (0 is stable f.p.) with $U_s(x)>0$ for $x\in(0,1]$,
then $\exists w_0<\infty$ such that $x_\infty^{(\infty)}=0$ for all $w>w_0$.

- Define relative potential (with $x_i(t)\triangleq x_i+t(x_{i-1}-x_i)$)

$$V_{\underline{x}}(t)\triangleq\frac{1}{2}\sum_{i=0}^{\infty}\left(x_i(t)^2-(x_i)^2\right)-\sum_{i=0}^{\infty}\left[F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i(t)\right)-F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i\right)\right]$$

- Sketch of Proof:
  - For $\underline{x}^{(0)}=\underline{1}$, let $\underline{z}=\underline{x}^{(\infty)}$ be limiting fixed-point of recursion
  - If $z_\infty=0$, then we're done. Suppose $z_\infty>0$
  - Then, $z_\infty=f(z_\infty)\geq$ smallest non-zero f.p. >0 (ind. of w)
  - Thus, $U(z_\infty)>0$ by hypothesis

Threshold Saturation

Theorem

If $f(0) = 0$ and $f'(0) < 1$ (0 is stable f.p.) with $U_{\mathrm{s}}(x) > 0$ for $x \in (0, 1]$, then $\exists w_0 < \infty$ such that $x_\infty^{(\infty)} = 0$ for all $w > w_0$.

- Define relative potential (with $x_i(t) \triangleq x_i + t(x_{i-1} - x_i)$)

$$V_{\underline{x}}(t) \triangleq \frac{1}{2} \sum_{i=0}^{\infty} \left(x_i(t)^2 - (x_i)^2 \right) - \sum_{i=0}^{\infty} \left[F\left(\frac{1}{w} \sum_{j=0}^{w-1} x_i(t) \right) - F\left(\frac{1}{w} \sum_{j=0}^{w-1} x_i \right) \right]$$

- Sketch of Proof:
  - For $\underline{x}^{(0)} = \underline{1}$, let $\underline{z} = \underline{x}^{(\infty)}$ be limiting fixed-point of recursion
  - If $z_\infty = 0$, then we're done. Suppose $z_\infty > 0$
  - Then, $z_\infty = f(z_\infty) \geq$ smallest non-zero f.p. > 0 (ind. of w)
  - Thus, $U(z_\infty) > 0$ by hypothesis
  - Telescoping sum for V shows $V_{\underline{z}}(1) \leq -U(z_\infty) < 0$

Threshold Saturation

Theorem

If $f(0)=0$ and $f'(0)<1$ (0 is stable f.p.) with $U_{\mathrm{s}}(x)>0$ for $x\in(0,1]$, then $\exists w_0 < \infty$ such that $x_\infty^{(\infty)} = 0$ for all $w > w_0$.

- Define relative potential (with $x_i(t) \triangleq x_i + t(x_{i-1} - x_i)$)

$$V_{\underline{x}}(t) \triangleq \frac{1}{2}\sum_{i=0}^{\infty}\left(x_i(t)^2-(x_i)^2\right)-\sum_{i=0}^{\infty}\left[F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i(t)\right)-F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i\right)\right]$$

- Sketch of Proof:
  - For $\underline{x}^{(0)} = \underline{1}$, let $\underline{z} = \underline{x}^{(\infty)}$ be limiting fixed-point of recursion
  - If $z_\infty = 0$, then we're done. Suppose $z_\infty > 0$
  - Then, $z_\infty = f(z_\infty) \geq$ smallest non-zero f.p. > 0 (ind. of w)
  - Thus, $U(z_\infty) > 0$ by hypothesis
  - Telescoping sum for V shows $V_{\underline{z}}(1) \leq -U(z_\infty) < 0$
  - Taylor series for V shows $|V_{\underline{z}}(1)| \leq K\frac{1}{w}\left(1 + \sup_{x\in[0,1]}|f'(x)|\right)$

Threshold Saturation

Theorem

If $f(0)=0$ and $f'(0)<1$ (0 is stable f.p.) with $U_s(x)>0$ for $x\in(0,1]$,
then $\exists w_0<\infty$ such that $x_\infty^{(\infty)}=0$ for all $w>w_0$.

- Define relative potential (with $x_i(t) \triangleq x_i + t(x_{i-1}-x_i)$)

$$V_{\underline{x}}(t) \triangleq \frac{1}{2}\sum_{i=0}^{\infty}\left(x_i(t)^2-(x_i)^2\right)-\sum_{i=0}^{\infty}\left[F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i(t)\right)-F\left(\frac{1}{w}\sum_{j=0}^{w-1}x_i\right)\right]$$

- Sketch of Proof:
  - For $\underline{x}^{(0)}=\underline{1}$, let $\underline{z}=\underline{x}^{(\infty)}$ be limiting fixed-point of recursion
  - If $z_\infty=0$, then we're done. Suppose $z_\infty>0$
  - Then, $z_\infty=f(z_\infty)\geq$ smallest non-zero f.p. >0 (ind. of w)
  - Thus, $U(z_\infty)>0$ by hypothesis
  - Telescoping sum for V shows $V_{\underline{z}}(1)\leq -U(z_\infty)<0$
  - Taylor series for V shows $|V_{\underline{z}}(1)|\leq K\frac{1}{w}\left(1+\sup_{x\in[0,1]}|f'(x)|\right)$
  - Thus, we get a contradiction for sufficiently large w

History of Threshold Saturation Proofs

- the BEC in 2010 [KRU11]
  - Established many properties and tools used by later approaches
- the Curie-Weiss model of physics in 2010 [HMU12]
- CDMA using a GA in 2011 [TTK12]
- CDMA with outer code via GA in 2011 [Tru12]
- compressive sensing using a GA in 2011 [DJM13]
- regular codes on BMS channels in 2012 [KRU13]
- increasing scalar and vector recursions in 2012 [YJNP14]
- irregular LDPC codes on BMS channels in 2012 [KYMP14]
- non-decreasing scalar recursions in 2012 [KRU15]
- non-binary LDPC codes on the BEC in 2014 [AG16]
- and more since 2014...

Summary and Open Problems

- Factor Graphs
  - Useful tool for modeling dependent random variables
  - Low-complexity algorithms for approximate inference
  - Density evolution can be used to analyze performance
- Spatial Coupling
  - Powerful technique for designing and understanding FGs.
  - Related to the statistical physics of supercooled liquids
  - Simple proof of threshold saturation for scalar recursions
- Interesting Open Problems
  - Code constructions that reduce the rate-loss due to termination
  - Compute the scaling exponent for SC codes
  - Finding new problems where SC provides benefits

Thanks for your attention

References I

[AG16] Iryna Andriyanova, Alexandre Graell i Amat. Threshold saturation for nonbinary SC-LDPC codes on the binary erasure channel.
arXiv preprint arXiv:1311.2003v4, 2016.

[DJM13] D.L. Donoho, A. Javanmard, A. Montanari.
Information-theoretically optimal compressed sensing via spatial coupling and approximate message passing.
IEEE Trans. Inform. Theory, 59(11):7434–7464, Nov. 2013.

[Gal63] Robert G. Gallager.
Low-Density Parity-Check Codes.
The M.I.T. Press, Cambridge, MA, USA, 1963.

[HMU12] S. H. Hassani, N. Macris, R. Urbanke.
Chains of mean-field models.
J. Stat. Mech., strona P02011, 2012.

[KFL01] Frank R. Kschischang, Brendan J. Frey, Hans-Andrea Loeliger.
Factor graphs and the sum-product algorithm.
IEEE Trans. Inform. Theory, 47(2):498–519, Feb. 2001.

[KRU11] S. Kudekar, T. J. Richardson, R. L. Urbanke.
Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC.
IEEE Trans. Inform. Theory, 57(2):803–834, Feb. 2011.

References II

[KRU13] S. Kudekar, T. Richardson, R. L. Urbanke.
Spatially coupled ensembles universally achieve capacity under belief propagation.
IEEE Trans. Inform. Theory, 59(12):7761–7813, Dec. 2013.

[KRU15] Shrinivas Kudekar, Thomas J Richardson, Rudiger L Urbanke.
Wave-like solutions of general 1-D spatially coupled systems.
IEEE Trans. Inform. Theory, 61(8):4117–4157, 2015.

[KYMP14] Santhosh Kumar, Andrew J. Young, Nicolas Macris, Henry D. Pfister.
Threshold saturation for spatially-coupled LDPC and LDGM codes on BMS channels.
IEEE Trans. Inform. Theory, 60(12):7389–7415, Dec. 2014.

[LMSS01] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman.
Efficient erasure correcting codes.
IEEE Trans. Inform. Theory, 47(2):569–584, Feb. 2001.

[Mac99] David J. C. MacKay.
Good error-correcting codes based on very sparse matrices.
IEEE Trans. Inform. Theory, 45(2):399–431, March 1999.

[MM09] M. Mezard, A. Montanari.
Information, Physics, and Computation.
Oxford University Press, New York, NY, 2009.

References III

[RSU01] Thomas J. Richardson, M. Amin Shokrollahi, Rüdiger L. Urbanke.
Design of capacity-approaching irregular low-density parity-check codes.
IEEE Trans. Inform. Theory, 47(2):619–637, Feb. 2001.

[RU01] Thomas J. Richardson, Rüdiger L. Urbanke.
The capacity of low-density parity-check codes under message-passing decoding.
IEEE Trans. Inform. Theory, 47(2):599–618, Feb. 2001.

[RU08] Thomas J. Richardson, Rüdiger L. Urbanke.
Modern Coding Theory.
Cambridge University Press, New York, NY, 2008.

[Tru12] Dmitri Truhachev.
Achieving AWGN multiple access channel capacity with spatial graph coupling.
IEEE Commun. Letters, 16(5):585–588, May 2012.

[TTK12] Keigo Takeuchi, Toshiyuki Tanaka, Tsutomu Kawabata.
A phenomenological study on threshold improvement via spatial coupling.
IEICE Trans. Fundamentals, E95-A(5):974–977, 2012.

[YJNP14] A. Yedla, Y.-Y. Jian, P. S. Nguyen, H. D. Pfister.
A simple proof of Maxwell saturation for coupled scalar recursions.
IEEE Trans. Inform. Theory, 60(11):6943–6965, Nov. 2014.