Doubly-Generalized Low-Density Parity-Check Codes

Marc Fossorier
University of Hawaii at Manoa

with

Enrico Paolini
University of Bologna

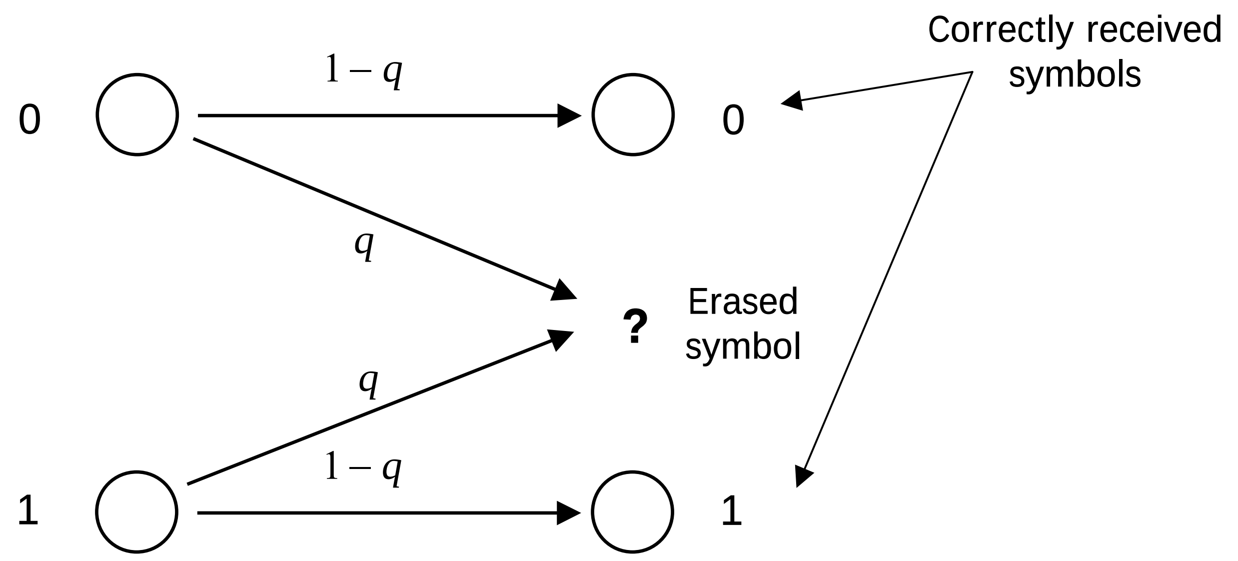
Yige Wang
University of Hawaii at Manoa

Part-I: Binary Erasure Channel

Outline Part-I:

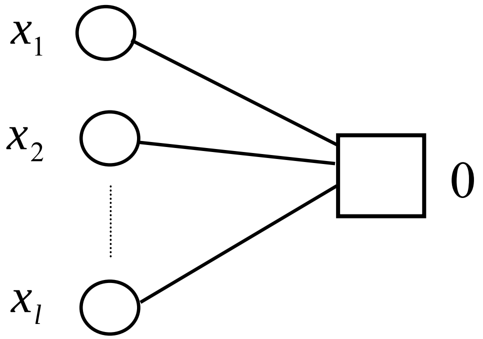
- **<u>Background: iterative decoding of LDPC codes on the BEC</u>**
- Generalized LDPC codes and Doubly-Generalized LDPC codes
- Asymptotic analysis of D-GLDPC codes based on EXIT charts
- Results

Memory-less Binary Erasure Channel (BEC)



- Each transmitted bit is either received without errors or it is lost
- Capacity: $C = 1 - q$ [bits / channel use]

Iterative decoding for the BEC: the basic idea



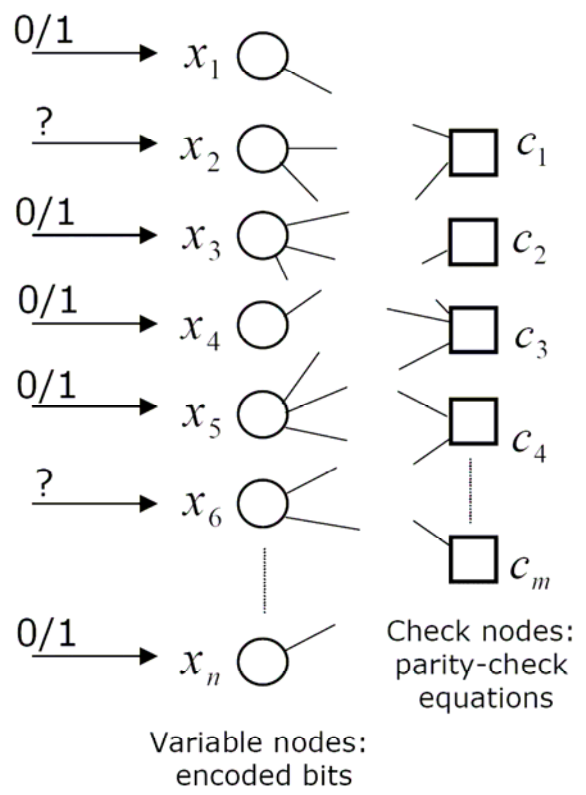
$$x_1 + x_2 + \ldots + x_l = 0$$

- The *l* bits $x_1,\ldots,x_l$ must satisfy a single parity-check constraint.
- If any of the *l* bits $x_1,\ldots,x_l$ is unknown, it can be reconstructed if the others are known.
- A single parity-check (SPC) code can correct at most one erasure.

Iterative decoding of LDPC codes on the BEC

Bipartite graph representation

- Degree of a variable (check) node.
- (λ,ρ): edge degree distribution.
- λ_i (ρ_i): fraction of edges towards the variable (check) nodes with degree i.



Iterative decoding

- The previously described decoding rule is iteratively applied to all the check nodes.
- Equivalent description as a message passing decoding algorithm (belief-propagation).
- Repetition codes and SPC codes.

Asymptotical analysis: the decoding threshold

• **Threshold of a degree distribution (λ,ρ)**: maximum value q* of the channel erasure probability, such that the fraction of erased bits converges to 0 in the limit where n tends to infinity, over a graph with edge degree distribution (λ,ρ).

• The asymptotic performance of LDPC codes under message passing decoder only depends on the edge degree distribution of the underlying bipartite graph.

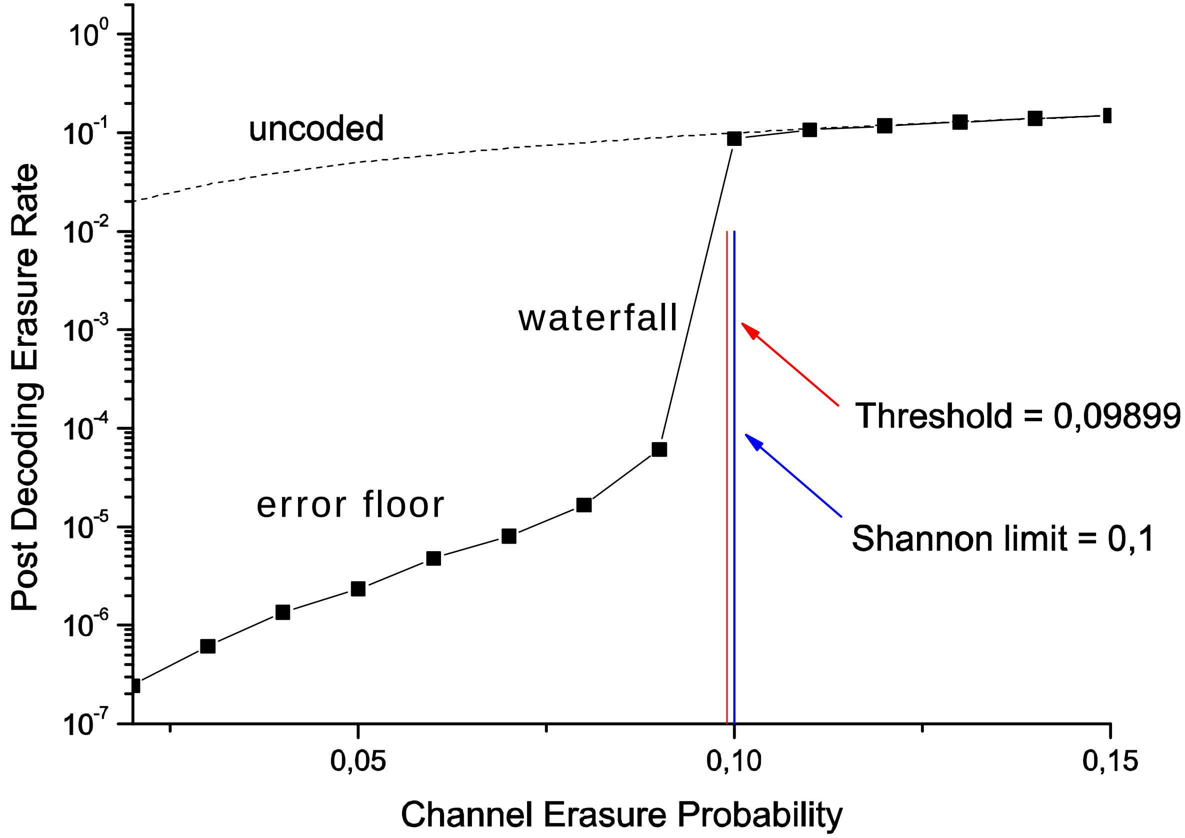
• From the channel coding theorem and from the expression of capacity for the BEC it follows.

$$q^* < 1 - R$$

for a LDPC code with code rate R.

• **Known result**: The iterative decoding of LDPC codes can achieve the BEC capacity (capacity achieving degree distributions).

Finite length performance of c.a. distributions



Heavy-tail / Poisson distribution (*)

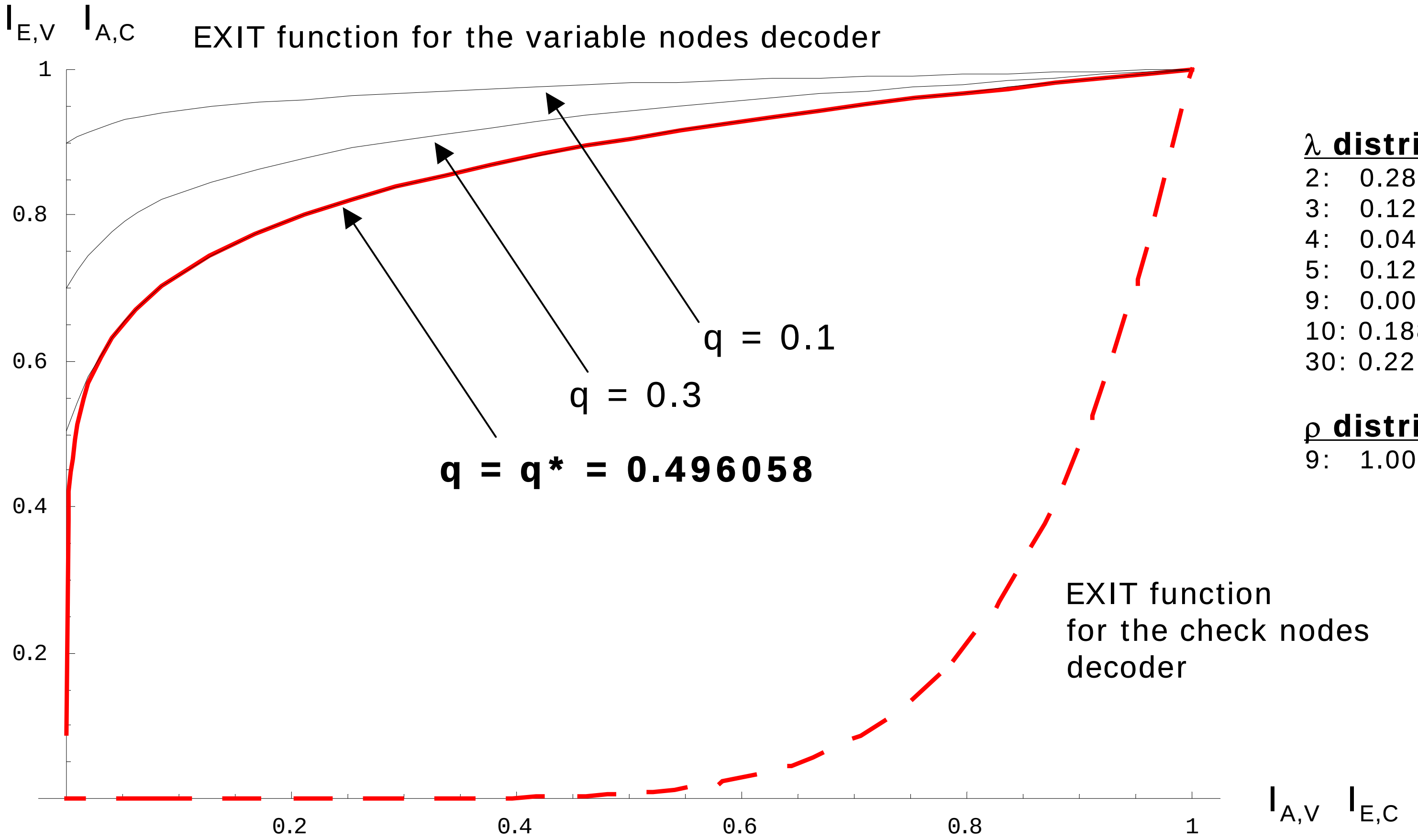
Random code

n = 100000
m = 10000

R = 0.9

(*) M. Luby, M. Mitzenmacher, M.A. Shokrollahi and D.A. Splielman, "Efficient Erasure Correcting Codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, February 2001, pages 569 - 584

Threshold analysis based on EXIT charts (*)



λ distribution

2: 0.287832
3: 0.126991
4: 0.048803
5: 0.126575
9: 0.000037
10: 0.188624
30: 0.221138

ρ distribution

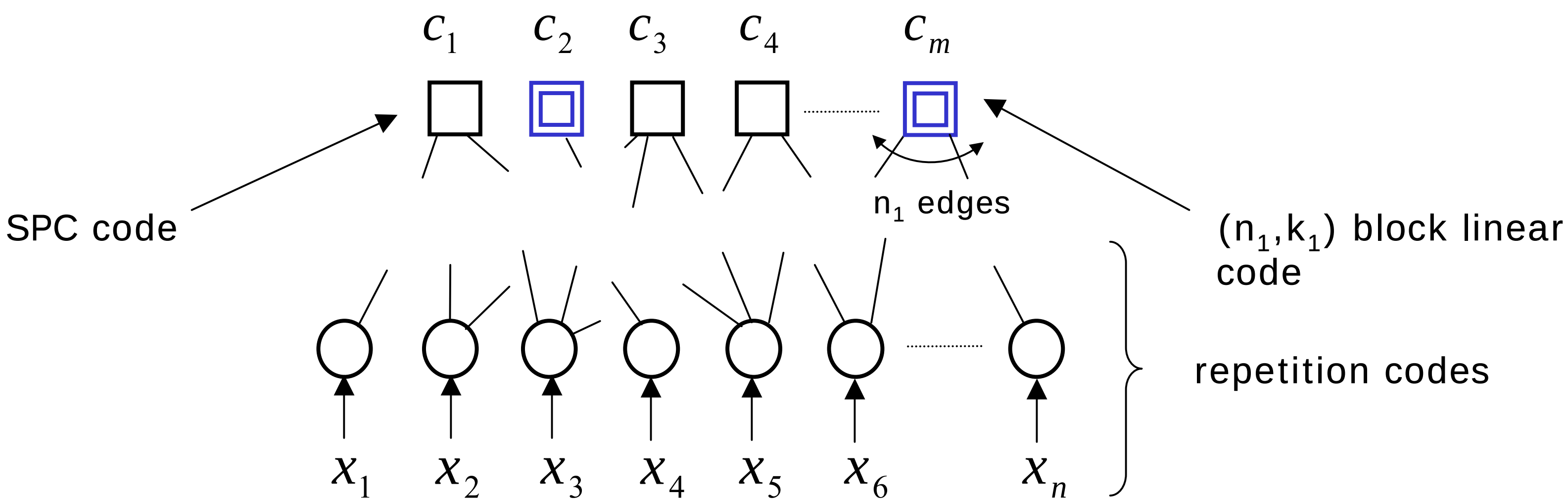
9: 1.000000

(*) A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: Model and erasure channel properties," *IEEE Trans. Inform. Theory*, vol. 50, pp. 2657–2673, Nov. 2004.

Outline Part-I:

- Background: iterative decoding of LDPC codes on the BEC
- **Generalized LDPC codes and Doubly-Generalized LDPC codes**
- Asymptotic analysis of D-GLDPC codes based on EXIT charts
- Results

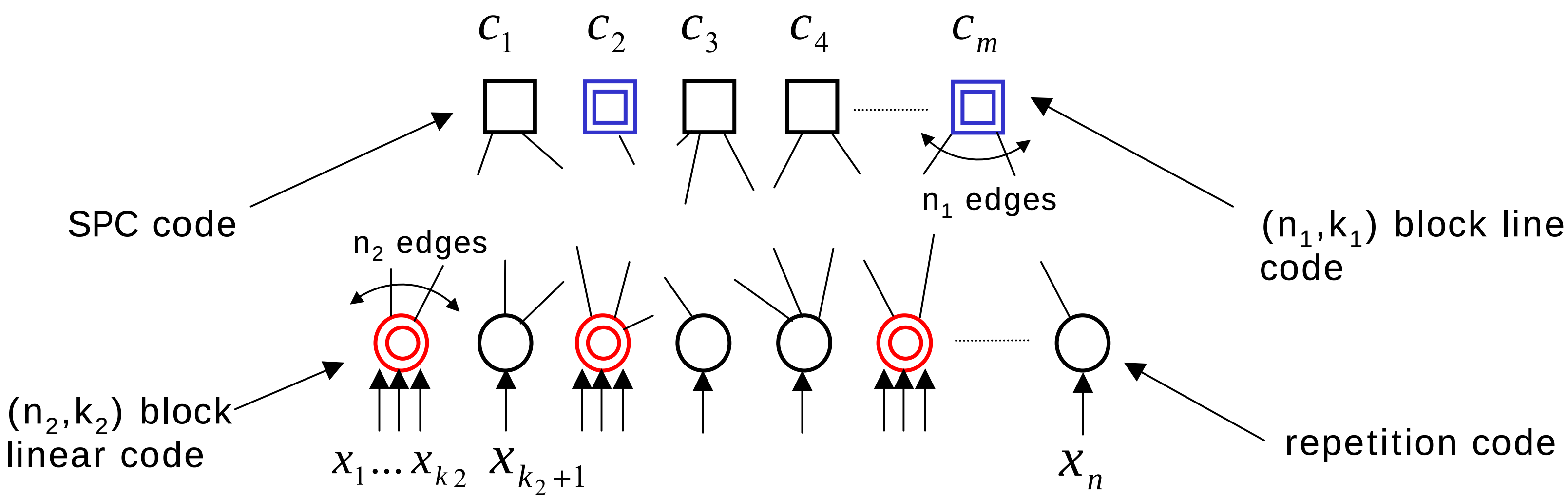
Generalized LDPC codes (GLDPC codes)



- Some check nodes are allowed to be (n, k) block linear codes different from SPC codes(*).
- Increased erasure correction capability at the generalized check nodes:
  - ➢ bounded distance decoding (correct up to d_{min} – 1 erasures);
  - ➢ Maximum a posteriori (MAP) decoding (most powerful decoding algorithms).
- "Naturally good" for low rate codes.

(*) R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, Sept. 1981.

Doubly-Generalized LDPC codes (D-GLDPC codes) (*)



- Besides the generalized check nodes, some variable nodes are allowed to be (n, k) block linear codes different from repetition codes.
- MAP decoding performed at both the generalized variable and check nodes.
- Possibility to design high rate codes.

(*) Y. Wang and M. Fossorier, "Doubly generalized low-density parity-check codes," in Proc. ISIT 2006, Seattle, USA, July 2006

Construction steps [1/2]

Step 1: row expansion

In every row of parity check matrix, each “1” is replaced with a subcolumn from the subcode parity check matrix of the corresponding super check node based on a one-to-one correspondence and each “0” is replaced with a zero subcolumn.

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

Subcode $\mathbf{H}_{(7,4)\text{Ham}} = \begin{pmatrix} 1&0&0&0&1&1&1 \\ 0&1&0&1&0&1&1 \\ 0&0&1&1&1&0&1 \end{pmatrix}$

$$\begin{matrix} 1&1&1&0&1&0&0&0&0&0&0&0&0&0 \\ 1&0&1&1&0&0&1&0&0&0&0&0&0&0 \\ 1&1&0&0&0&1&1&0&0&0&0&0&0&0 \end{matrix}$$

Construction steps[2/2]

Step 2: column expansion

In every column of parity check matrix each "1" in the same subcolumn is replaced with the same subrow in the transposed generator matrix of the corresponding super variable node based on a one-to-one correspondence and each "0" in a column vector is replaced with a zero row vector.

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

Subcode

$$\mathbf{G}_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{G}_1^T = \begin{pmatrix} 10 \\ 11 \\ 01 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \xrightarrow{\text{row expansion}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \xrightarrow{\text{column expansion}} \begin{pmatrix} 10 \\ 10 \\ 10 \\ 00 \\ 00 \\ 00 \\ 00 \\ 11 \\ 11 \\ 00 \\ 00 \\ 00 \\ 00 \\ 00 \\ 00 \\ 01 \\ 00 \\ 00 \end{pmatrix}$$

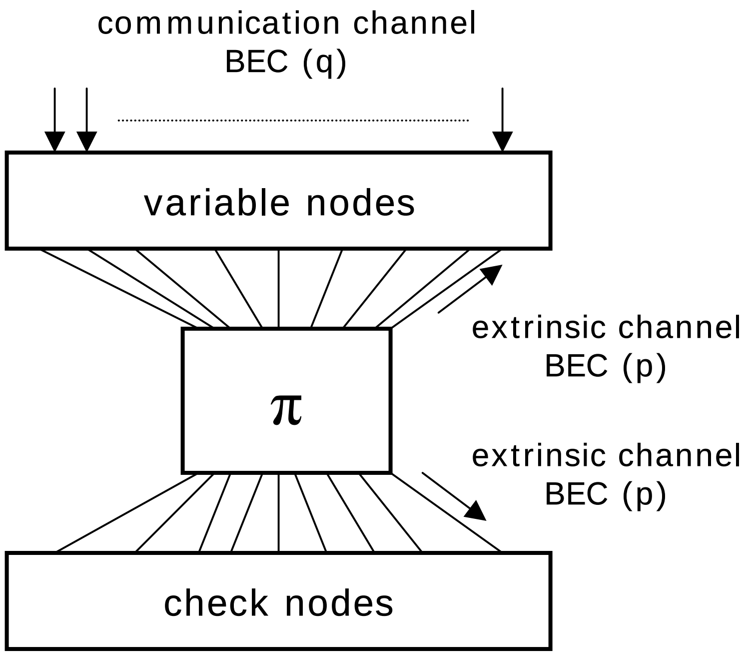
Outline Part-I:

- Background: iterative decoding of LDPC codes on the BEC
- Generalized LDPC codes and Doubly-Generalized LDPC codes
- **Asymptotic analysis of D-GLDPC codes based on EXIT charts**
- Results

EXIT Functions for the variable and check nodes decoders of a D-GLDPC code

$$I_{E,V}(p,q) = \sum_{i=1}^{\mathcal{I}_V} \lambda_i \, I_{E,V}^{(i)}(p,q)$$

$$I_{E,C}(p) = \sum_{i=1}^{\mathcal{I}_C} \rho_i \, I_{E,C}^{(i)}(p)$$



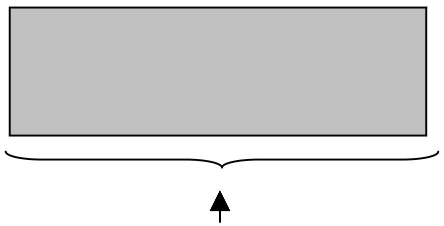
- In order to evaluate the EXIT functions for the variable node decoder and for the check nodes decoder, the knowledge of the EXIT function for all the variable nodes and check nodes types is needed.

EXIT Function of any check node of a D-GLDPC code on the BEC

$$I_E(p) = 1 - \frac{1}{n} \sum_{t=0}^{n-1} (1-p)^{n-t-1} p^t \cdot [(n-t)\,\tilde{e}_{n-t} - (t+1)\,\tilde{e}_{n-t-1}]$$

Information functions (*)

(k x n) Generator Matrix



choose g columns

- e_g: summation of the ranks of all the possible submatrices obtained choosing g columns in **G**.
- Independency on the code representation.

(*) T. Helleseth, T. Kløve and V.I. Levenshtein,"On the information function of an error correcting code," *IEEE Trans. Inform.Theory*, vol. 43, pp. 549–557, Mar. 1997.

Bounded distance EXIT Function of any check node of a D-GLDPC code on the BEC (*)

Decoding strategy: if the number of erasures incoming from the extrinsic channel (i.e. from the variable nodes) is lower than or equal to *d*, perform MAP decoding. Otherwise declare a decoding failure.

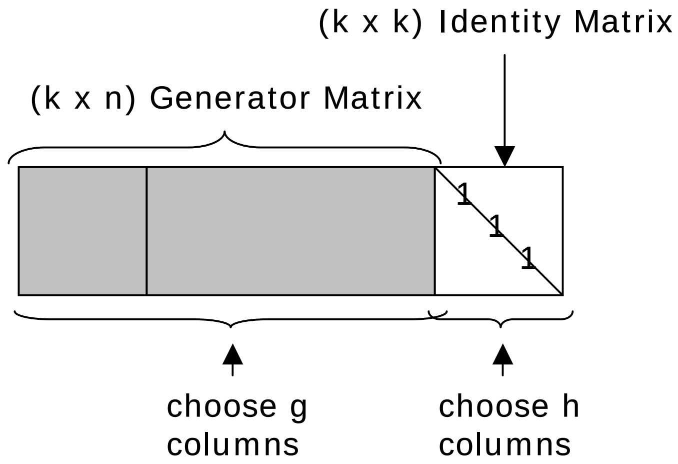
$$I_E(p) = 1 - \frac{1}{n}\sum_{t=0}^{d-1}(1-p)^{n-t-1}p^t[(n-t)\tilde{e}_{n-t} - (t+1)\tilde{e}_{n-t-1}] - \sum_{t=d}^{n-1}(1-p)^{n-t-1}p^t\binom{n-1}{t}$$

(*) E. Paolini, M. Fossorier and M. Chiani, "Analysis of generalized LDPC codes with random component codes for the binary erasure channel," in Proc. ISITA 2006, Seoul, Korea, Oct. 2006.

EXIT Function of any variable node of a D-GLDPC code on the BEC (*)

$$I_E(p,q) = 1 - \frac{1}{n} \sum_{t=0}^{n-1} \sum_{z=0}^{k} p^t (1-p)^{n-1-t} q^z (1-q)^{k-z} \cdot \left[(n-t)\tilde{e}_{n-t,k-z} - (t+1)\tilde{e}_{n-t-1,k-z} \right]$$

Split information functions (*)



- $e_{g,h}$: summation of the ranks of all the possible submatrices obtained choosing g columns in **G** and h columns in the identity matrix
- Dependency on the code representation

(*) A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: Model and erasure channel properties," *IEEE Trans. Inform. Theory*, vol. 50, pp. 2657–2673, Nov. 2004.

The Random Codes Approach

• **<u>Problem</u>**: both the split information functions (for the variable nodes) and the information functions (for the check nodes) are unknown for most block linear codes (e.g. for BCH codes, Reed-Muller codes,…).

• Direct computation usually not possible (huge computation time). EXAMPLE: direct approach not possible for a k=10, n=31 code.

• **<u>Proposed approach</u>**: instead of considering specific component codes, consider ***random*** codes and evaluate the ***<u>expected EXIT function</u>*** for the variable and for the check nodes decoder of the overall D-GLDPC code.

• In order to solve this problem, it is sufficient to develop formulas to evaluate the *<u>expected split information functions</u>* for the variable nodes and the *<u>expected information functions</u>* for the check nodes (see previous formulas).

The Main Problem of the Random Codes Approach

• **Problem**: a correct application of the EXIT charts approach requires the following conditions (see examples in next 2 slides):

for any variable node: $\lim_{p \to 0} E[I_E(p,q)] = 1 \quad \forall\, q.$

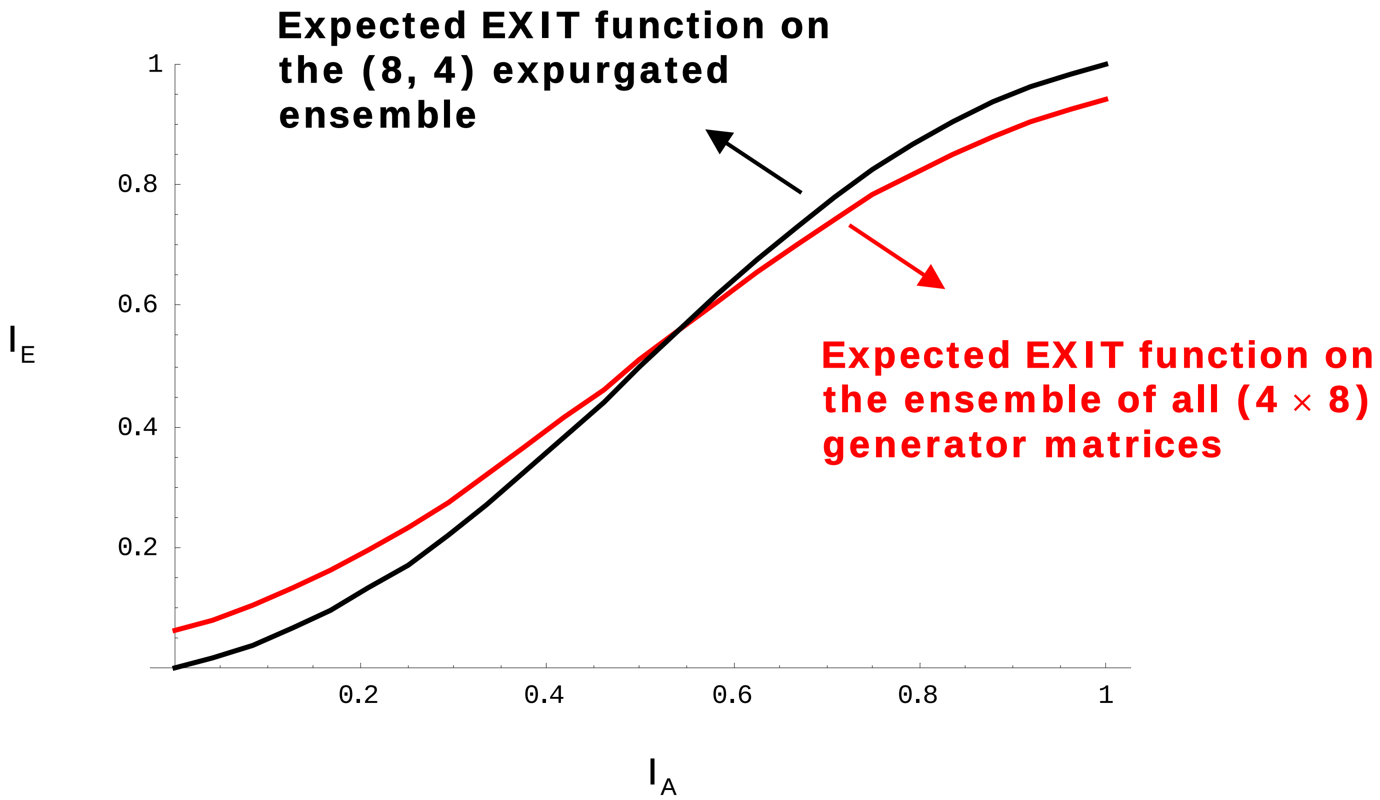
for any check node: $\lim_{p \to 0} E[I_E(p)] = 1$

$\lim_{p \to 1} E[I_E(p)] = 0$

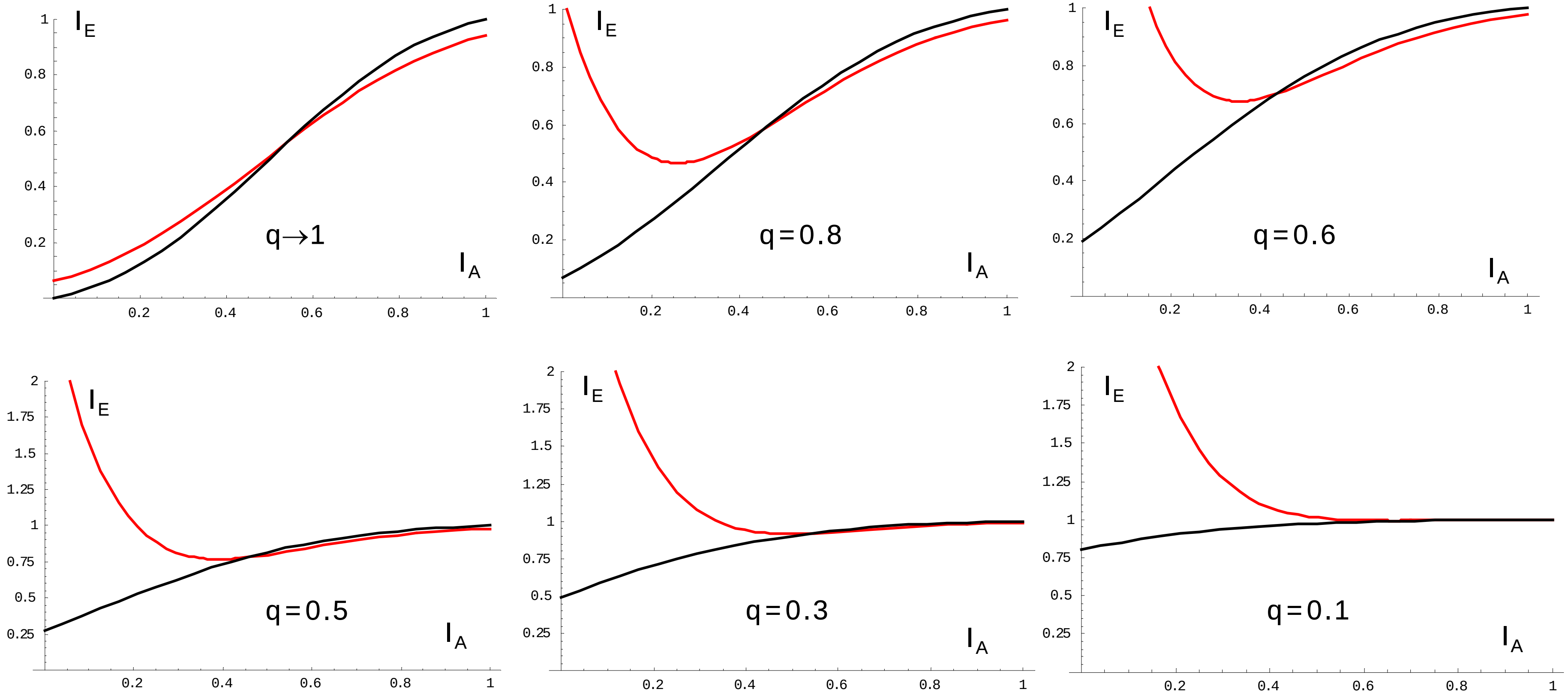
• This condition is in general **not** satisfied if the generator matrix for the (n,k) variable or check node is randomly chosen from the ensemble of all binary ($k \times n$) matrices.

• Necessity to expurgate the ensemble of generator matrices.

Example: n=8, k=4 check nodes



Example: n=8, k=4 variable nodes



Expected Information Functions evaluation on the expurgated ensemble [1/3] (*)

$$E_{\mathcal{C}_*^{(n,k)}}[\tilde{e}_g] = \binom{n}{g} E_{\mathcal{C}_*^{(n,k)}}[\text{rank}(\mathcal{S}_g)]$$

$$E_{\mathcal{C}_*^{(n,k)}}[\text{rank}(\mathcal{S}_g)] = \sum_{u=1}^{\min\{k,g\}} u \frac{K(k,n,g,u,k)}{J(k,n,k)}$$

(k x n) Generator Matrix

S_g : submatrix obtained by choosing any g columns

Number of (k × n) binary matrices:
1- without all-zero columns;
2- full rank;
3- without independent columns.

Number of (k × n) binary matrices:
1- without all-zero columns;
2- full rank;
3- without independent columns;
4- such that the first g columns have rank u.

(*) E. Paolini, M. Fossorier and M. Chiani, "Analysis of generalized LDPC codes with random component codes for the binary erasure channel," in Proc. ISITA 2006, Seoul, Korea, October 2006.

Expected Information Functions evaluation on the expurgated ensemble [2/3]

• It turns out that the expected information functions (with expectation on the expurgated ensemble) can be evaluated by solving the following problem:

Evaluate the exact number of (m x n) binary matrices with these properties:

1- rank = m (full rank)

2- no all-zero columns

3- rank of the submatrix composed of the first g columns = u

4- no independent columns (removing any column from the matrix, the rank doesn't change).

Found solution to this problem in the general case where the (m x n) matrix, $m < n$, has rank $r \leq m$ (matrix not necessarily full rank).

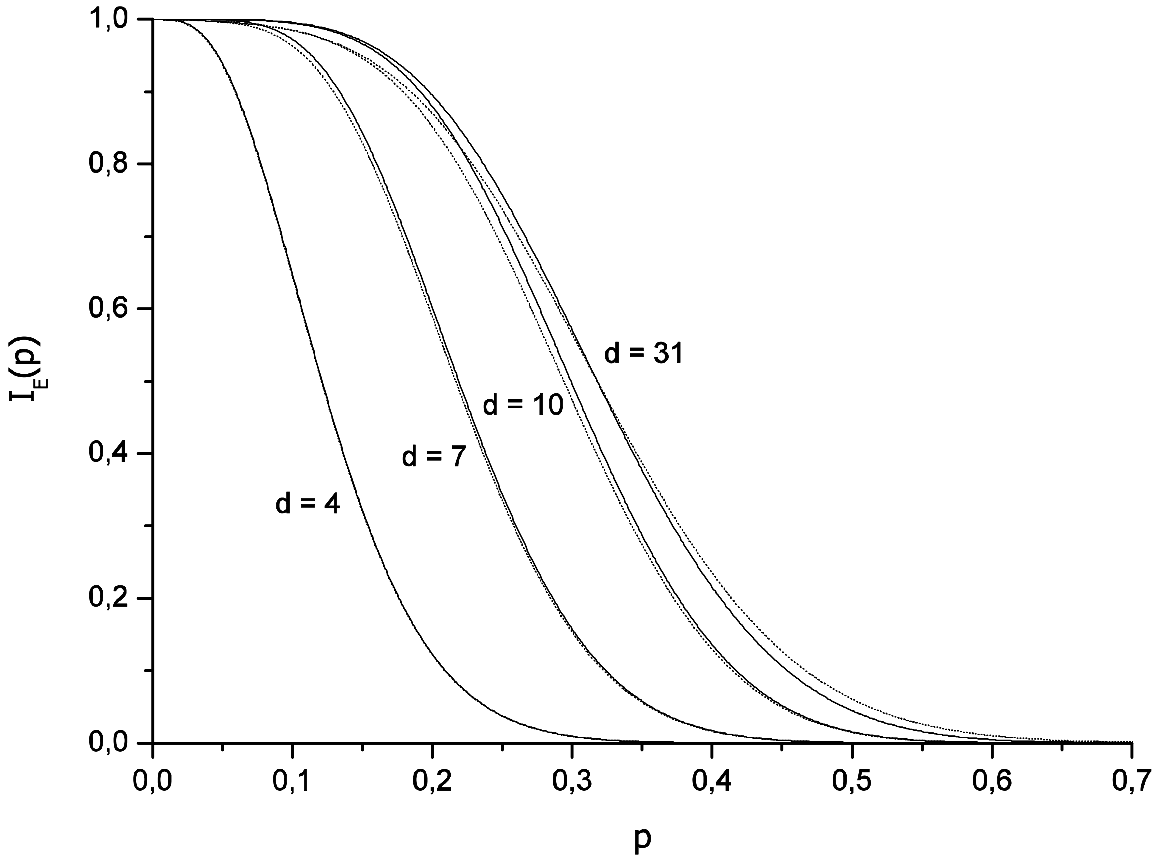
Expected Information Functions evaluation on the expurgated ensemble [3/3]

$$J(m,n,r) = F(m,n,r) - \sum_{j=1}^{r-1} \binom{n}{j} \left[\prod_{i=0}^{j-1} (2^m - 2^i) \right] \cdot 2^{j(r-j)} J(m-j, n-j, r-j)$$

Found recursive formulas for exactly evaluating the function J() and the function K()

$$K(m,n,g,u,r) = M(m,n,g,u,r) - \sum_{j=1}^{r-1} \sum_{l=0}^{\min\{u,j\}} \binom{g}{l} \binom{n-g}{j-l} \left[\prod_{i=0}^{j-1} (2^m - 2^i) \right] 2^{j(r-j)} \cdot K(m-j, n-j, g-l, u-l, r-j)$$

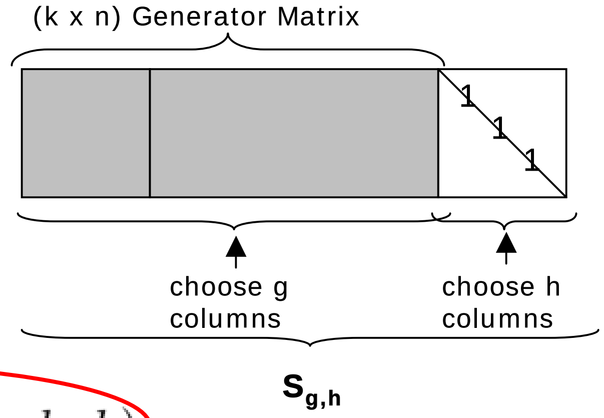
Example: (31,21)-BCH check node VS expurgated ensemble average under bounded distance decoding



Solid: (31,21) narrowsense BCH code

Dotted: analytical expectation on the (31,21) expurgated ensemble

Expected split Information Functions evaluation on the expurgated ensemble [1/3]



$$\mathbb{E}_{\mathcal{G}_*^{(n,k)}}[\tilde{e}_{g,h}] = \binom{n}{g}\binom{k}{h}\mathbb{E}_{\mathcal{G}_*^{(n,k)}}[\mathrm{rank}(\mathcal{S}_{g,h})]$$

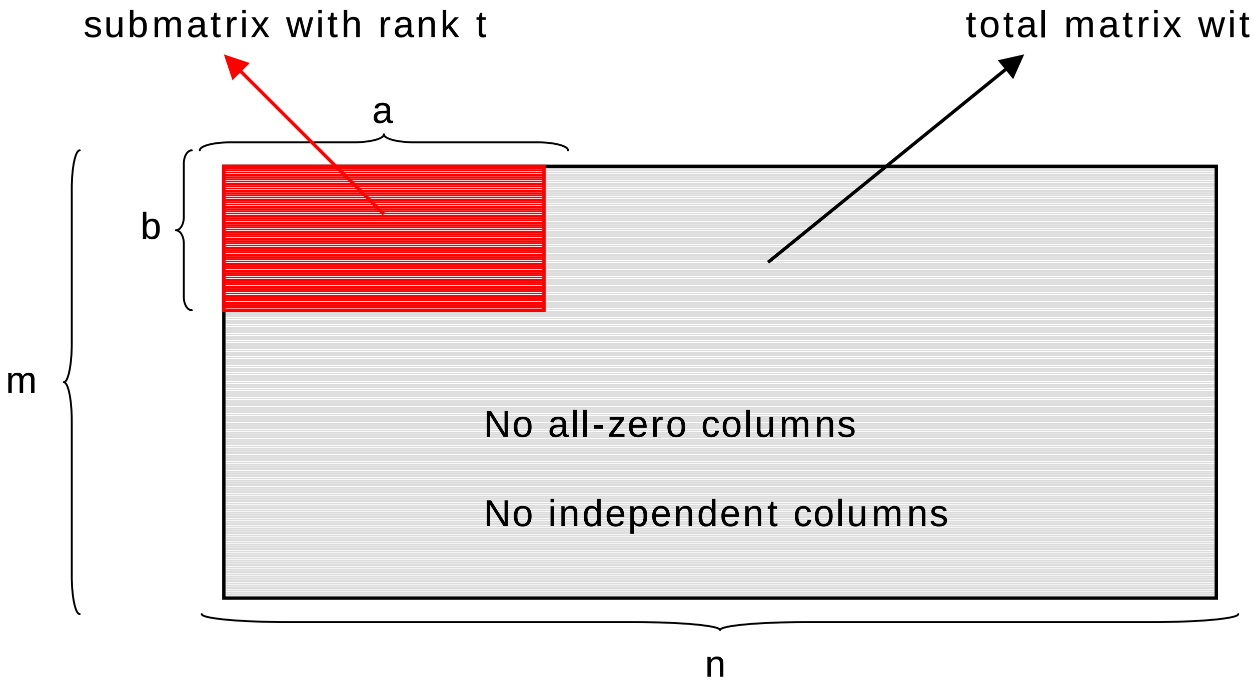
$$\mathbb{E}_{\mathcal{G}_*^{(n,k)}}[\mathrm{rank}(\mathcal{S}_{g,h})] = \sum_{u=h}^{\min\{k,g+h\}} u \frac{\widetilde{K}(k,n,g,k-h,u-h,k)}{J(k,n,k)}$$

Number of (k × n) binary matrices:
1- without all-zero columns;
2- full rank;
3- without independent columns.

Number of (k × n) binary matrices:
1- without all-zero columns;
2- full rank;
3- without independent columns;
4- such that the submatrix composed of the first g columns and the first k-h rows has rank u-h.

Expected split Information Functions evaluation on the expurgated ensemble [2/3]

$\widetilde{K}(m,n,a,b,t,r)$: total number of matrices with the following properties



Expected split Information Functions evaluation on the expurgated ensemble [3/3]

• **Definition** $\hat{K}(m,n,a,b,t,u,r)$ = number of (m x n) binary matrices with the following properties:

1- rank = r

2- no zero columns

3- rank of the submatrix intersection of first b rows and of first a columns = t

4- rank of the submatrix composed of the first a columns = u

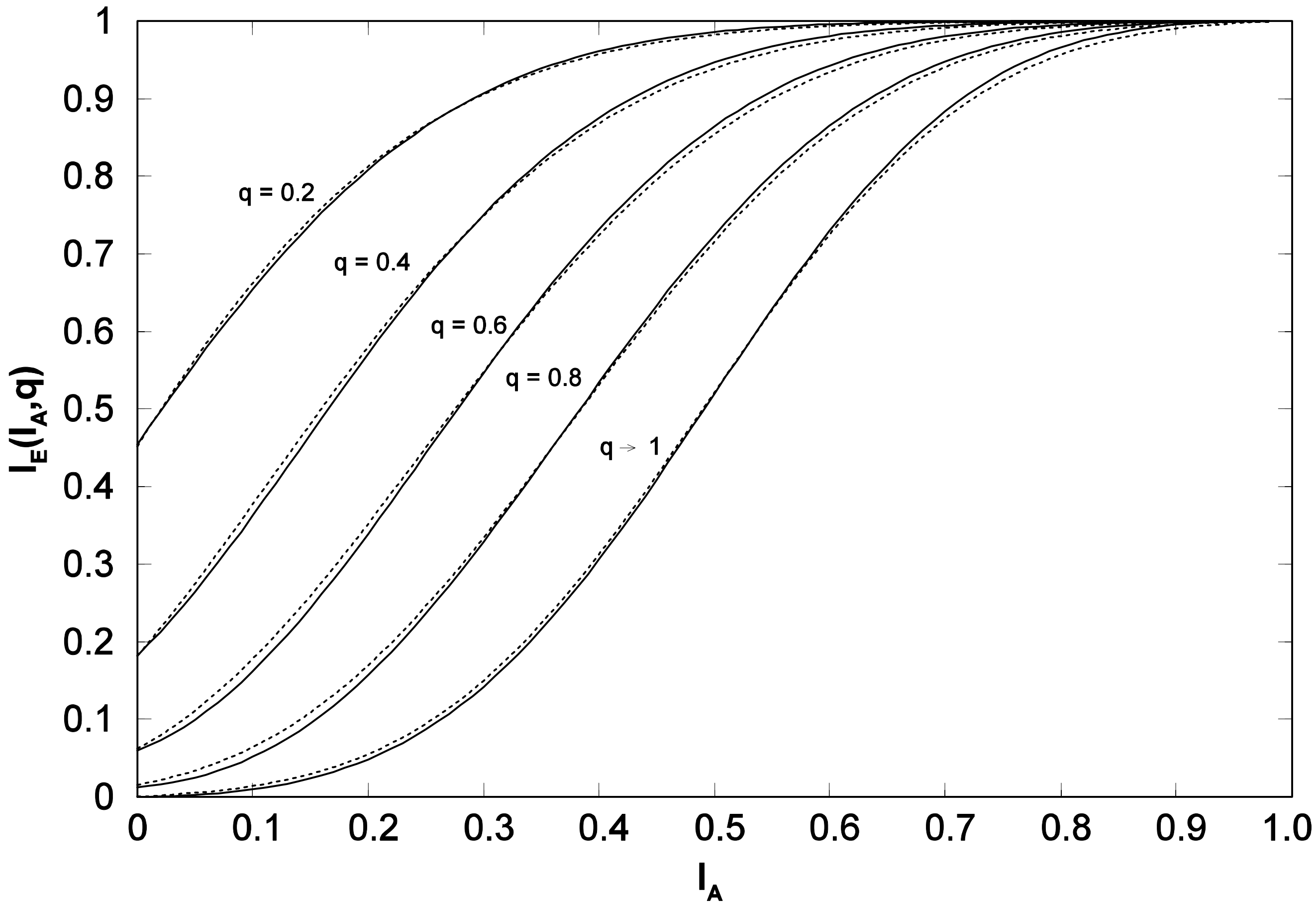
5- no independent columns (removing any column from the matrix, the rank doesn't change).

• It results:

$$\widetilde{K}(m,n,a,b,t,r) = \sum_{u} \hat{K}(m,n,a,b,t,u,r)$$

• Found a formula for evaluating $\hat{K}(m,n,a,b,t,u,r)$

Example: Random (16,8) variable node VS expurgated ensemble average



Dotted: analytical expectation on the (16,8) expurgated ensemble

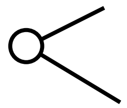
Solid: specific (16,8) random code

VERY GOOD MATCH EVEN FOR SHORT CODES

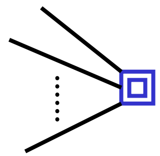
Outline Part-I:

- Background: iterative decoding of LDPC codes on the BEC
- Generalized LDPC codes and Doubly-Generalized LDPC codes
- Asymptotic analysis of D-GLDPC codes based on EXIT charts
- **Results**

GLDPC codes with uniform check node structure [1/2]



Variable nodes with degree 2



(31,21) linear block codes
[MAP decoding up to d erasures]

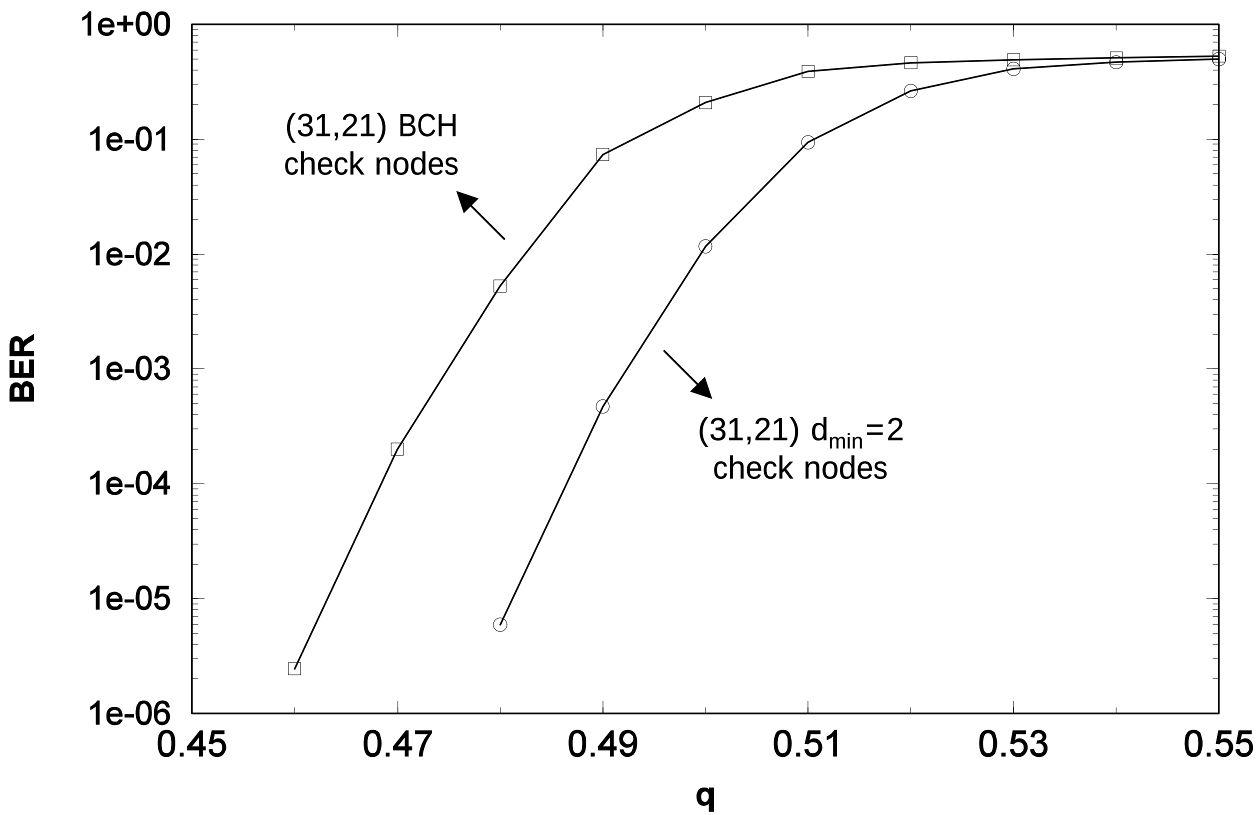
R = 0.355

Thresholds (found via EXIT chart)

d	BCH	expectation
4	0.21915	0.21879
7	0.35596	0.35407
10	0.46256	0.45929
31	0.50187	0.51426

- For unconstrained MAP decoding, the ensemble average outperforms the BCH check nodes.
- Better codes must exist within the expurgated ensemble.
- **Better to use weak component codes (e.g. d_{min}=2) instead of the BCH codes if the GLDPC code has a uniform check node decoder!**

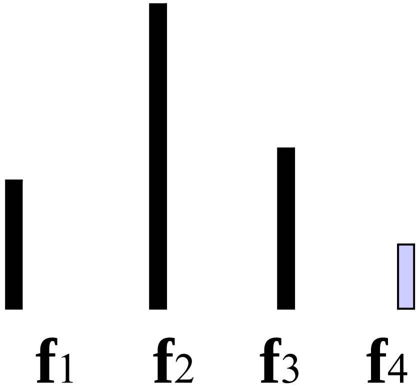
GLDPC codes with uniform check node structure [2/2]

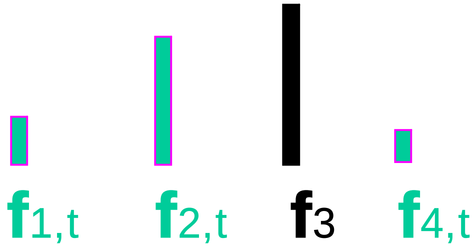


- The same bipartite graph, only the check nodes are different
- N=3999 bits, R=0.355
- This result confirms the asymptotic investigation based on the ensemble average

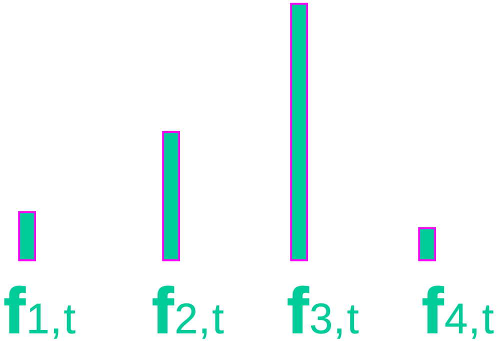
Differential Evolution

Step 1: initialization
Step 2: mutation and test
Step 3: compare and update.
Step 4: stopping test





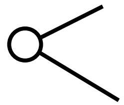
$$\mathbf{f}_{k,t} = \mathbf{f}_4 + r \times (\mathbf{f}_\mathrm{i} + \mathbf{f}_\mathrm{j})$$



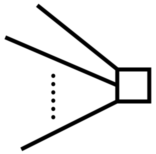
next iteration

Optimized LDPC distribution

Design rate: R = 0.5



Repetition codes with degrees from 2 to 30



SPC codes with degrees from 5 to 14

Variable Distribution

2: 0.281884
3: 0.123242
4: 0.060701
5: 0.106412
9: 0.084976
10: 0.103547
30: 0.239238

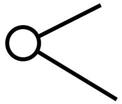
Check Distribution

8: 0.925027
10: 0.074973

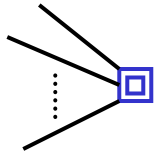
Threshold: **q* = 0.49611**

Optimized GLDPC distribution with (31,21) BCH check nodes

Design rate: R = 0.5



Repetition codes with degrees from 2 to 30



- SPC codes with degrees from 5 to 14
- (31,21)-BCH [assume MAP decoding]

Variable Distribution

2:	0.270712
3:	0.168858
5:	0.165958
8:	0.230227
30:	0.164246

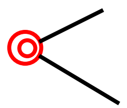
Check Distribution

9:	0.912838
(31,21) BCH:	**0.087162**

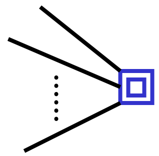
Threshold: **q* = 0.49671**

Optimized D-GLDPC distribution with (31,21) BCH check nodes and (31,10) variable nodes [D-GLDPC$_1$]

Design rate: R = 0.5



- Repetition codes with degrees from 2 to 30
- (31,10) ensemble average [assume MAP decoding]



- SPC codes with degrees from 5 to 14
- (31,21)-BCH [assume MAP decoding]

Variable Distribution

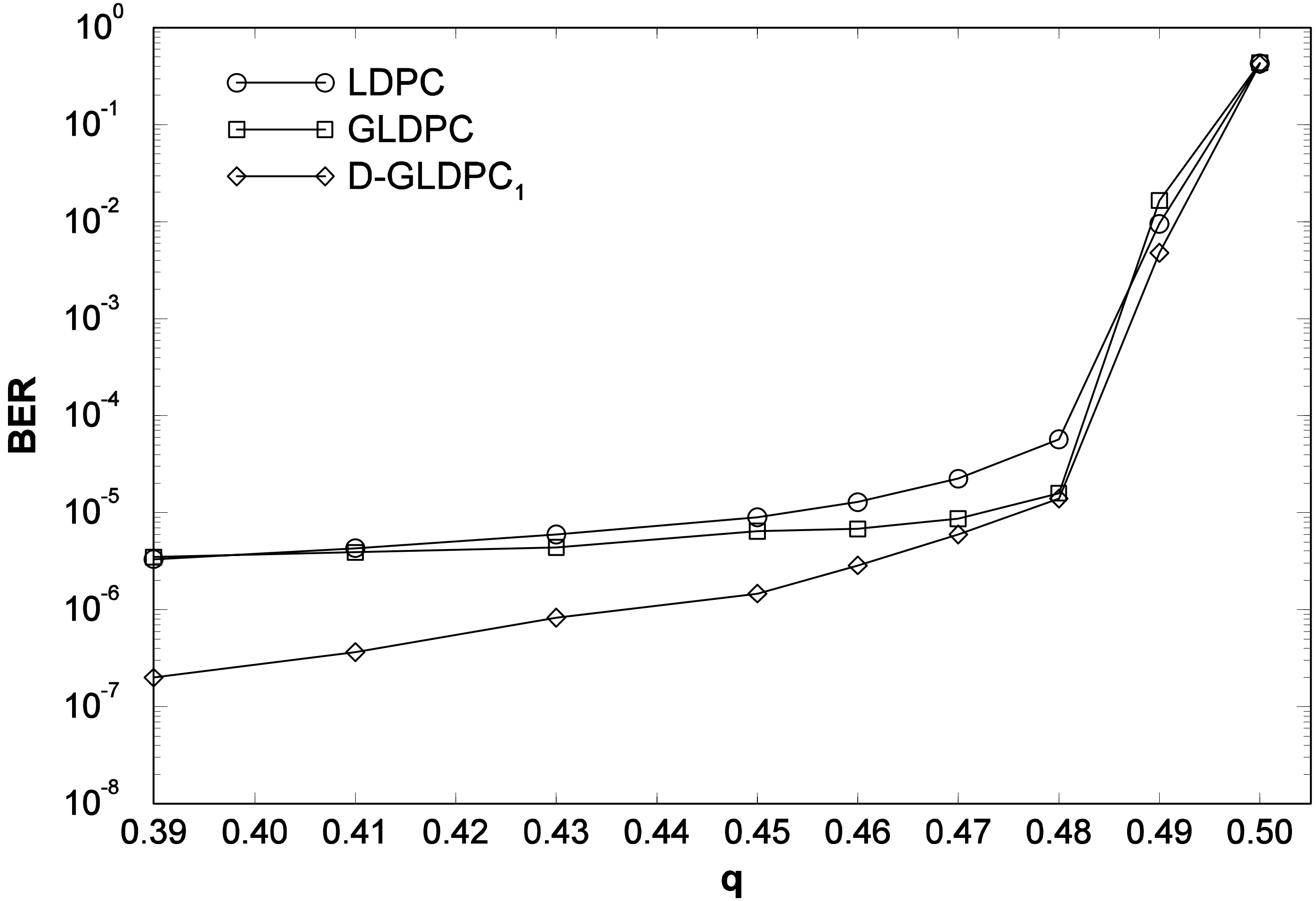
2:	0.287410
3:	0.161606
5:	0.293870
29:	0.217547
(31,10) avg:	**0.039568**

Check Distribution

9:	0.871398
(31,21) BCH:	**0.128602**

Threshold: **q* = 0.49759**

Random codes performance comparison



Discussion [1/2]

• The error floor of random and capacity-approaching LDPC codes is due to poor minimum distance (*).

• The random and capacity approaching D-GLDPC code has the best threshold and the best error floor.

• Typically, the addition of generalized check nodes is beneficial in terms of minimum distance, but its drawback is a lowering of the code rate (rate-loss), which reveals unacceptable in many cases.

• The introduction of (weaker) generalized variable nodes permits to face the rate-loss, thus enabling for using a larger number of generalized check nodes.

(*) C. Di, R. Urbanke and T. Richardson, "Weight distributions: How deviant can you be?" in *Proc. of IEEE ISIT 2001*, Washington, DC, June 2001.

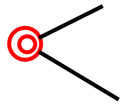
Discussion [2/2]

• In order to generate D-GLDPC codes with error floor better than that of the D-GLDPC$_1$ code, and still a good threshold, the following approach has been adopted:

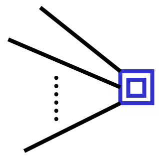
> "Look for the degree distribution with the best threshold under the further constraint of a lower bound on the fraction of edges towards the generalized check nodes"

Optimized D-GLDPC distribution with (31,21) BCH check nodes and (31,10) variable nodes with further constraint [D-GLDPC$_2$]

Design rate: R = 0.5



- Repetition codes with degrees from 2 to 30
- (31,10) ensemble average [assume MAP decoding]



- SPC codes with degrees from 5 to 14
- (31,21)-BCH [assume MAP decoding] – **minimum required fraction of edges towards the generalized check nodes: 16%**

Variable Distribution

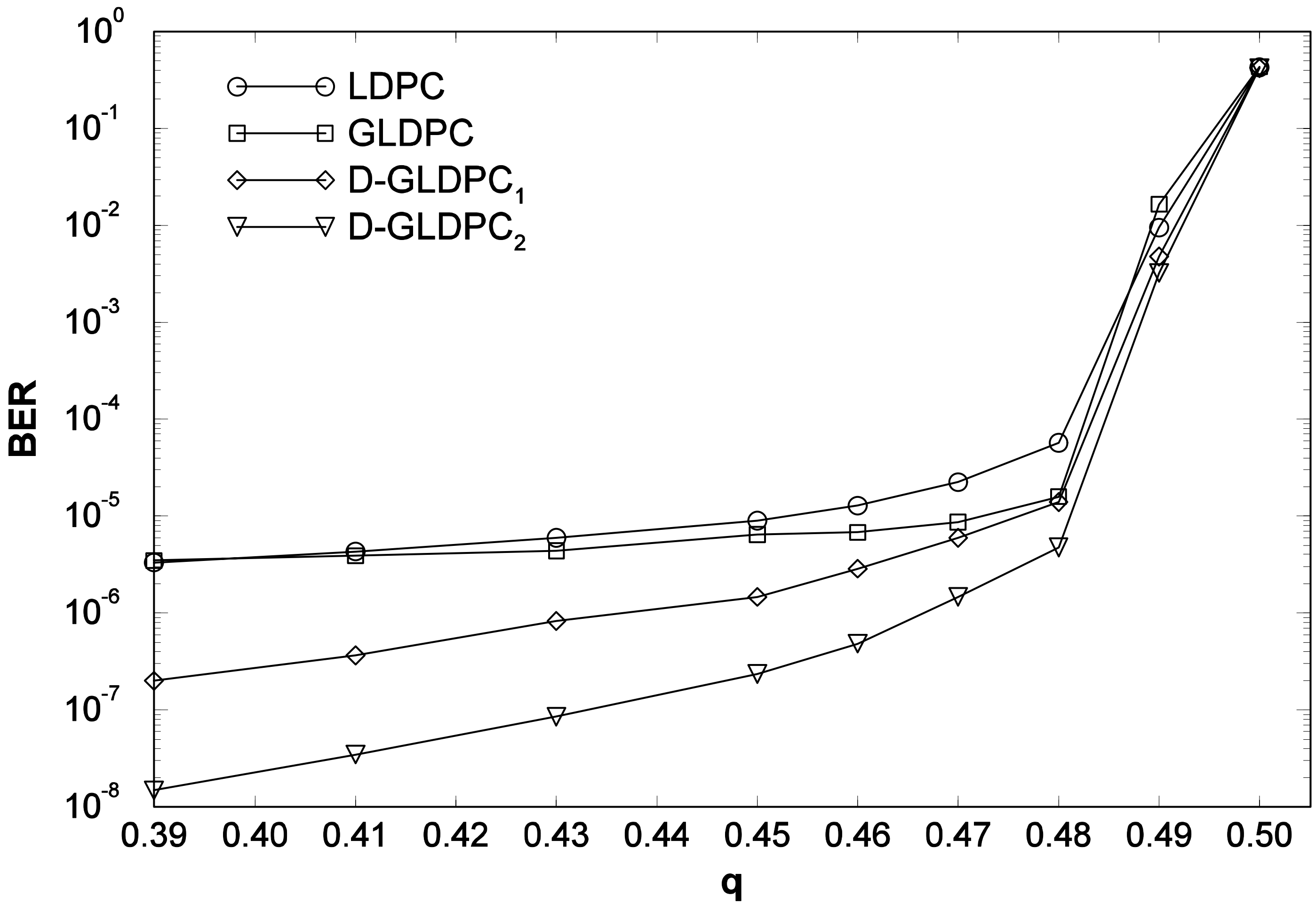
2:	0.275116
3:	0.280377
5:	0.358294
30:	0.174163
(31,10) avg:	**0.030803**

Check Distribution

9:	0.381799
10:	0.458201
(31,21) BCH:	**0.160000**

Threshold: **q* = 0.49655**

Random codes performance comparison



Conclusion

- A technique for the asymptotic analysis of D-GLDPC codes on the binary erasure channel has been presented.

- Useful and effective for the threshold analysis of D-GLDPC codes for which the (split) information functions of the component codes are unknown.

- The technique can be combined with optimization tools (e.g. differential evolution) in order to generate optimal D-GLDPC degree distributions.

- Our results reveal that D-GLDPC represent a promising solution for near-capacity iterative decoding on the BEC (competitive threshold, lower error floor with respect to standard LDPC codes).

- Numerical results presented for long, capacity-approaching, random codes.

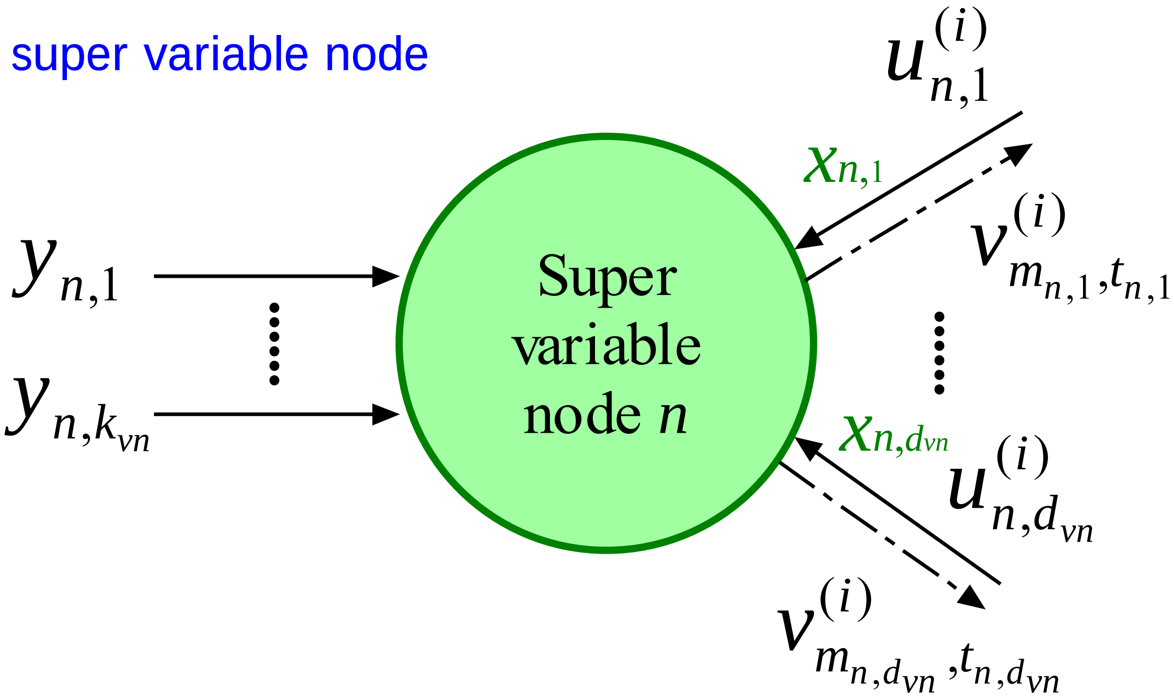
Part-II: AWGN Channel

Outline Part-II:

- **Decoding algorithm for AWGN channel**
- Asymptotic analysis of D-GLDPC codes based on EXIT charts
- Results

Iterative decoding of D-GLDPC codes over AWGN Channel [1/4]

For super variable node



$$V_{m_{n,p},t_{n,p}}^{(i)} = \log \frac{P(x_{n,p} = 0 \mid \mathrm{u}_{n[p]}^{(i)}, \mathrm{y}_n)}{P(x_{n,p} = 1 \mid \mathrm{u}_{n[p]}^{(i)}, \mathrm{y}_n)}$$

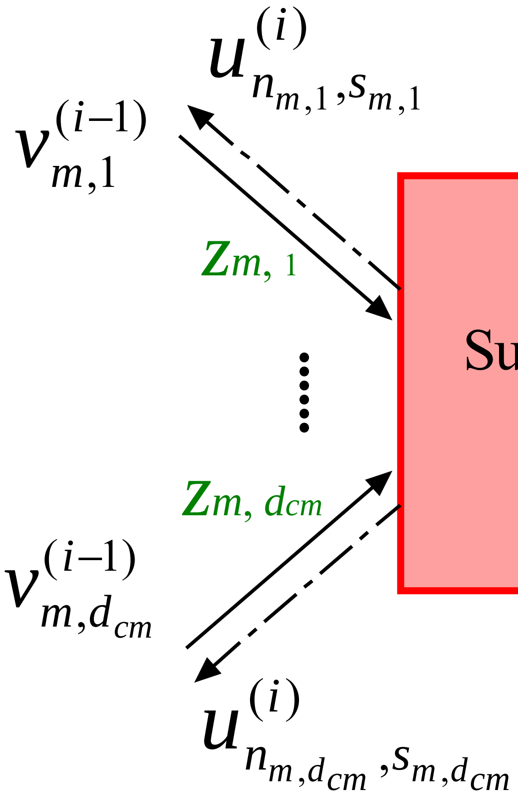
Iterative decoding of D-GLDPC codes over AWGN Channel [2/4]

$$V^{(i)}_{m_{n,p},t_{n,p}} = \log \frac{\mathrm{P}(x_{n,p}=0 \mid \mathrm{u}^{(i)}_{n[p]}, \mathrm{y}_n)}{\mathrm{P}(x_{n,p}=1 \mid \mathrm{u}^{(i)}_{n[p]}, \mathrm{y}_n)}$$

$$V^{(i)}_{m_{n,p},t_{n,p}} = \log \frac{\sum\limits_{\mathbf{b}_n : x_{n,p}=0} \prod\limits_{j=1,\ j\neq p}^{d_{vn}} \mathrm{e}^{-U^{(i)}_{n,j} x_{n,j}} \prod\limits_{j=1}^{k_{vn}} \mathrm{e}^{\frac{2 y_{n,j} c_{n,j}}{N_0}}}{\sum\limits_{\mathbf{b}_n : x_{n,p}=1} \prod\limits_{j=1,\ j\neq p}^{d_{vn}} \mathrm{e}^{-U^{(i)}_{n,j} x_{n,j}} \prod\limits_{j=1}^{k_{vn}} \mathrm{e}^{\frac{2 y_{n,j} c_{n,j}}{N_0}}}$$

Iterative decoding of DGLDPC codes over AWGN channel [3/4]

For super check node



$$U^{(i)}_{n_{m,q},s_{m,q}} = \log \frac{P(z_{m,q}=0 \mid v^{(i-1)}_{m[q]})}{P(z_{m,q}=1 \mid v^{(i-1)}_{m[q]})}$$

Iterative decoding of DGLDPC codes over AWGN channel [4/4]

$$U^{(i)}_{n_{m,q},s_{m,q}} = \log \frac{\mathrm{P}(z_{m,q}=0 \mid \mathrm{v}^{(i-1)}_{m[q]})}{\mathrm{P}(z_{m,q}=1 \mid \mathrm{v}^{(i-1)}_{m[q]})}$$

$$U^{(i)}_{n_{m,q},s_{m,q}} = \log \frac{\displaystyle\sum_{\mathbf{z}_m : z_{m,q}=0} \prod_{j=1, j\neq q}^{d_{cm}} \mathrm{e}^{-V^{(i-1)}_{m,j} z_{m,j}}}{\displaystyle\sum_{\mathbf{z}_m : z_{m,q}=1} \prod_{j=1, j\neq q}^{d_{cm}} \mathrm{e}^{-V^{(i-1)}_{m,j} z_{m,j}}}$$

Outline Part-II:

- Decoding algorithm for AWGN channel
- **Asymptotic analysis of D-GLDPC codes based on EXIT charts**
- Results

Analysis by EXIT charts [1/2]

- Assume *D* and *E* different subcodes in variable nodes and check nodes, respectively.

Average variable node transfer curve is

$$I_V(I_U, \frac{E_b}{N_0}, R) = \sum_{i=1}^{D} \lambda_i \cdot I_{V_i}(I_U, \frac{E_b}{N_0}, R)$$

Average check node transfer curve is

$$I_U(I_V) = \sum_{i=1}^{E} \rho_i \cdot I_{U_i}(I_V)$$

Analysis by EXIT charts [2/2]

Since the curve-fitting technique works well for BPSK signaling over an AWGN channel, we constructed three doubly-GLDPC codes, $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$, using this technique.

Outline Part-II:

- Decoding algorithm for AWGN channel
- Asymptotic analysis of D-GLDPC codes based on EXIT charts
- **Results**

Construction of DGLDPC code C_1

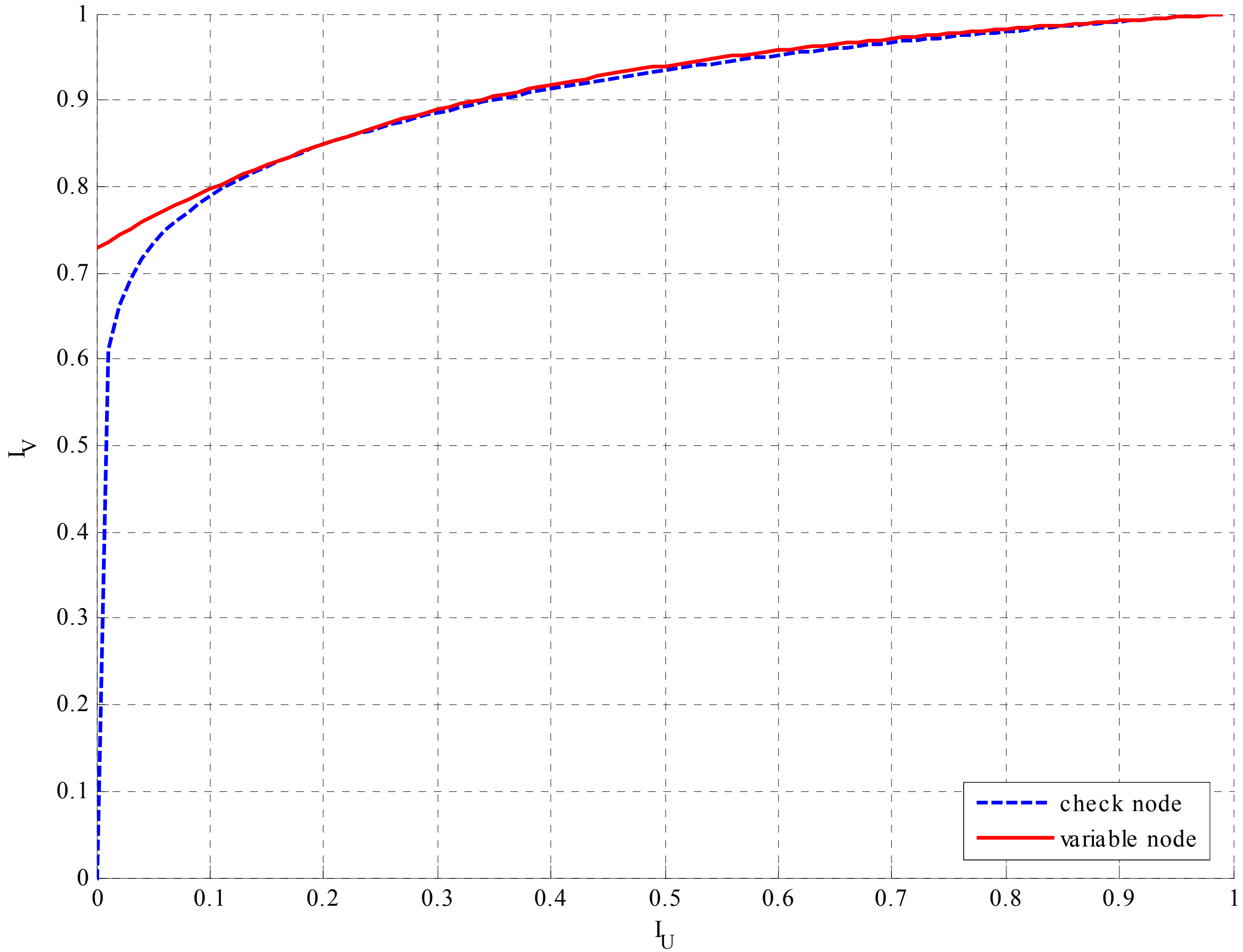
- C_1 is a rate-3/4 length-100000 code.
- Super variable nodes:

  (6,1) repetition code, (6,3) code with generator matrix $\begin{pmatrix} 100101 \\ 010011 \\ 001110 \end{pmatrix}$,

  (6,4) code with generator matrix $\begin{pmatrix} 111000 \\ 011100 \\ 001110 \\ 000111 \end{pmatrix}$, (6,5) SPC code.

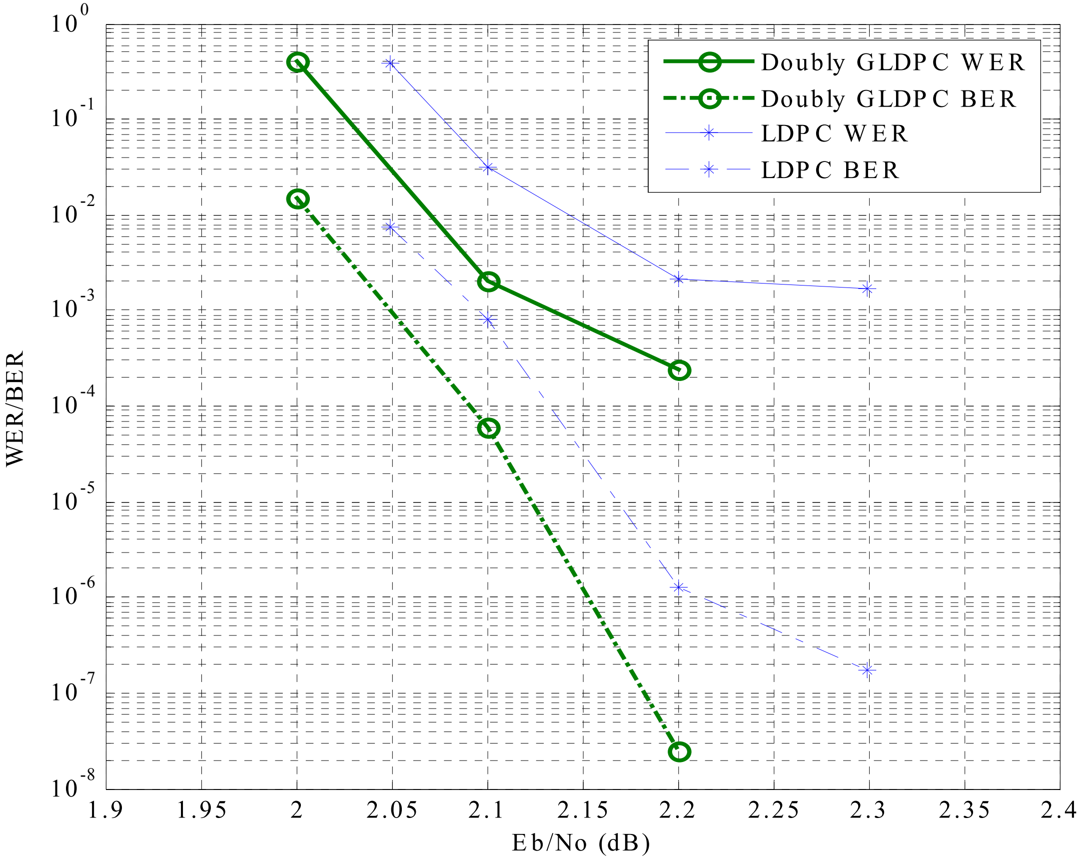
- Super check node: (12,11) SPC code
- Variable node distribution is $\lambda_1 = 0.69$, $\lambda_2 = 0.01$, $\lambda_3 = 0.22$, and $\lambda_4 = 0.08$.
- Threshold is **1.9dB** and capacity is **1.63dB**.

Curve fitting of DGLDPC code C_1 at 1.9dB



Simulation Result of C_1

The LDPC code, which is used to compare with C_1, is chosen from *LDPCopt* with the maximum left degree set to 6, i.e. the same as C_1. The threshold of the chosen LDPC code is 2.0dB.



Construction of DGLDPC code C_2

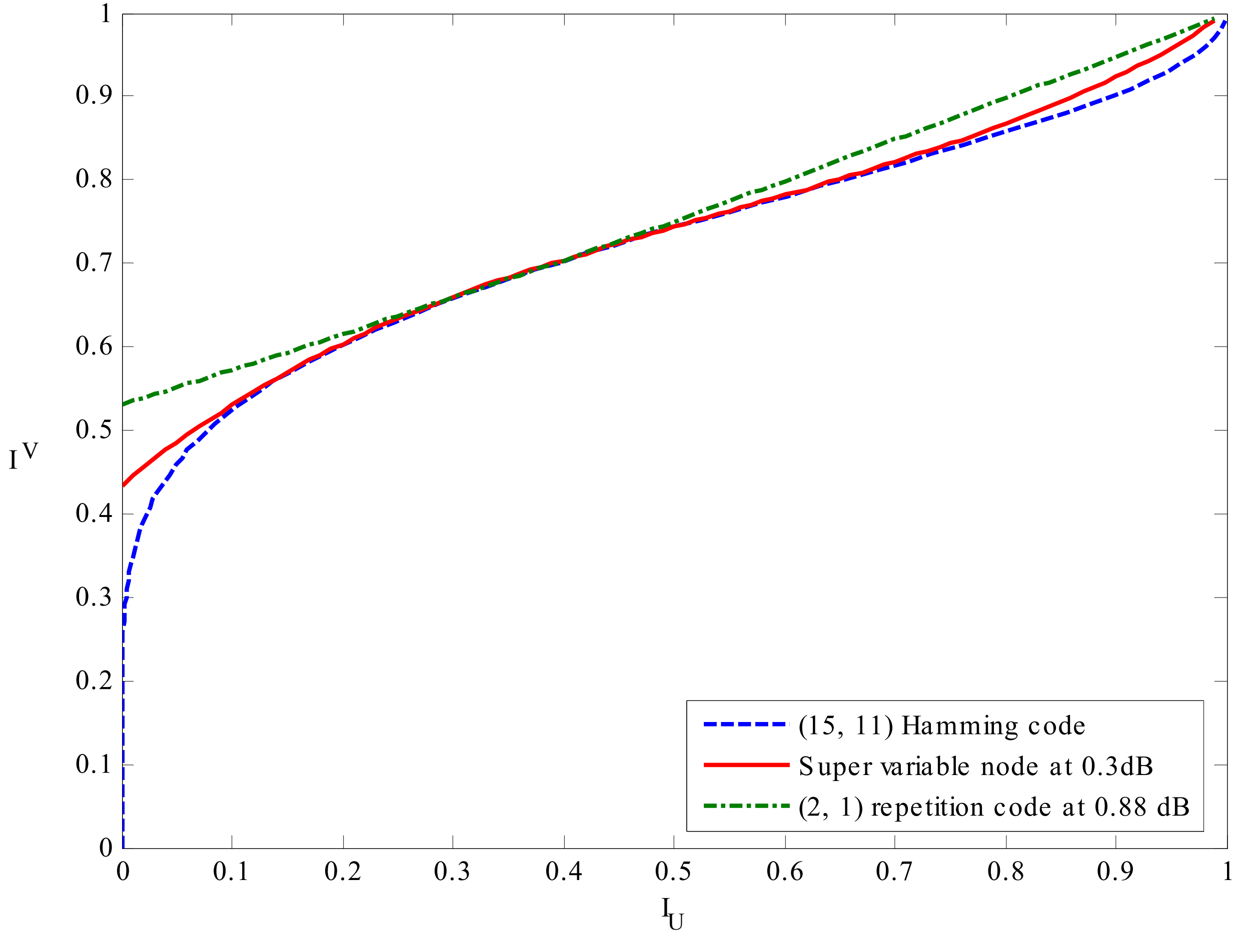
- C_2 is a rate-7/15 length-7650 code.
- Super variable nodes:

  (6,1) repetition code, (6,2) code with generator matrix $\begin{pmatrix} 111100 \\ 001111 \end{pmatrix}$,

  (6,4) code with generator matrix $\begin{pmatrix} 111000 \\ 011100 \\ 001110 \\ 000111 \end{pmatrix}$, (6,5) SPC code.
- Super check node: (15,11) Hamming code
- Variable node distribution is $\lambda_1 = 0.425$, $\lambda_2 = 0.075$, $\lambda_3 = 0.075$, and $\lambda_4 = 0.425$.
- Threshold is **0.3 dB** and capacity is **0.04 dB**.

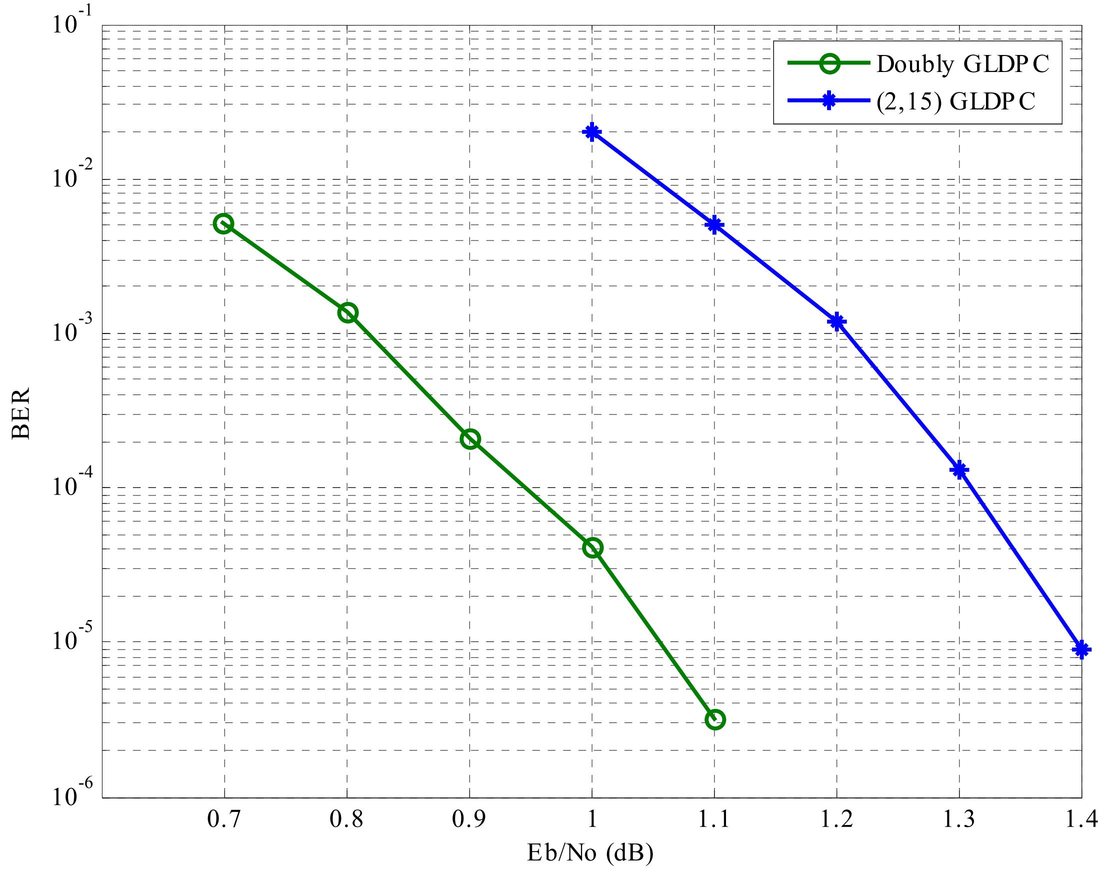
Curve fitting of DGLDPC code C_2

Compare C_2 with a (2,15) GLDPC code



Simulation Result of C_2

The (2, 15) GLDPC code, which is used to compare with C_2 , has the same kind of check node as C_2 , i.e., (15,11) Hamming codes. The simulation result of the (2, 15) GLDPC code is obtained from [Lentmaier *et al*.-CommLett99].



Construction of DGLDPC code C_3

- C_3 is a rate-1/2 length-1536 code.
- Super variable nodes: the (4,1) repetition code and the (4,3) SPC code.
- Super check node: (15,11) Hamming code
- Threshold is **0.77dB** and capacity is **0.18dB**.

Complexity comparison

- The computational complexity per check node of C_3 is

$$O(d_c \times 2^{d_c - k_c}) = O(15 \times 2^{15-11}) = O(15 \times 16)$$

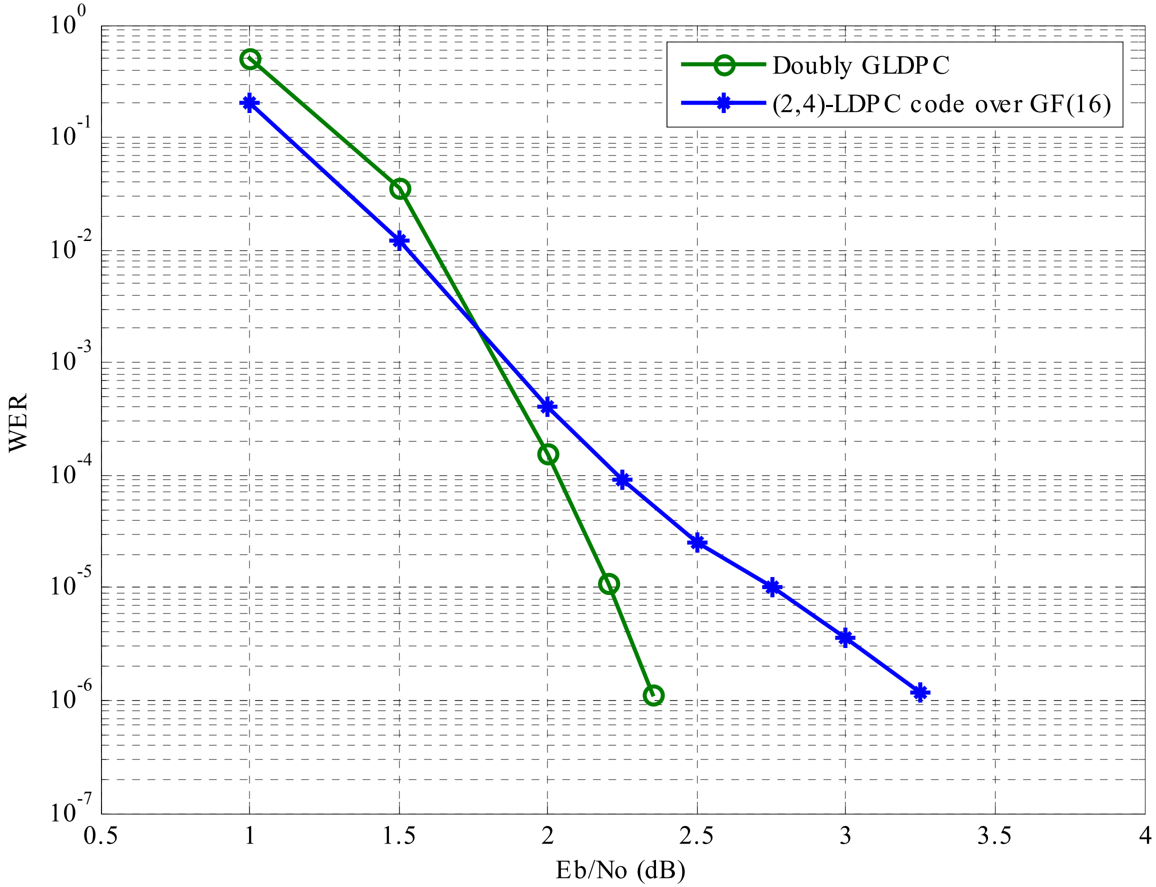
- The computational complexity per check node of the (2,4)-LDPC code over GF(16) is

$$O(d_c \times q \log_2 q) = O(4 \times 16 \log_2 16) = O(16 \times 16)$$

- The variable node processing remains lower.

Simulation Result of C_3

C_3 is compared with an optimized rate-1/2 length-1504 (2,4)-LDPC code over GF(16) [Poulliat *et al.*-ISTCRT06].



Conclusion

- Doubly GLDPC codes can achieve very good threshold. And compared with LDPC and GLDPC codes with the same kind of check nodes, doubly GLDPC codes can improve both error floor region and water fall region.
- Compared with LDPC codes over GF(q) with comparable computational complexity, doubly GLDPC codes can also lead to lower error floor.

Further research

- Very large number of choices (variable node component codes and mappings, check node component codes): good design criteria.
- Evaluate minimum distance of DG-LDPC codes.
- Evaluate stopping distance of DG-LDPC codes.