Low-Density Parity-Check Convolutional Codes[a]

What are they ?
How do they work ?
Are they any good ?

Kamil Sh. Zigangirov[bc], Daniel J. Costello, Jr.[b],

Michael Lentmaier[bc], Arvind Sridharan[b]

[b]University of Notre Dame

Notre Dame, IN 46556, USA

[c]Lund University

SE-22100 Lund, Sweden

ICORE Lecture Series,

University of Calgary,

Calgary, Canada

October 15, 2003

[a]This work was supported in part by NSF Grant CCR02-05310 and NASA Grant NAG5-10503.

Low-Density Parity-Check Convolutional Codes

What are they ?

How do they work ?

Are they any good ?

- History of LDPC coding
- LDPC block codes
- LDPC convolutional codes
- Graph representation of codes
- LDPC convolutional code construction techniques

HISTORY OF LDPC CODING

- 1962 - R. Gallager invented LDPC block codes
- 1975 - V. Zyablov and M. Pinsker analyzed iterative decoding of LDPC codes
- 1981 - R. M. Tanner generalized Gallager's LDPC codes and developed graph approach to LDPC coding
- 1993 - C. Berrou, A. Glaveux and P. Thitimajshima invented turbo codes
- 1999 - A. Jimenez and K. Zigangirov introduced LDPC convolutional codes
- 2003 - A. Sridharan and D. J. Costello suggested algebraic constructions of LDPC convolutional codes

BLOCK AND CONVOLUTIONAL CODES

- Block Codes were invented in the end of 1940s. The first nontrivial block codes were Hamming Codes.
- Convolutional Codes were invented by Elias in 1955.
- What is the differencs between this two codes?

Block codes are analog of batch production.

Convolutional codes are analog of conveyor (pipeline) production.

LOW-DENSITY PARITY-CHECK BLOCK CODES

Definition: A regular low-density parity-check (LDPC) block (N, J, K) code is a code defined by an $L \times N$ parity-check matrix $\boldsymbol{H}$, $L < N$, with exactly J ones in each column and K ones in each row.

- Number of rows $L = NJ/K$
- Rate $R \geq 1 - \frac{J}{K}$
- Fraction of ones decreases with increasing block length N

LDPC Block Codes: Gallager's Ensemble

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{M}\boldsymbol{P}_0 \\ \boldsymbol{M}\boldsymbol{P}_1 \\ \vdots \\ \boldsymbol{M}\boldsymbol{P}_{J-1} \end{pmatrix}$$

$\boldsymbol{M} = (\boldsymbol{I}|\cdots|\boldsymbol{I})$: concatenation of K size N/K identity matrices

$\boldsymbol{P}_j$: $N \times N$ permutation matrices

Example: $J = 3, K = 4, N = 20$

$\boldsymbol{H}=$

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

Gallager, 1962: *Low-Density Parity-Check Codes*, MIT Press

CONVOLUTIONAL CODES: SYNDROME FORMER

Convolutional code: set of sequences $\boldsymbol{v}$ satisfying $\boldsymbol{v}\boldsymbol{H}^{\mathrm{T}} = 0$

The semi-infinite matrix

$$\boldsymbol{H}^{\mathrm{T}} = \begin{pmatrix} \ddots & & \ddots & & \\ \boldsymbol{H}_0^{\mathrm{T}}(0) & \cdots & \boldsymbol{H}_{m_{\mathrm{s}}}^{\mathrm{T}}(m_{\mathrm{s}}) & & \\ & \ddots & & \ddots & \\ & & \boldsymbol{H}_0^{\mathrm{T}}(t) & \cdots & \boldsymbol{H}_{m_{\mathrm{s}}}^{\mathrm{T}}(t+m_{\mathrm{s}}) \\ & & \ddots & & \ddots \end{pmatrix}$$

where $$\boldsymbol{H}_i^{\mathrm{T}}(t) = \begin{pmatrix} h_i^{(1,1)}(t) & \cdots & h_i^{(1,c-b)}(t) \\ \vdots & & \vdots \\ h_i^{(c,1)}(t) & \cdots & h_i^{(c,c-b)}(t) \end{pmatrix}_{c\times(c-b)}$$

is called the syndrome former of the rate $R = b/c$ convolutional code.

syndrome former memory m_s: maximal width for which there is a t such that $\boldsymbol{H}_{m_s}^{\mathrm{T}}(t) \neq 0$

LDPC Convolutional Codes

Definition: A regular LDPC convolutional (m_s, J, K) code is a code defined by a syndrome former $\boldsymbol{H}^{\mathrm{T}}$, having exactly J ones in each row, where $J << (c-b)m_s$, and K ones in each column.

- practical implementation $\longrightarrow$ periodic syndrome formers.
- a syndrome former $\boldsymbol{H}^{\mathrm{T}}$ is called periodic if

$$\boldsymbol{H}_i^{\mathrm{T}}(t) = \boldsymbol{H}_i^{\mathrm{T}}(t+T), \qquad i = 0, 1, \ldots, m_{\mathsf{s}}, \quad t \in \mathbb{Z}$$

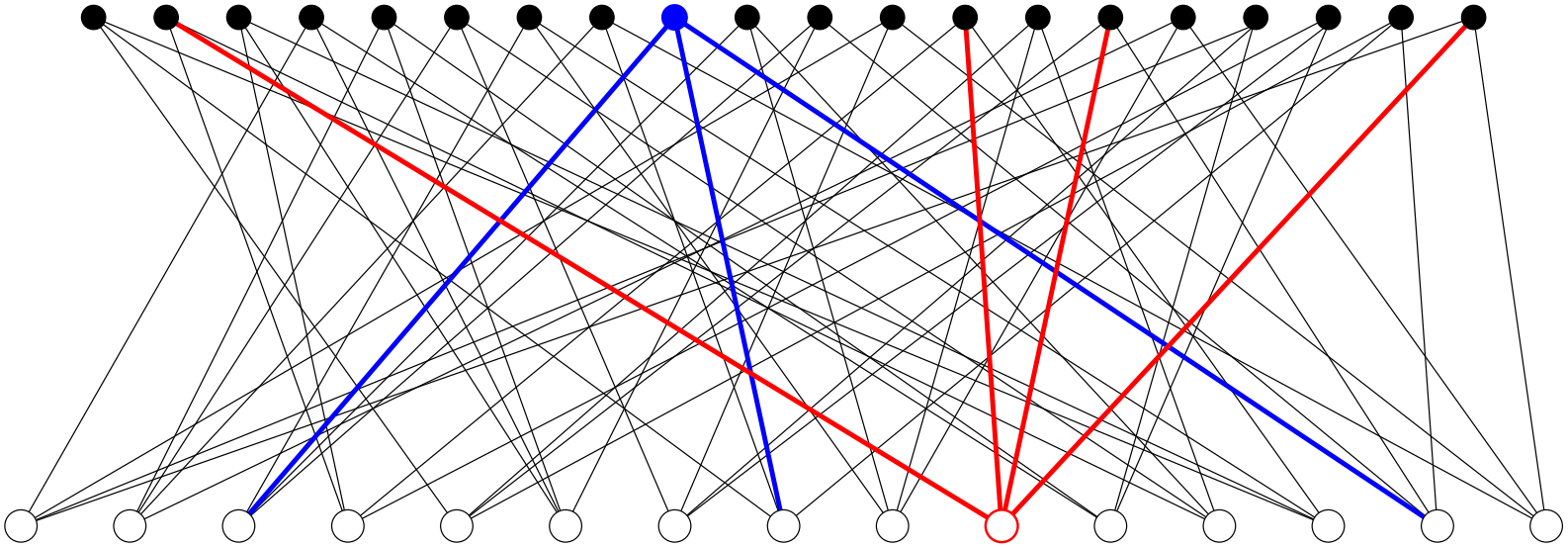
  for some period T.

- period $T = 1$ is the time invariant case

GRAPH REPRESENTATION: LDPC BLOCK CODES

Bipartite Tanner graph: follows directly from parity-check matrix

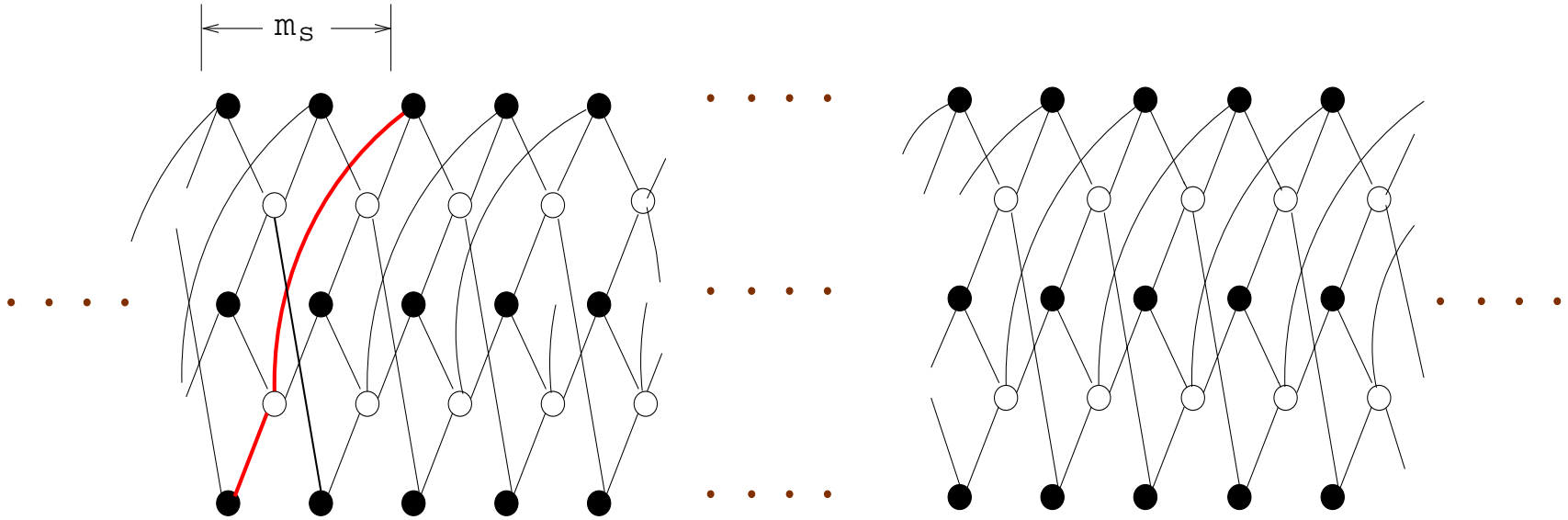
$N = 20$ symbol nodes, $J = 3$ edges from each



$L = 15$ parity-check nodes, $K = 4$ edges from each

GRAPH REPRESENTATION: LDPC CONVOLUTIONAL CODES

- graph has infinitely many nodes
- special structure due to memory constraints
- the time index t of symbols associated with the nodes is significant



LDPC Convolutional Codes: A Time-Varying Construction

Periodically time-varying LDPC convolutional code from LDPC block codes

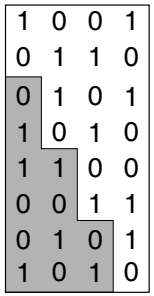
Example:

$m_s = 4$

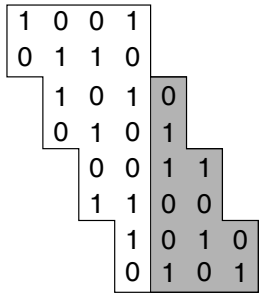
$J = 3, K = 6$

$R = 1/2$

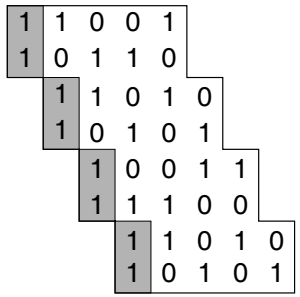
$T = m_s = 4$



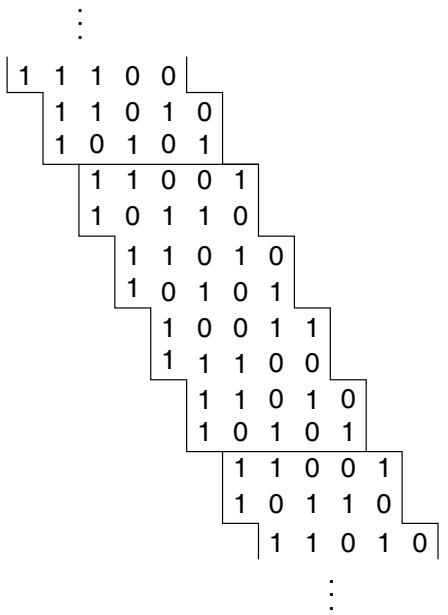
(a)



(b)



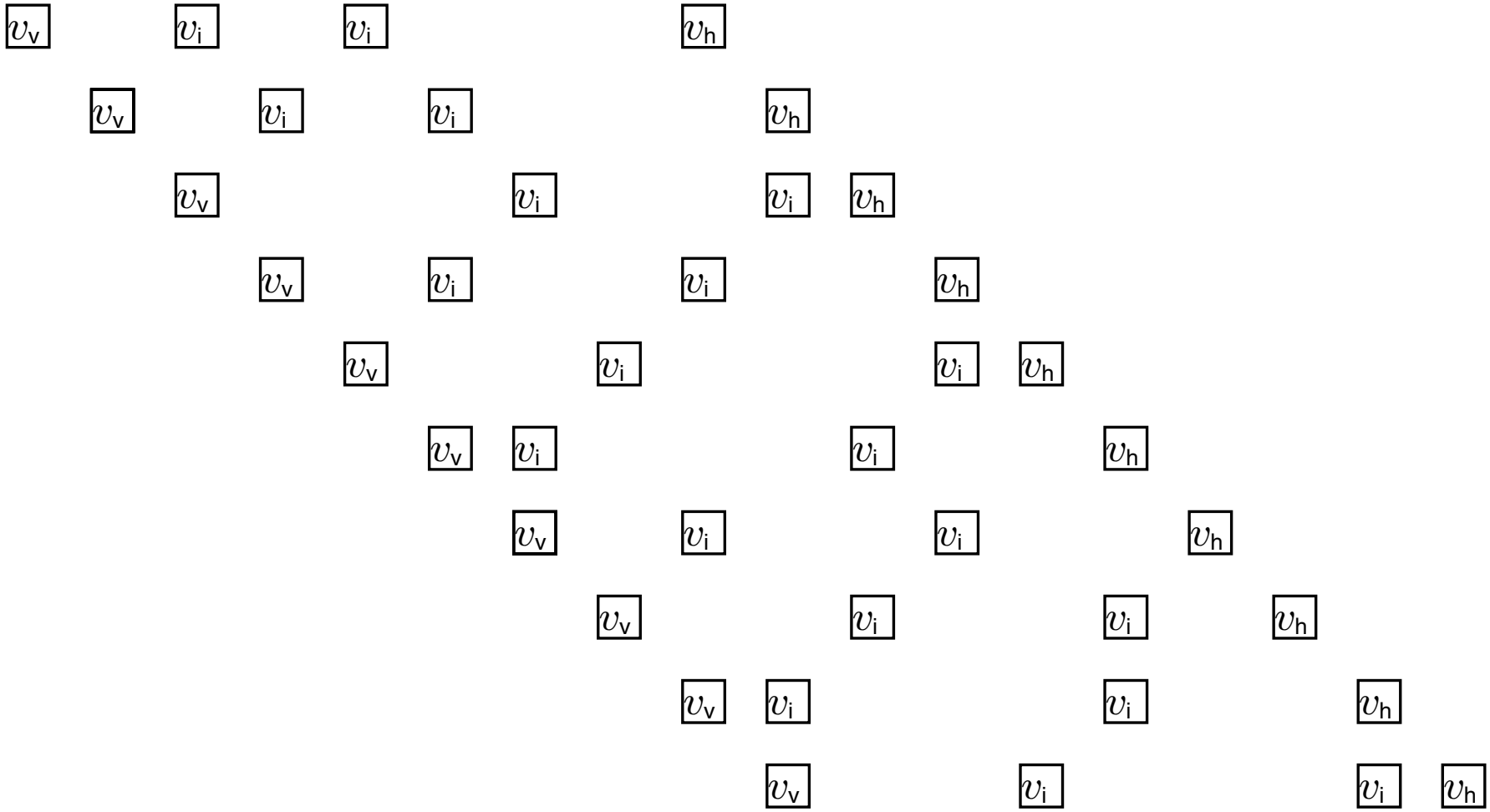
(c)



(d)

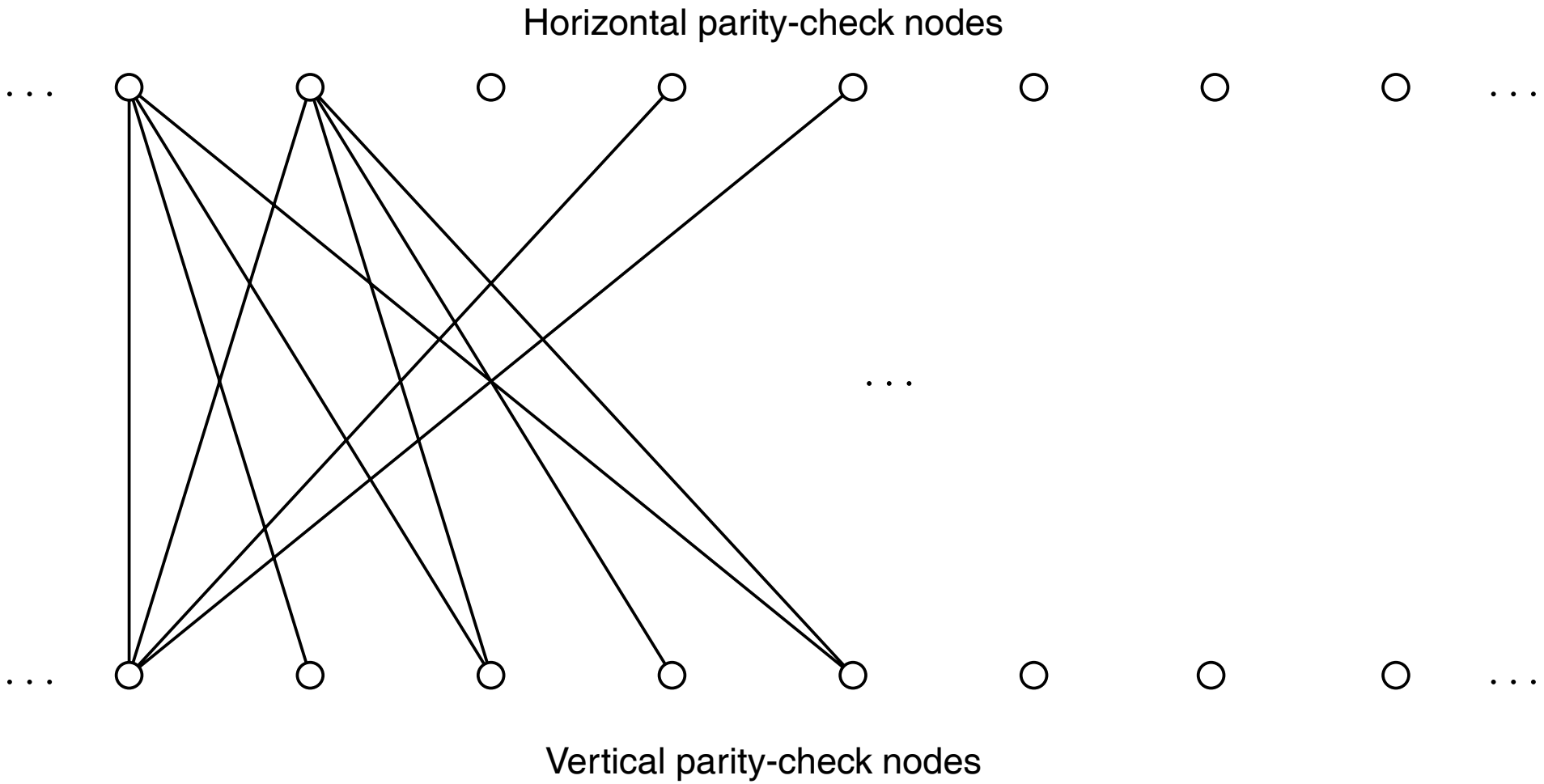
(Jiménez Feltström, Zigangirov)

BRAIDED BLOCK CODES: AN EXAMPLE



- Truhachev, Lentmaier, Zigangirov 2002

BRAIDED BLOCK CODES: GRAPH REPRESENTATION



LDPC Convolutional Codes: A Time-Invariant Construction

Quasi-Cyclic LDPC Block Code

- For any prime m, the set $\{0, 1, \ldots, m-1\}$ forms a field, $GF(m)$, of the integers modulo m.
- Order of the multiplicative group in $GF(m)$ is $m-1$.
- The parity-check matrix of a (J, K) regular LDPC block code is constructed by choosing elements a and b, $a, b \in GF(m) - \{0\}$:

$$o(a) = K,\ o(b) = J,$$

- J and K are chosen from the set of prime factors of $m-1$.
- Construct matrix $\boldsymbol{P}$ as:

$$\boldsymbol{P} = \begin{bmatrix} 1 & a & a^2 & \ldots & a^{K-1} \\ b & ab & a^2b & \ldots & a^{K-1}b \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ b^{J-1} & ab^{J-1} & a^2b^{J-1} & \ldots & a^{K-1}b^{J-1} \end{bmatrix}_{(J \times K)}$$

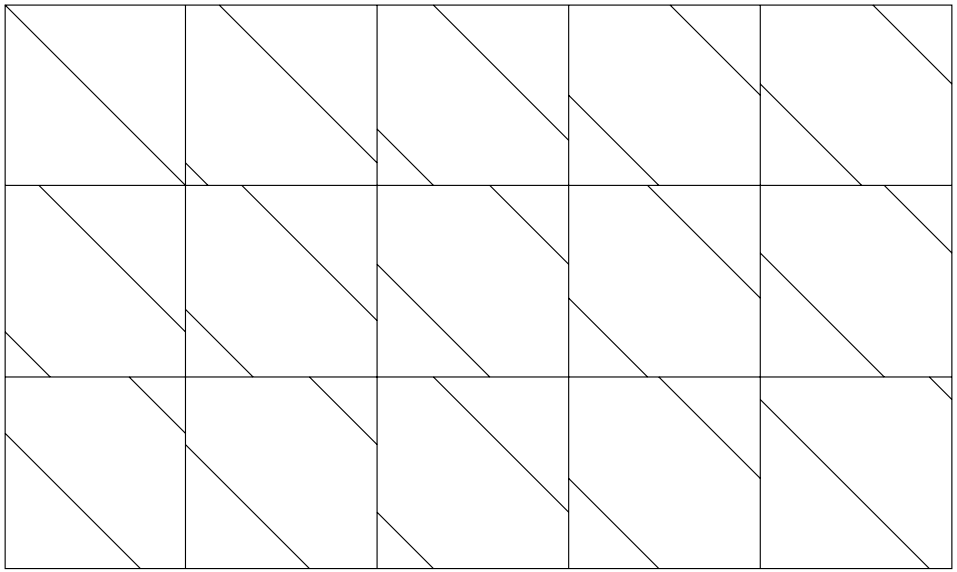
TIME INVARIANT CONSTRUCTION: QUASI-CYCLIC LDPC BLOCK CODE

$$\boldsymbol{H} = \begin{bmatrix} I_1 & I_a & I_{a^2} & \dots & I_{a^{K-1}} \\ I_b & I_{ab} & I_{a^2b} & \dots & I_{a^{K-1}b} \\ \dots & \dots & \dots & \dots & \dots \\ I_{b^{J-1}} & I_{ab^{J-1}} & I_{a^2b^{J-1}} & \dots & I_{a^{K-1}b^{J-1}} \end{bmatrix}_{(Jm \times Km)}$$

I_x is the $m \times m$ identity matrix circularly shifted to the right by $x - 1$ positions.

Example: $J = 3, K = 5$

$\boldsymbol{H}$ =



TIME INVARIANT CONSTRUCTION: EXAMPLE

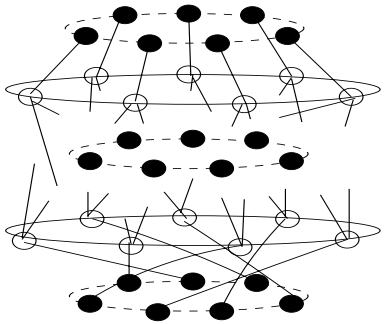
Elements $a = 2, b = 6$ are chosen from $GF(7)$:

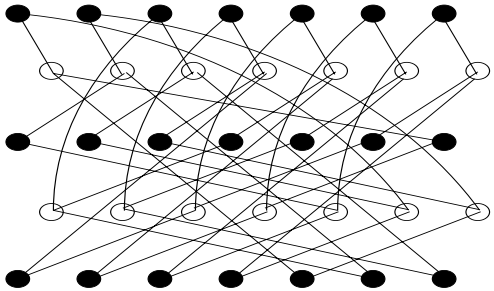
$$o(a) = 3, o(b) = 2$$

A (2,3) LDPC matrix is constructed as follows:

$$\boldsymbol{H} = \begin{bmatrix} I_1 & I_2 & I_4 \\ I_6 & I_5 & I_3 \end{bmatrix}_{(14\times 21),}$$

Example: $J = 2, K = 3, m = 7$





TIME INVARIANT CONSTRUCTION: LDPC CONVOLUTIONAL CODES

- Express every circulant block of the parity check matrix in polynomial form

$$\boldsymbol{H}(X) = \begin{bmatrix} 1 & X^{a-1} & X^{a^2-1} & \dots & X^{a^{K-1}-1} \\ X^{b-1} & X^{ab-1} & X^{a^2b-1} & \dots & X^{a^{K-1}b-1} \\ \dots & \dots & \dots & \dots & \dots \\ X^{b^{J-1}-1} & X^{ab^{J-1}-1} & X^{a^2b^{J-1}-1} & \dots & X^{a^{K-1}b^{J-1}-1} \end{bmatrix}_{(J \times K)}$$

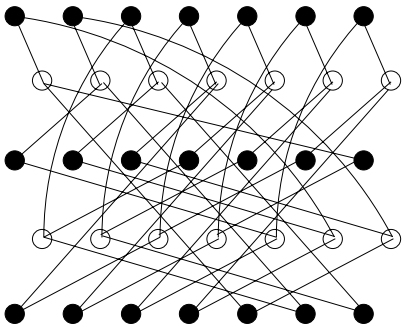
- Every entry in $\boldsymbol{H}(X)$ is a monomial in the ring $GF(2)[x]/(X^m + 1)$.
- $\boldsymbol{H}(X) \rightarrow$ parity check matrix of LDPC convolutional code.
- However, the monomials in $\boldsymbol{H}(X)$ are now treated as elements of $GF(2)(x)$,i.e., there is no wrapping around.
- Syndrome former memory $m_s \leq m$.

TIME INVARIANT CONSTRUCTION: EXAMPLE (CONT'D)

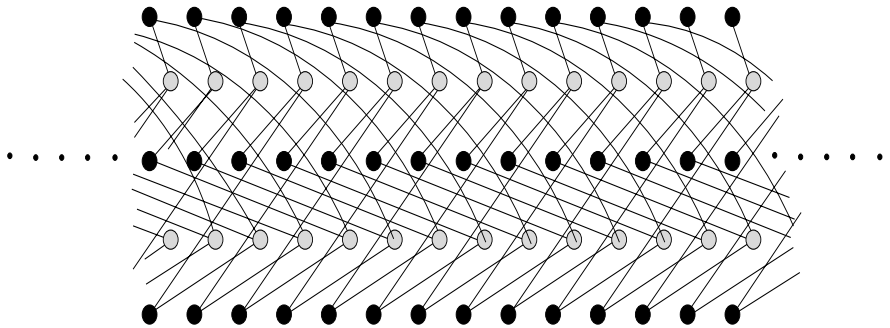
The corresponding rate $R = 1/3$ LDPC convolutional code is described by the parity check matrix:

$$\boldsymbol{H}(D) = \begin{bmatrix} 1 & D & D^3 \\ D^5 & D^4 & D^2 \end{bmatrix}_{(2\times 3)}$$

Example : $J = 2, K = 3, R = 1/3$ LDPC convolutional code



Quasi-cyclic block code



Convolutional code

Low-Density Parity-Check Convolutional codes

What are they ?

How do they work ?

Are they any good ?

- Encoding LDPC convolutional codes
- Decoding convolutional codes

ENCODING LDPC CONVOLUTIONAL CODES

- Obtain generator matrix $\boldsymbol{G}$ as usual from $\boldsymbol{H}$
- Shift register based encoding

Controller canonical form realization:

- b shift registers of length $O(m_s)$
- $\boldsymbol{G}$ not sparse $\Rightarrow$ encoding complexity per symbol is $O(m_s)$
- Encoding complexity can be reduced by directly using the syndrome former

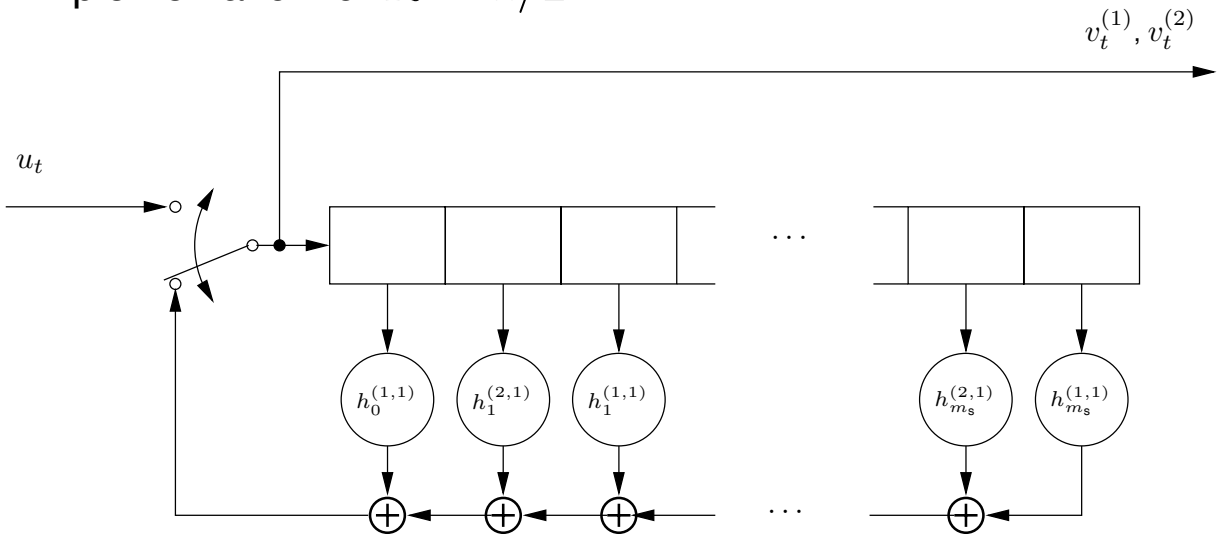
ENCODING LDPC CONVOLUTIONAL CODES

A systematic encoder for a rate b/c convolutional code can be obtained from

$$\boldsymbol{v}_t \boldsymbol{H}_0^{\mathrm{T}}(t) + \boldsymbol{v}_{t-1} \boldsymbol{H}_1^{\mathrm{T}}(t) + \cdots + \boldsymbol{v}_{t-m_{\mathrm{s}}} \boldsymbol{H}_{m_{\mathrm{s}}}^{\mathrm{T}}(t) = \mathbf{0},\ t \in \mathbb{Z}$$

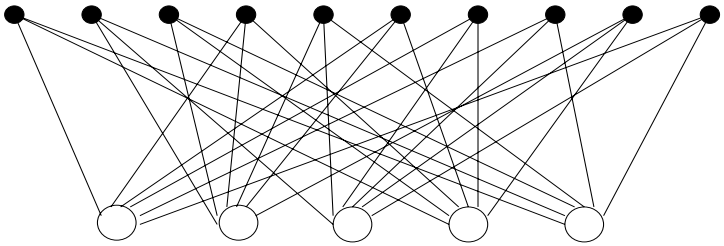
if $\boldsymbol{H}_0^{\mathrm{T}}(t)_{c\times(c-b)}$ has full rank

Example: shift register implementation for $R = 1/2$

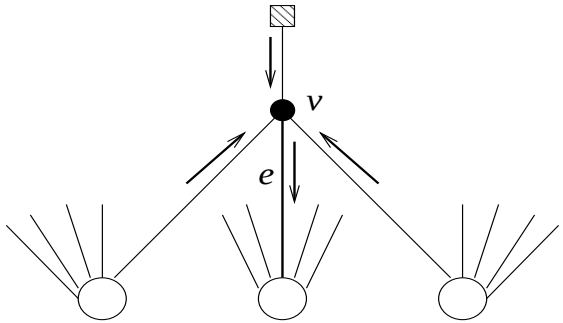


- the tap weights $h_i^{(\cdot,\cdot)}$, $i = 0, \ldots, m_{\mathrm{s}}$, are time varying
- each time $K-1$ taps are active $\Rightarrow$ complexity/symbol independent of m_{s}
- a similar encoder can be obtained for the time invariant codes, tap weights are time invariant in this case

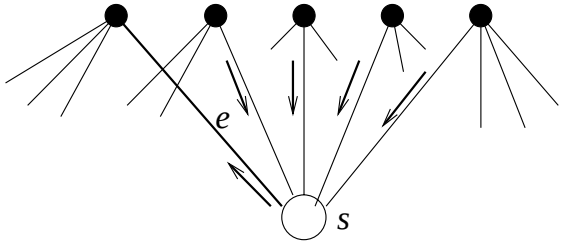
ITERATIVE DECODING (MESSAGE PASSING)



At a variable node



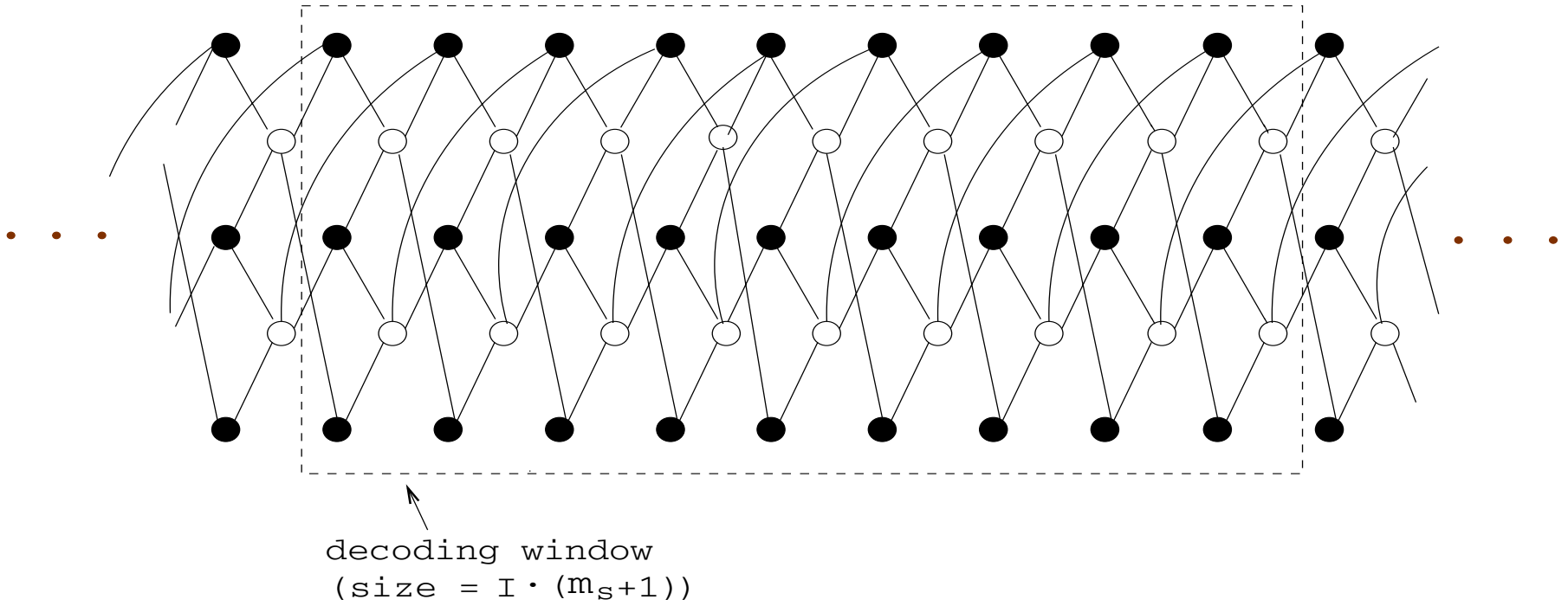
At a constraint node



WINDOW DECODING OF LDPC CONVOLUTIONAL CODES

- Sliding window decoder
- Window size, $\mathtt{I} \cdot \left(\mathtt{m_s} + 1\right)$.

  I - number of decoding iterations ; m_s - syndrome former memory.



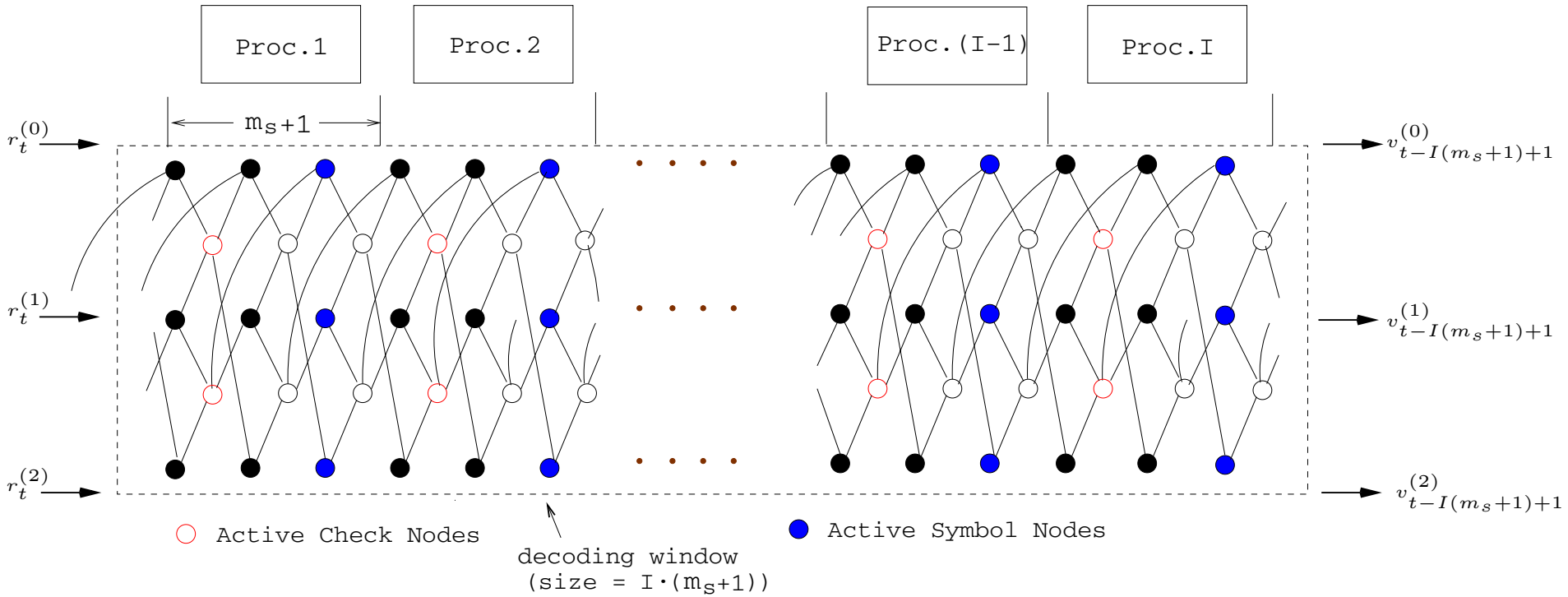
WINDOW DECODING OF LDPC CONVOLUTIONAL CODES

- Pipeline decoding possible: continuous output

  (Jiménez Feltström, Zigangirov, *IEEE Trans. IT*, Sept. '99)

- Initial delay equal to the window size.

- Iterations performed in parallel by independent processors.



Low-Density Parity-Check Convolutional codes

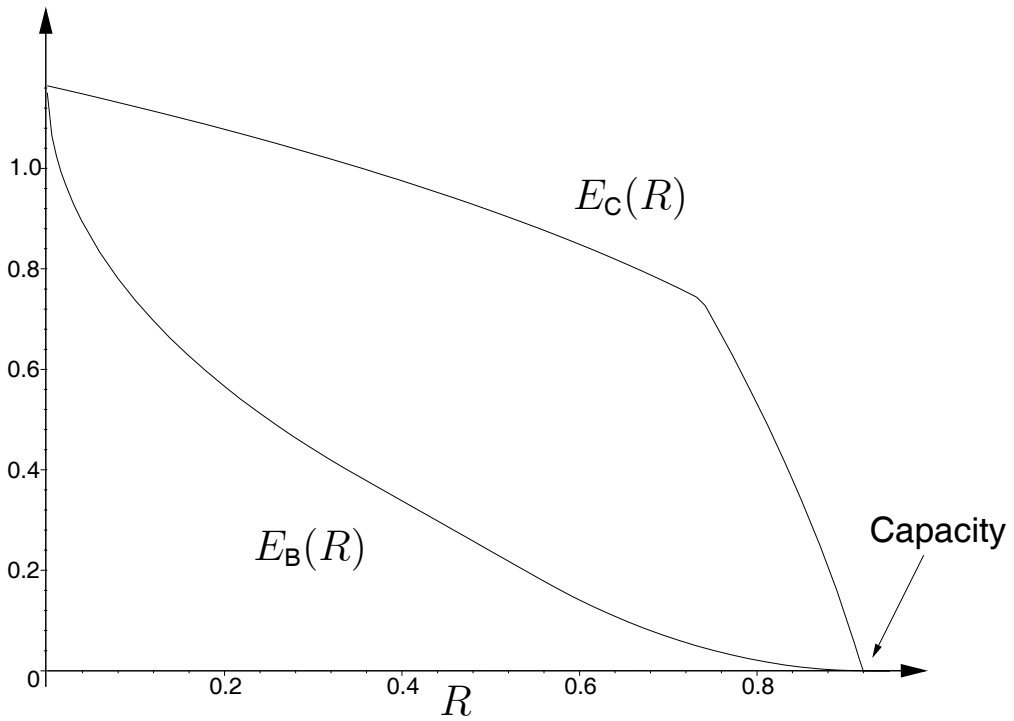
What are they ?

How do they work ?

Are they any good ?

- Error exponents for block and convolutional codes
- Asymptotic distance ratios
- Iterative decoding analysis
- Simulation results
- Better thresholds for LDPC convolutional codes ?

ML DECODING OF RANDOM CODES: ERROR EXPONENTS



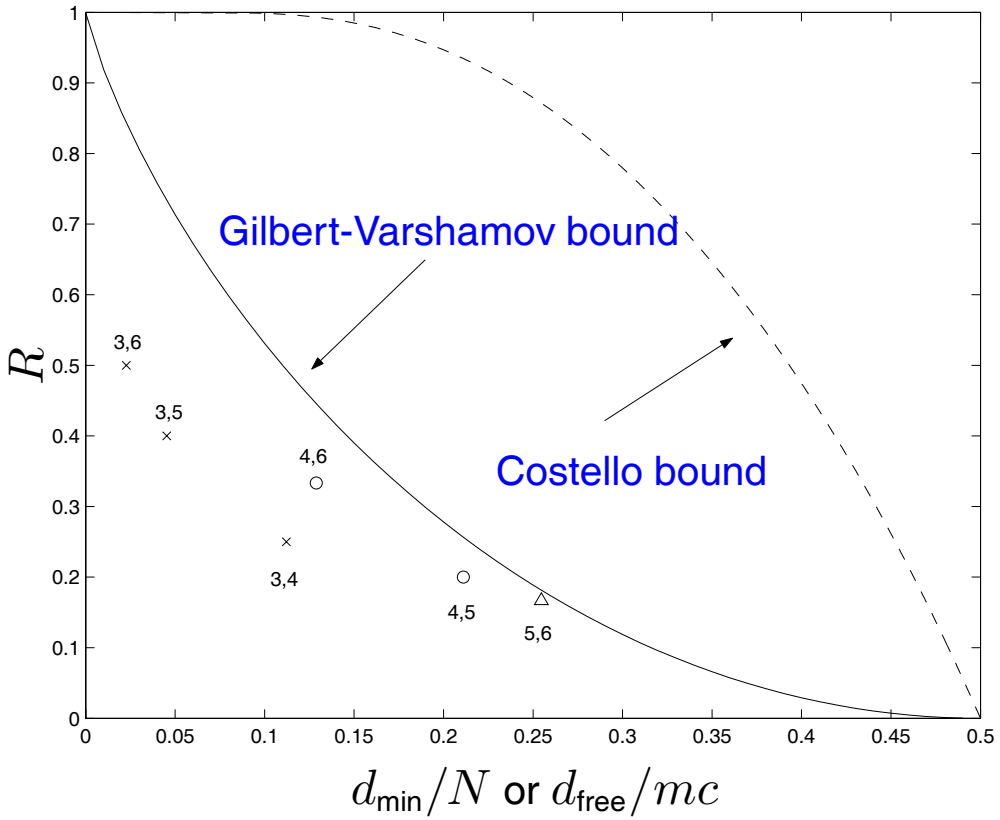
Block codes: block coding exponent $E_B(R)$ (Gallager)

$$P_{\text{Block}} \leq 2^{-N(E_B(R)+o(1))} \approx C^{-\frac{1}{R}(E_B(R)+o(1))}$$

Convolutional codes: convolutional coding exponent $E_C(R)$ (Yudkin, Viterbi)

$$P_{\text{Event}} \leq 2^{-mc(E_C(R)+o(1))} \approx C^{-\frac{1}{R}(E_C(R)+o(1))}$$

LDPC BLOCK CODES: ASYMPTOTIC DISTANCE RATIOS



d_{min}/N or d_{free}/mc

Gallager: for typical LDPC (J, K) block codes from ensemble:

$$d_{\text{min}} \geq \delta_{J,K} \cdot N, \quad \text{as } N \to \infty$$

$\Rightarrow$ asymptotically good codes

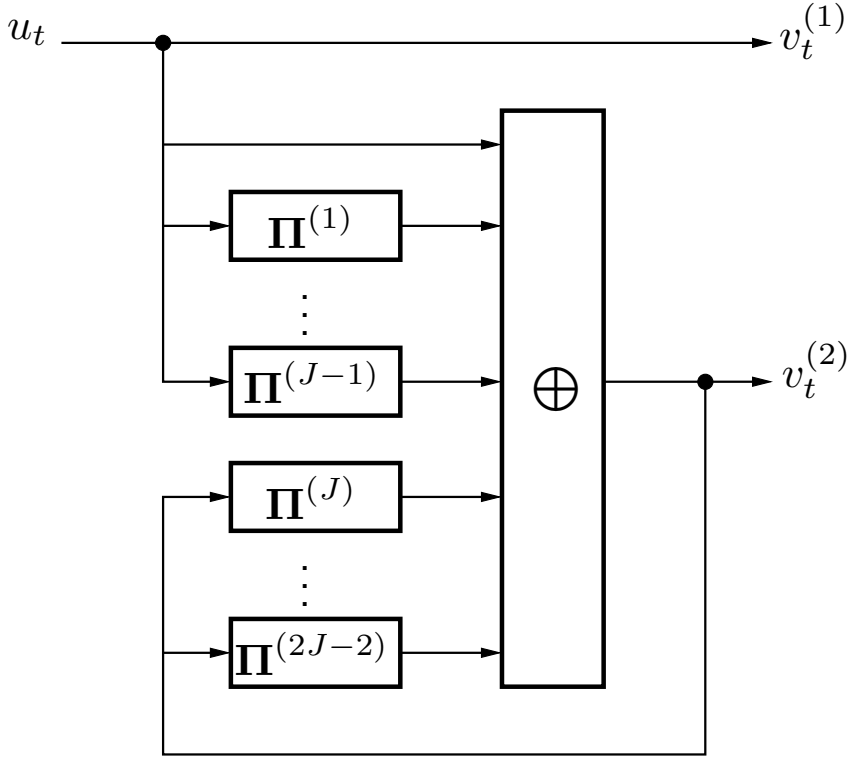
LDPC Convolutional Codes: Turbo-Like Representation

Encoder for regular LDPC convolutional codes:

Example:

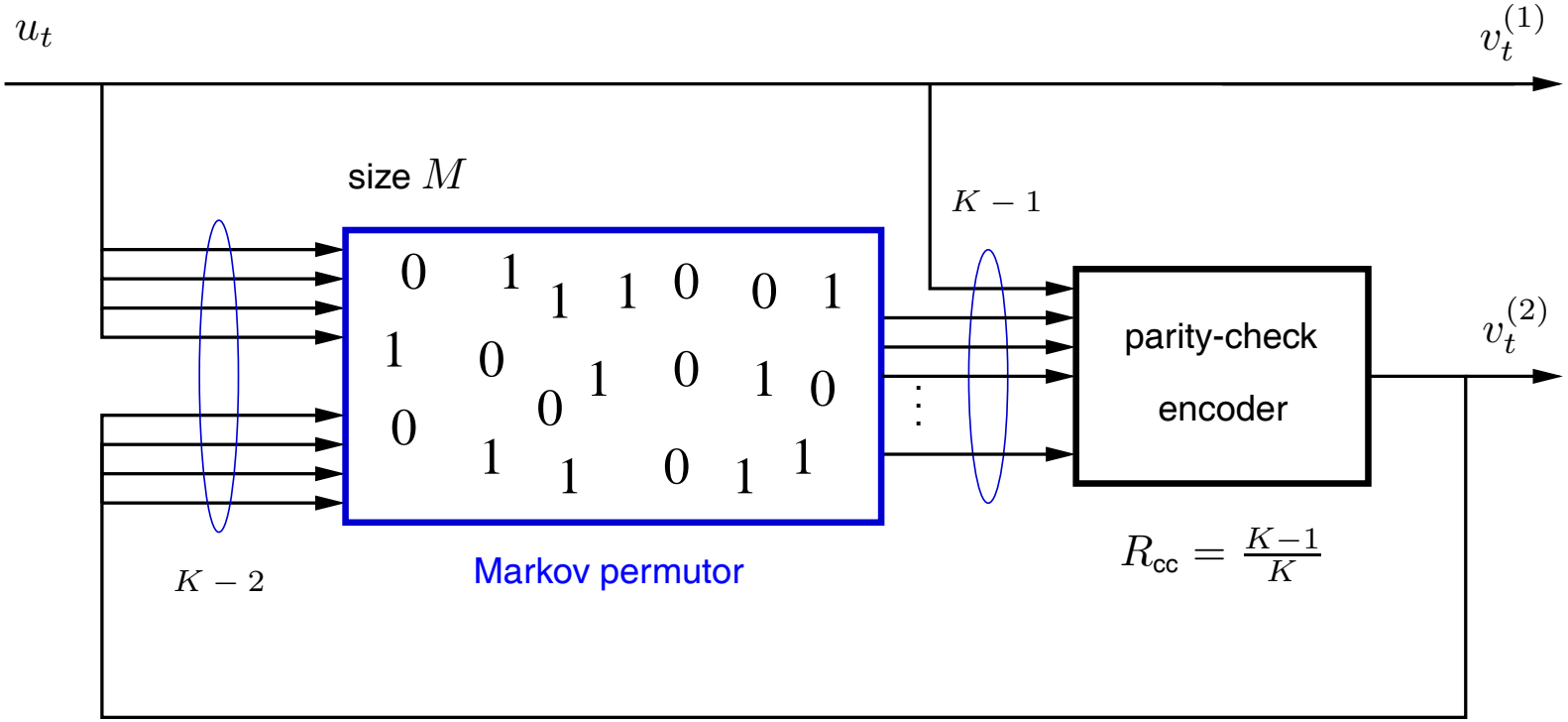
$K = 2J,$

$\Rightarrow\ R = 1/2$



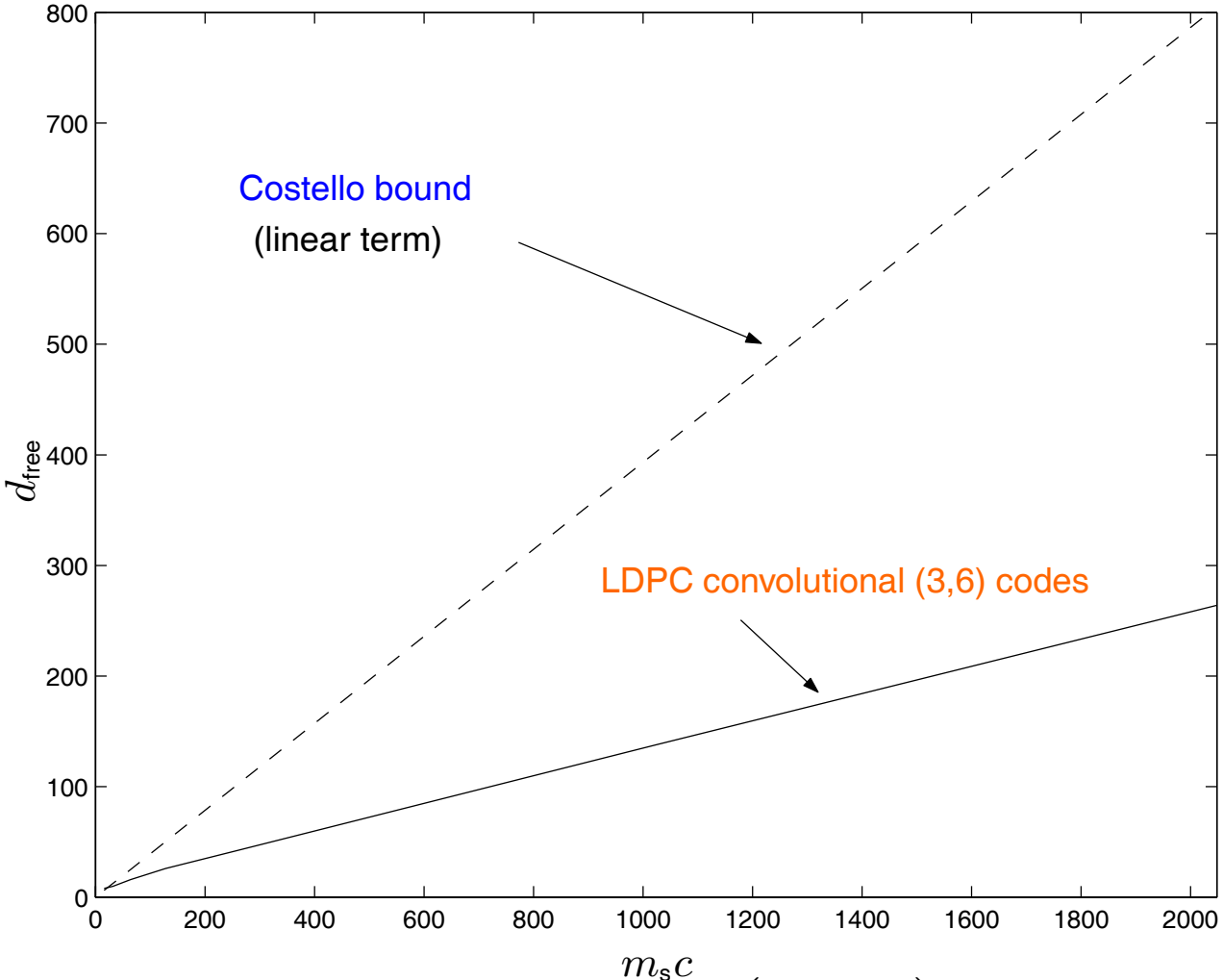
- rate $R_{\text{cc}} = \frac{K-1}{K}$ single parity-check component encoder (modulo-two adder)
- different permutations of the information sequence at the $K-1$ inputs

LDPC Convolutional Codes: Statistical Ensemble Analysis



- Markov permutor: modeled as box holding a constant number M of symbols
- defines an ensemble of LDPC convolutional codes
- average weight spectrum as solution to a system of recurrent equations (Engdahl, Lentmaier, Zigangirov, *AAECC*, '99)

LDPC Convolutional Codes: Lower Bound on Free Distance



Random Convolutional codes: $\frac{d_{\text{free}}}{m_s c} \geq \frac{R}{-\log_2(2^{(1-R)}-1)} + O\left(\frac{\log_2(m_s)}{m_s}\right)$

Distance Ratios: Comparison

Convolutional codes: $\frac{d_{\text{free}}}{m_{\text{s}} c} \geq \delta_{\text{C}}$, $m_{\text{s}} \to \infty$

LDPC convolutional codes	Costello bound	Ratio
$\delta_{3,6} = 0.125$	$\delta_{\text{C}} = 0.393$	$\delta_{\text{C}}/\delta_{3,6} = 3.144$

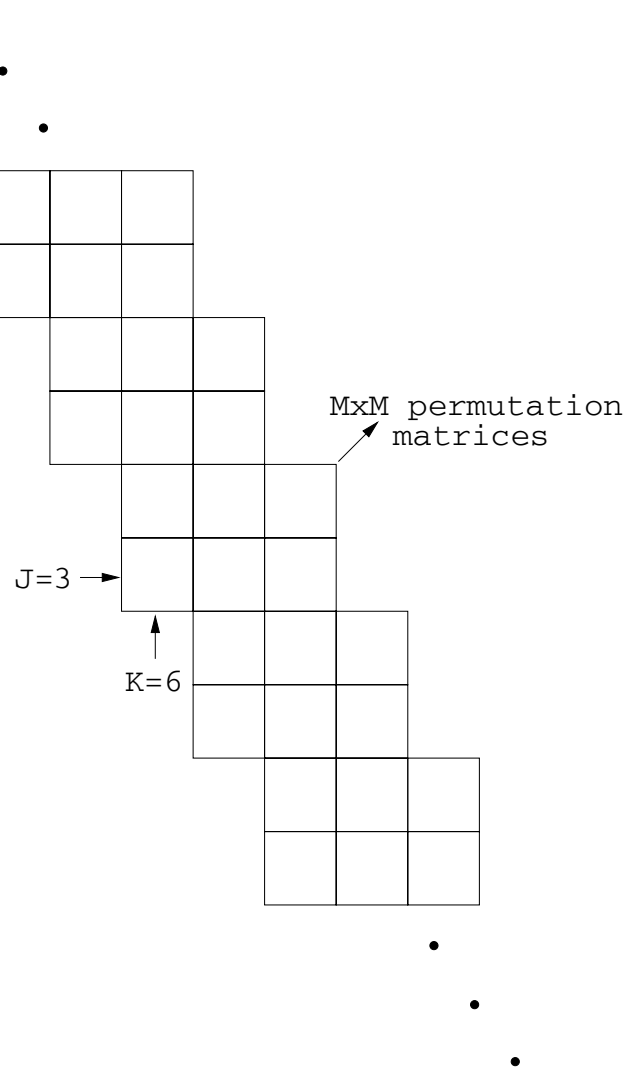
Block codes: $\frac{d_{\text{min}}}{N} \geq \delta_{\text{GV}}$, $N \to \infty$

LDPC block codes	Gilbert-Varshamov bound	Ratio
$\delta_{3,6} = 0.023$	$\delta_{\text{GV}} = 0.110$	$\delta_{\text{GV}}/\delta_{3,6} = 4.783$

LDPC Convolutional Codes: An Asymptotic Distance Bound

Consider the permutation matrix based ensemble:

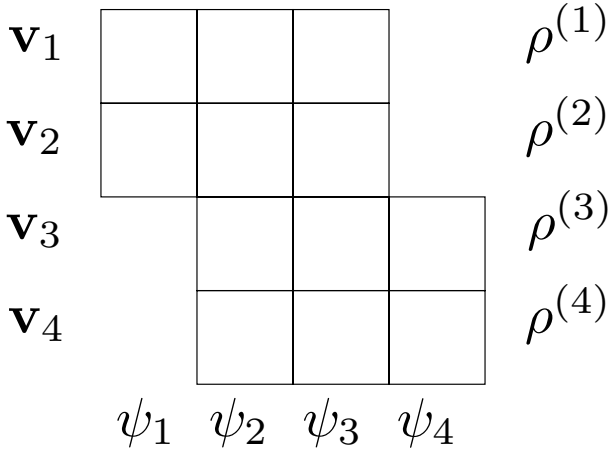
$$\boldsymbol{H}^{\mathrm{T}} =$$



LDPC Convolutional Codes: Combinatorial analysis

Example:

Code sequence of length $4M$



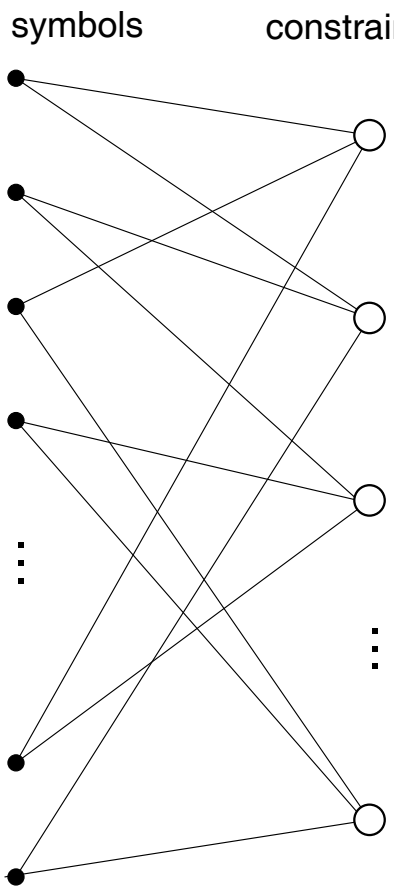
- Consider an arbitrary vector $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \cdots)$ with normalized weight $\rho = \frac{d}{M} = \sum_i \rho^{(i)}$
- $m_s c = KM$
- ψ_i = Probability that $\mathbf{v}$ satisfies ith block of constraints
- Probability that $\mathbf{v}$ is a code-sequence $\rightarrow \psi = \prod_i \psi_i$
- $\psi_i \leq e^{-MF(s_i, \rho^{(1)}, \rho^{(2)}, \cdots)}$ (Chernoff bounding technique)
- $\mathcal{E} = \psi \prod_i \binom{M}{M\rho^{(i)}} \rightarrow$ expected number of codewords of weight ρ
- If $\mathcal{E} < 1 \Rightarrow$ at least one code has $d_{free}/M > \rho$

LDPC CONVOLUTIONAL CODES: LOWER BOUND ON DISTANCE

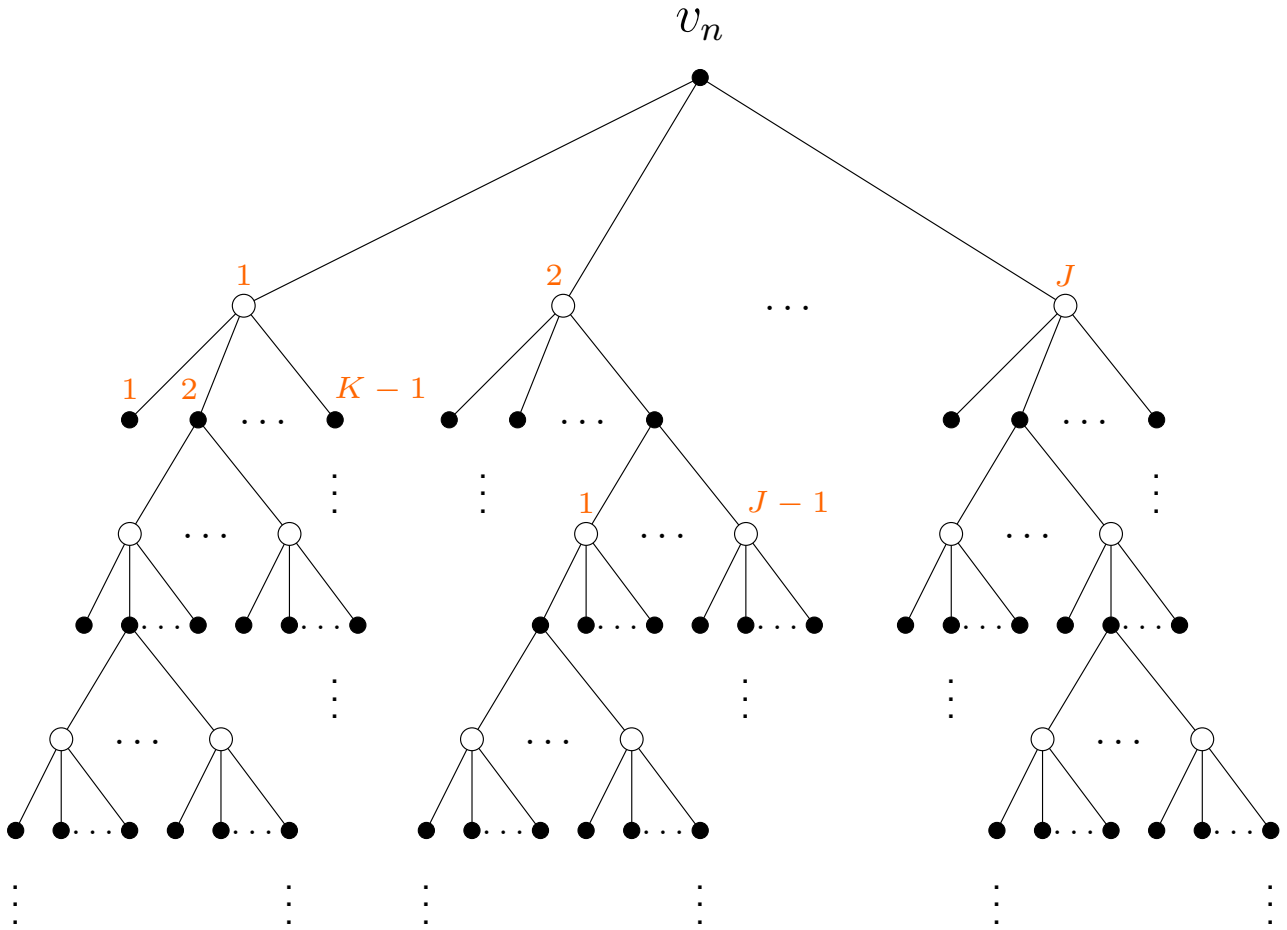
- Find s_i's and $\rho^{(i)}$'s so that ρ is maximized with constraint $\mathcal{E} < 1$
- Consider code-sequences $\mathbf{v}$ with lengths $2M, 4M, 6M, \ldots$
- Minimum ρ over all possible lengths gives lower bound on d_{free}
- Similar to Costello bound for random convolutional codes
- For $J = 3, K = 6$, $\frac{d_{\text{free}}}{m_s c} \geq 0.128$

Tree Representation of LDPC Codes

Tanner graph



Tree-like graph (clan)



Gallager, '62: arbitrary number of independent iterations possible

INDEPENDENT DECODING ITERATIONS

Theorem: [Gallager, '62] For any block length N there exists an l_0-nondegenerate LDPC block (J, K) code, for which

$$l_0 > \frac{\log N}{2\log(K-1)(J-1)} - c_1 \ ,$$

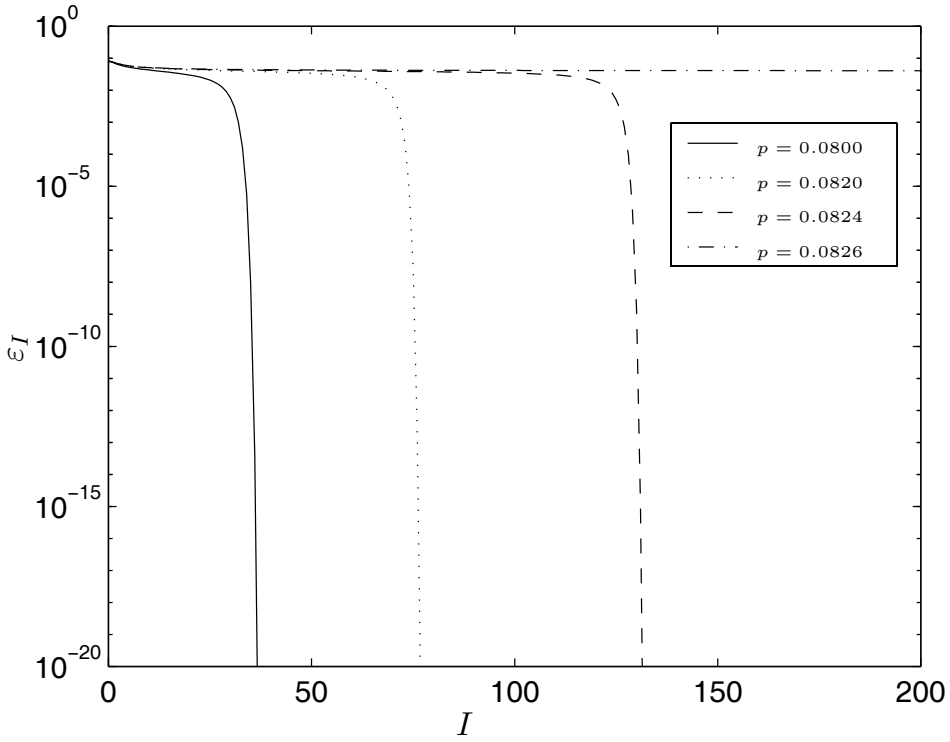
where the constant c_1 does not depend on N.

Theorem: [Lentmaier, Truhachev, Zigangirov, '01] For any memory m_{s} there exists an l_0-nondegenerate, rate $R = b/c$ LDPC convolutional (J, K) code, for which

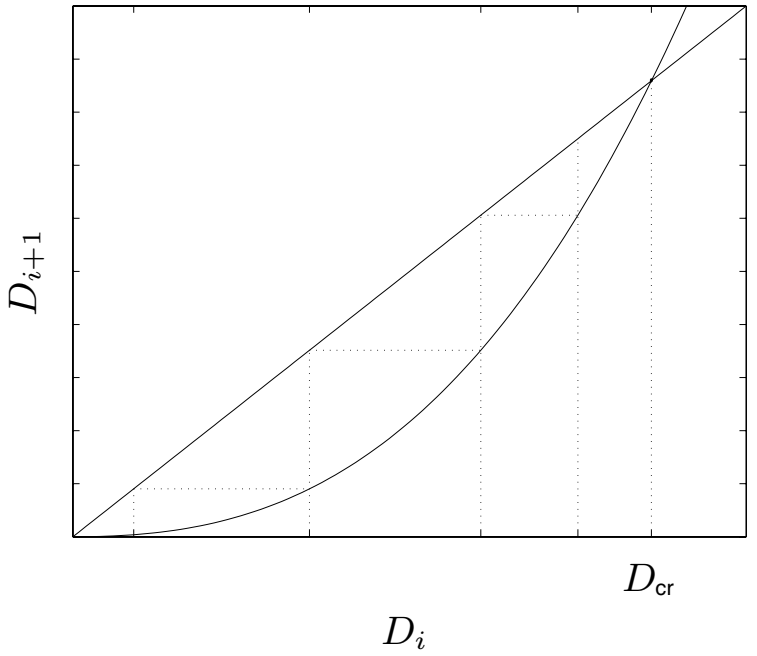
$$l_0 > \frac{\log_2(m_{\mathsf{s}}+1) + \log_2 \frac{c(KJ-K-J)}{4K}}{2\log_2(K-1)(J-1)} - 1.$$

Analysis of Iterative Decoding (BSC)

Error probability (density evolution):



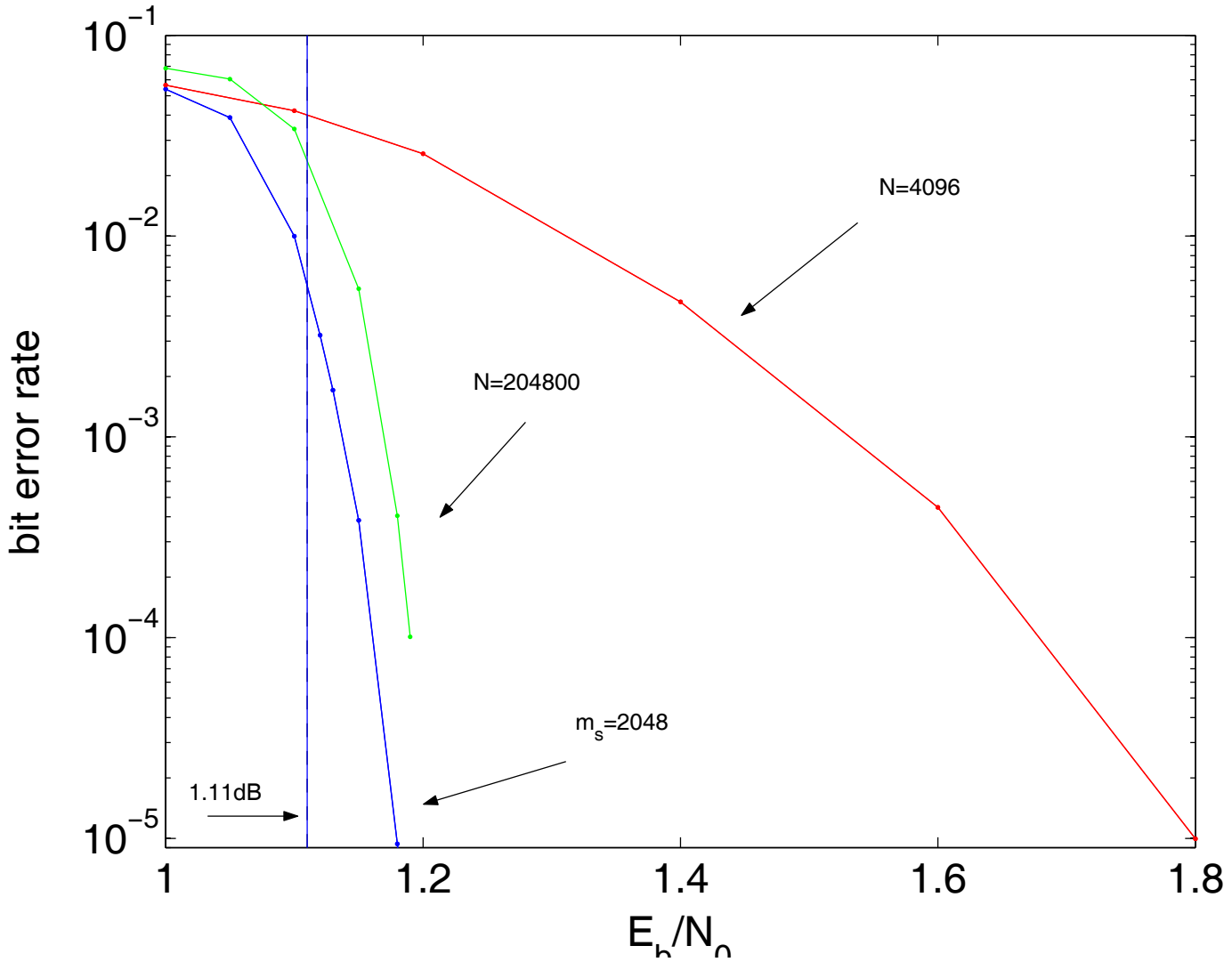
Bhattacharyya parameter:



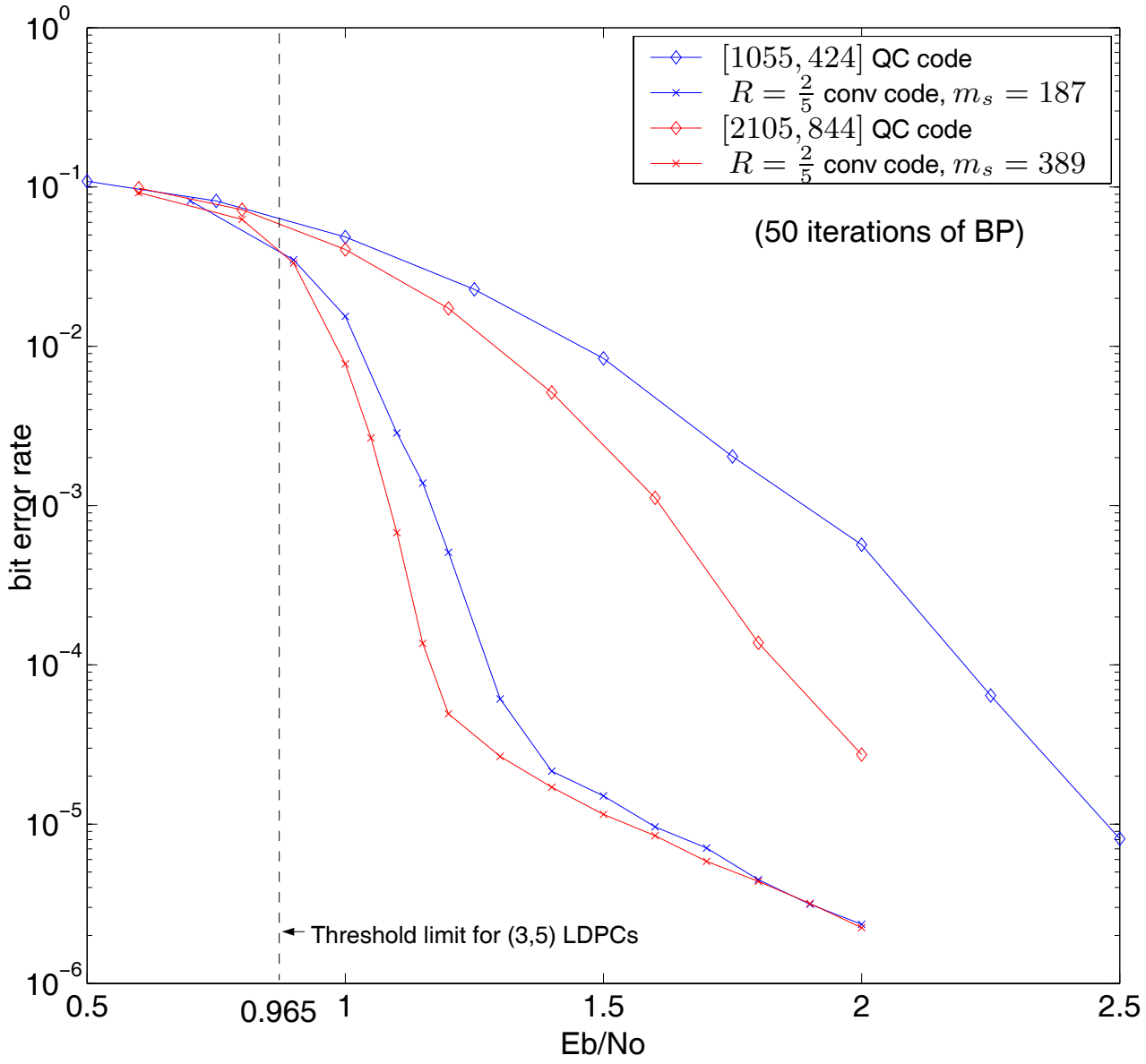
Union bound: $D_{i+1} \leq f(D_i)$

- any p for which $D_I = 2\sqrt{\varepsilon_I(1-\varepsilon_I)}$ falls below a critical value $D_{\text{cr}}(J, K, p)$ is a lower bound on the iterative limit (threshold)
- result is valid for LDPC block and convolutional codes

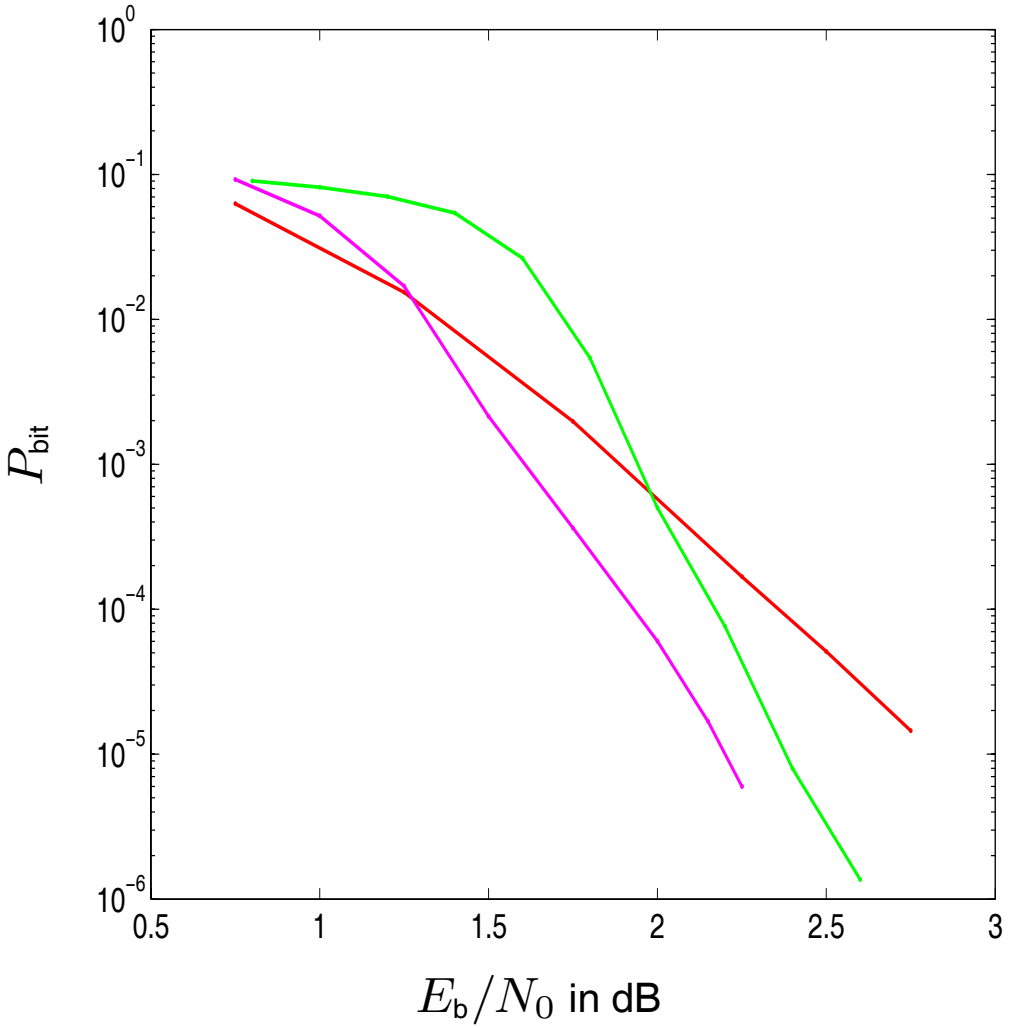
SIMULATION RESULTS: PERIODIC TIME-VARYING LDPC CONVOLUTIONAL CODES



SIMULATION RESULTS: TIME-INVARIANT LDPC CONVOLUTIONAL CODES



TIGHTLY BRAIDED BLOCK CODES: SIMULATION RESULTS



Iterative decoding, $I = 50$ iterations

Hamming components C^{v} and C^{h}

red: $(15, 11)$ and $(15, 11)$

Rate $R = 7/15 = 0.467$

memory $m = 7$

magenta: $(32, 26)$ and $(16, 11)$

Rate $R = 16/32 = 0.5$

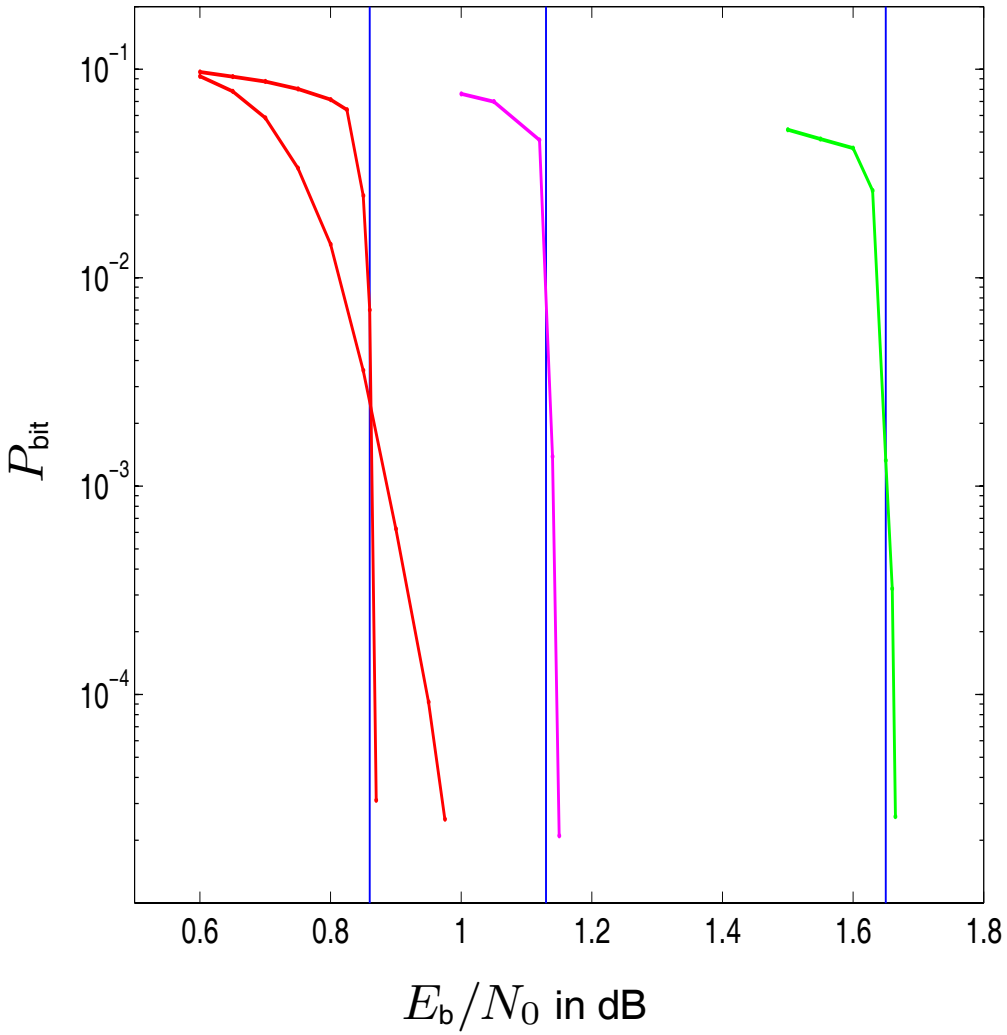
memory $m = 8$

green: $(31, 26)$ and $(31, 26)$

Rate $R = 21/31 = 0.677$

memory $m = 15$

SPARSELY BRAIDED BLOCK CODES: SIMULATION RESULTS



Iterative decoding, $I = 50$ iterations

Hamming components C^{v} and C^{h}

red: $(15, 11)$ and $(15, 11)$

Rate $R = 7/15$

memory $m = 105$ and $m = 4004$

magenta: $(32, 26)$ and $(16, 11)$

Rate $R = 16/32$

memory $m = 2240$

green: $(31, 26)$ and $(31, 26)$

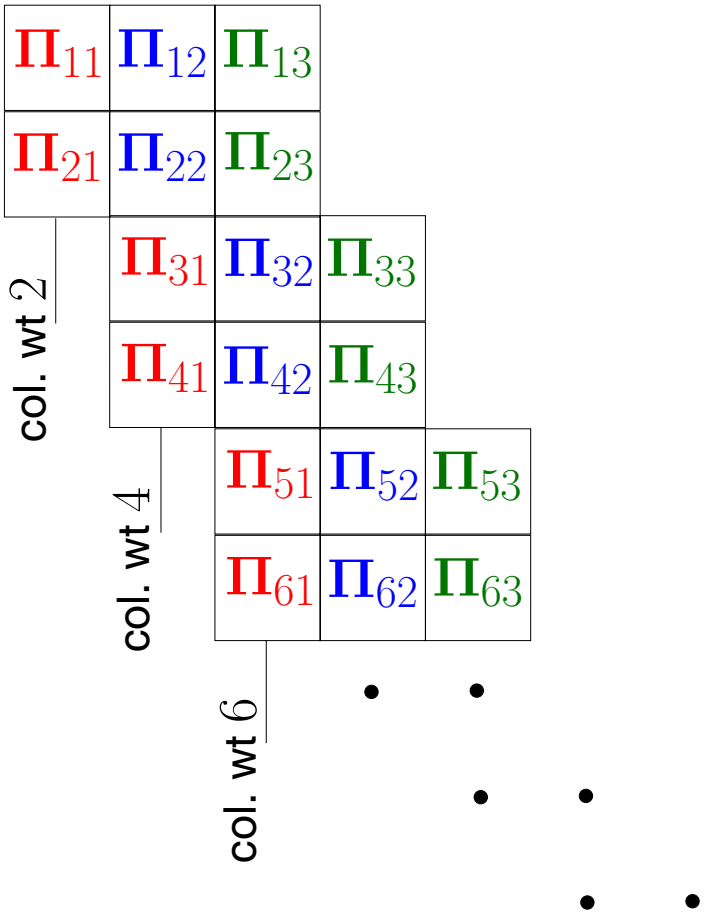
Rate $R = 21/31$

memory $m = 2250$

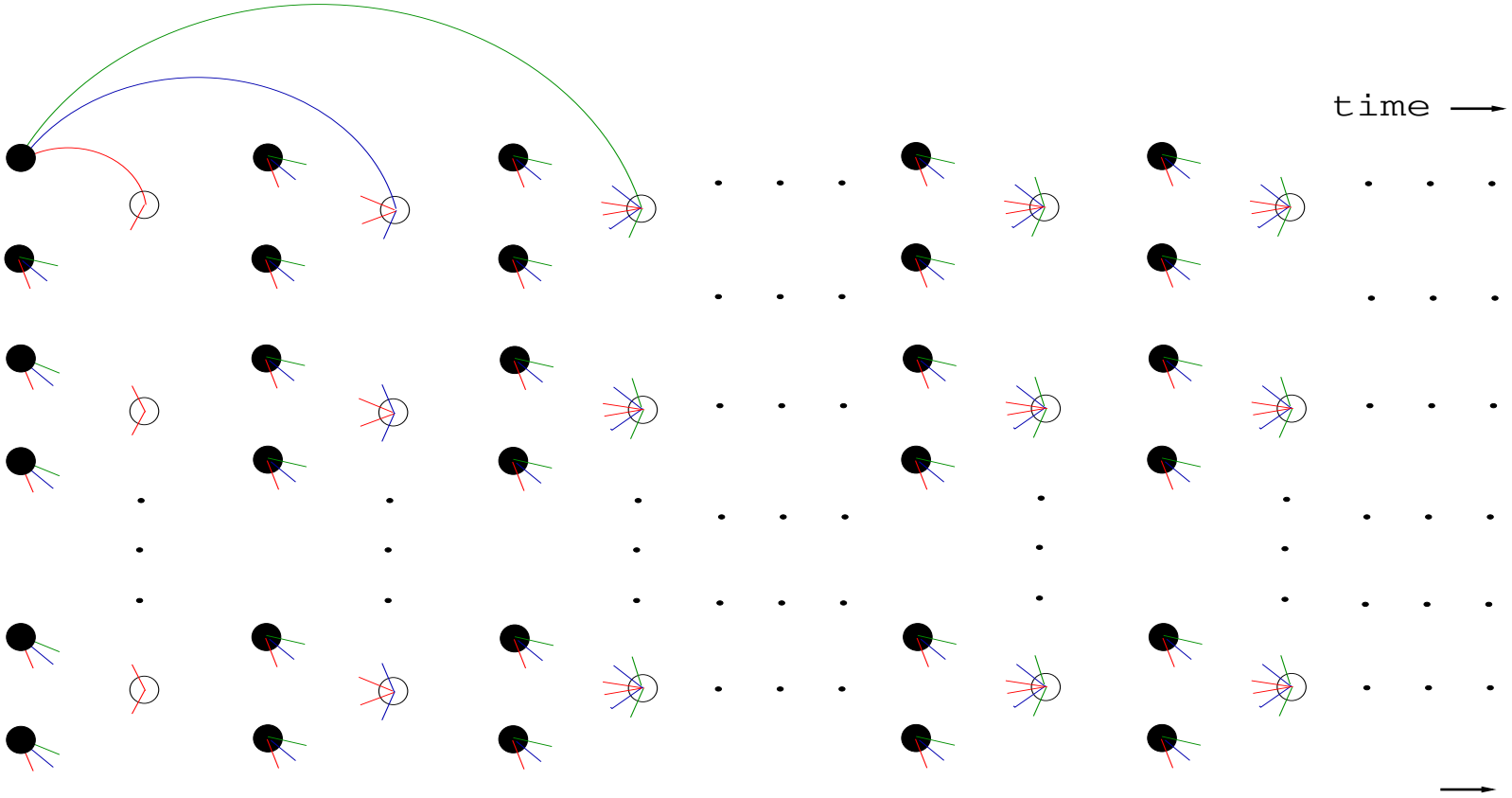
blue: upper bounds on the iterative limits

LDPC Convolutional Code Thresholds

- Syndrome former based on permutation matrices Π_{ij}
- Initial code symbols better protected
- $R = 1/2$, $J = 3$, $K = 6$



TANNER GRAPH



Better Thresholds?

- Binary erasure channel
- Assumes ∞ independent iterations
- Slightly lower rate for convolutional codes due to termination

(J, K)	R_{conv}	p_{conv}	p_{blk}
(3,6)	0.484	0.479	0.429
(4,8)	0.495	0.488	0.383
(5,10)	0.495	0.494	0.341

CONCLUSION

Why use LDPC convolutional codes:

- simple and efficient encoding
- continuous transmission with pipeline decoding
- better distance ratios
- better iterative decoding performance?
  - simulation results
  - faster convergence to threshold?
  - better threshold?