Shuffled Belief Propagation Decoding

Juntan Zhang and Marc Fossorier
Department of Electrical Engineering
University of Hawaii at Manoa
Honolulu, HI 96816

Outline

- Review of LDPC Codes
- Standard Belief Propagation Algorithm
- Shuffled Belief Propagation Algorithm
- Optimality and Convergence
- Parallel Shuffled Belief Propagation
- A Small Example
- Simulation Results
- Conclusion

Low-Density Parity Check (LDPC) Codes

- First proposed by R. G. Gallager in 1960's, and resurrected recently [Gallager-IRE62, MacKay-IT99] .
- Can achieve near Shannon limit performance with belief propagation (BP) or sum-product algorithm [Richardson-Urbanke-IT01] .
- Advantages over turbo codes:

  better distance properties;

  parallel decoding structure for high speed decoders.
- Disadvantages:

  encoding complexity is high;

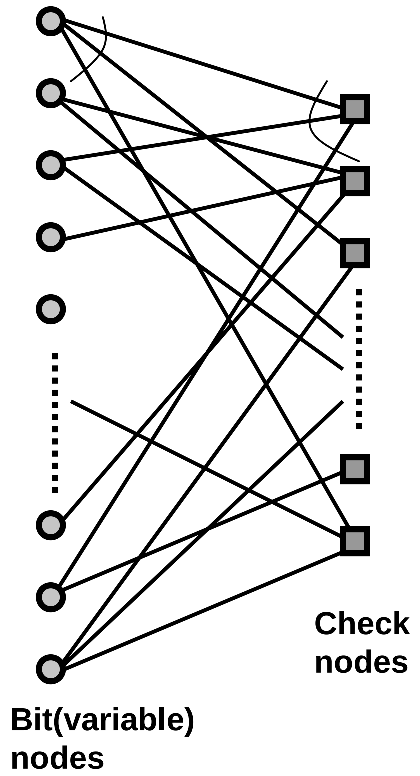
  decoder complexity is high for full parallel implementation.

Representations of LDPC Codes

parity check matrix

$$\mathbf{H}=\underbrace{\begin{bmatrix} 1 & 0 & 1 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \ddots & & & & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & & \ddots & & & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & & & \ddots & & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & & & \ddots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \cdots & \cdots & \cdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 & 1 \end{bmatrix}}_{N} \Bigg\} M$$

bipartite graph



Regular and Irregular LDPC Codes

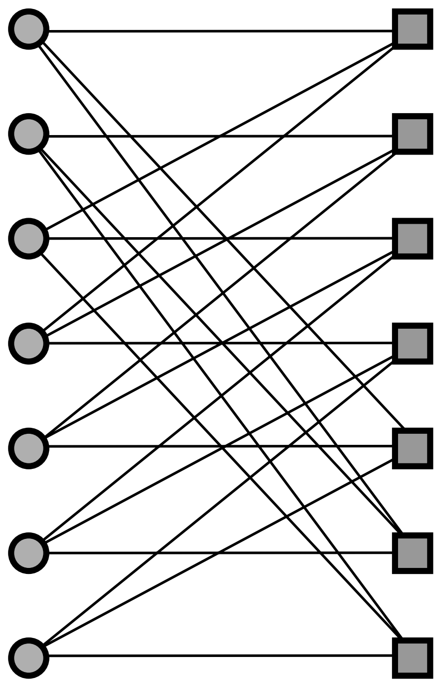
- An LDPC code is *regular* if H has constant row weight and column weight, or equivalently, the check nodes have constant degree d_c and variable nodes have constant degree d_v;
- An LDPC code is *irregular* if row and column weights are not constants.
- *Irregular* LDPC codes are defined by degree distributions
- Long *irregular* LDPC codes have better performance than *regular* LDPC codes, and can beat turbo codes [Richardson-Urbanke-IT01] .

Geometric LDPC Codes

- Originally studied for majority logic decoding decades ago, and constructed based on finite geometries (Euclidean and projective geometries) [Weldon-Bell66, Rudolph-IT67];
- BP algorithm can be applied to the decoding of this family of codes [Lucas-Fossorier-Kou-Lin-COM00, Kou-Lin-Fossorier-IT01];
- Encoding can be easily implemented with shift registers since they are cyclic codes;
- They have very good minimum distance properties;
- Decoding complexity is high.

An example: (7, 3) DSC code:

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



- **Parity matrix is Squared !**
- **Not full rank.**

- **Equal number of bit nodes and check nodes.**
- **Node degrees are larger.**

Some one-step majority decodable codes:

PG-LDPC codes (DSC)

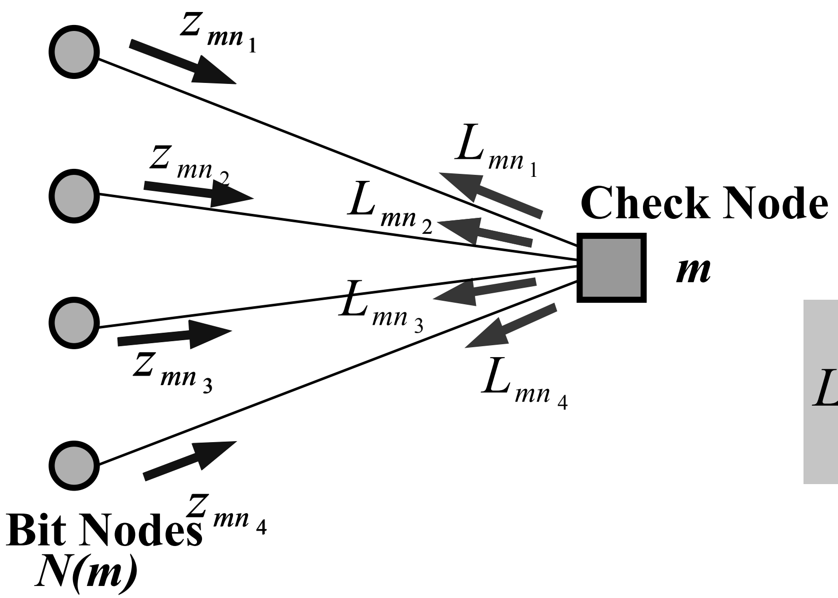
(N, K)	**rate**	d_{min}
(7, 3)	0.429	4
(21, 11)	0.524	6
(73, 45)	0.616	10
(273, 191)	0.700	18
(1057, 813)	0.769	34
(4161, 3431)	0.825	66

EG-LDPC codes

(N, K)	**rate**	d_{min}
(15, 7)	0.467	5
(63, 37)	0.587	9
(255, 175)	0.686	17
(1023, 781)	0.763	33
(4095, 3367)	0.822	65

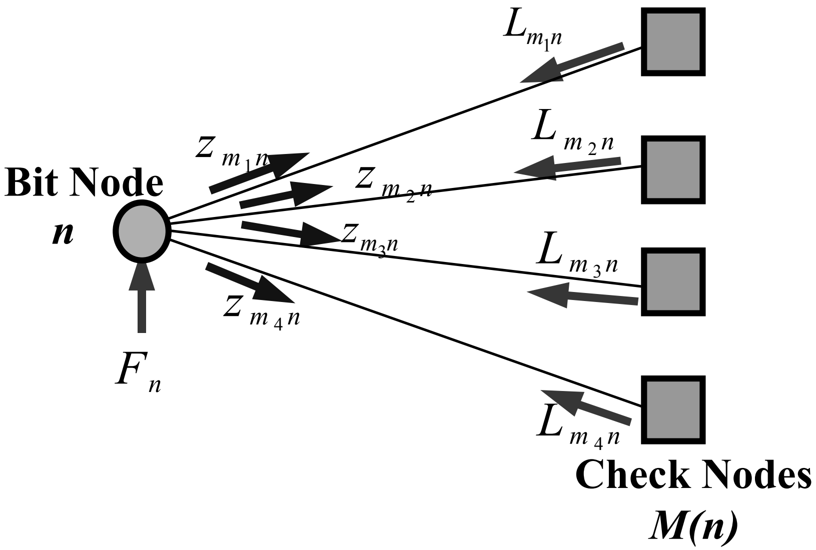
Processing in check nodes: Principles:

incoming messages + constraints ⇒ outgoing messages



$$L_{mn} = 2\tanh^{-1}\left(\prod_{n' \in N(m)\backslash n} \tanh\left(z_{mn'}/2\right)\right)$$

Processing in bit nodes:



$$z_{mn} = F_n + \sum_{m' \in M(n) \backslash m} L_{m'n}$$

$$z_n = F_n + \sum_{m \in M(n)} L_{mn} \text{, for hard decision}$$

Standard IDBP

- Initialization
- Step1: Update the Belief Matrix
  - Horizontal Step: Update the whole Check-to-Bit Matrix
  - Vertical Step: Update the whole Bit-to-Check Matrix
- Step2: Hard decision and stopping test
- Step3: Output the decoded codeword

Standard BP Algorithm; Step 1:

(i) Horizontal Step:

$$\varepsilon_{mn}^{(i)} = \log \frac{1 + \prod_{n' \in N(m) \backslash n} \tanh(z_{mn'}^{(i-1)} / 2)}{1 - \prod_{n' \in N(m) \backslash n} \tanh(z_{mn'}^{(i-1)} / 2)}$$

(ii)Vertical Step:

$$z_{mn}^{(i)} = F_n + \sum_{m' \in M(n) \backslash m} \varepsilon_{m'n}^{(i)}$$

$$z_n^{(i)} = F_n + \sum_{m \in M(n)} \varepsilon_{mn}^{(i)}$$

Update the belief matrix in i-th iteration with standard belief propagation decoding

$$
\begin{bmatrix}
\varepsilon_{00}^{(i)} & \varepsilon_{01}^{(i)} & & \varepsilon_{03}^{(i)} & & & \\
& \varepsilon_{11}^{(i)} & \varepsilon_{12}^{(i)} & & \varepsilon_{14}^{(i)} & & \\
& & \varepsilon_{22}^{(i)} & \varepsilon_{23}^{(i)} & & \varepsilon_{25}^{(i)} & \\
& & & \varepsilon_{33}^{(i)} & \varepsilon_{34}^{(i)} & & \varepsilon_{36}^{(i)} \\
\varepsilon_{40}^{(i)} & & & & \varepsilon_{44}^{(i)} & \varepsilon_{45}^{(i)} & \\
& \varepsilon_{51}^{(i)} & & & & \varepsilon_{55}^{(i)} & \varepsilon_{56}^{(i)} \\
\varepsilon_{60}^{(i)} & & \varepsilon_{62}^{(i)} & & & & \varepsilon_{66}^{(i)}
\end{bmatrix}
\begin{bmatrix}
z_{00}^{(i)} & z_{01}^{(i)} & & z_{03}^{(i)} & & & \\
& z_{11}^{(i)} & z_{12}^{(i)} & & z_{14}^{(i)} & & \\
& & z_{22}^{(i)} & z_{23}^{(i)} & & z_{25}^{(i)} & \\
& & & z_{33}^{(i)} & z_{34}^{(i)} & & z_{36}^{(i)} \\
z_{40}^{(i)} & & & & z_{44}^{(i)} & z_{45}^{(i)} & \\
& z_{51}^{(i)} & & & & z_{55}^{(i)} & z_{56}^{(i)} \\
z_{60}^{(i)} & & z_{62}^{(i)} & & & & z_{66}^{(i)}
\end{bmatrix}
$$

Shuffled Belief Propagation

- Initialization
- Step1: Update the Belief Matrix, for n=0,..N-1
  - Horizontal Step: Update the n-th column of Check-to-Bit Matrix
  - Vertical Step: Update the n-th column of Bit-to-Check Matrix
- Step2: Hard decision and stopping test
- Step3: Output the decoded codeword

Shuffled Belief Propagation; Step 1:

(i) Horizontal Step:

$$\varepsilon_{mn}^{(i)} = \log \frac{1 + \prod_{\substack{n' \in N(m) \backslash n \\ n' < n}} \tanh(z_{mn'}^{(i)} / 2) \prod_{\substack{n' \in N(m) \backslash n \\ n' > n}} \tanh(z_{mn'}^{(i-1)} / 2)}{1 - \prod_{\substack{n' \in N(m) \backslash n \\ n' < n}} \tanh(z_{mn'}^{(i)} / 2) \prod_{\substack{n' \in N(m) \backslash n \\ n' > n}} \tanh(z_{mn'}^{(i-1)} / 2)}$$

(ii)Vertical Step:

$$z_{mn}^{(i)} = F_n + \sum_{m' \in M(n) \backslash m} \varepsilon_{m'n}^{(i)}$$

$$z_n^{(i)} = F_n + \sum_{m \in M(n)} \varepsilon_{mn}^{(i)}$$

Update the belief matrix in i-th iteration with shuffled belief propagation decoding

$$
\begin{bmatrix}
\varepsilon_{00}^{(i)} & \varepsilon_{01}^{(i)} & & \varepsilon_{03}^{(i)} & & & \\
 & \varepsilon_{11}^{(i)} & \varepsilon_{12}^{(i)} & & \varepsilon_{14}^{(i)} & & \\
 & & \varepsilon_{22}^{(i)} & \varepsilon_{23}^{(i)} & & \varepsilon_{25}^{(i)} & \\
 & & & \varepsilon_{33}^{(i)} & \varepsilon_{34}^{(i)} & & \varepsilon_{36}^{(i)} \\
\varepsilon_{40}^{(i)} & & & & \varepsilon_{44}^{(i)} & \varepsilon_{45}^{(i)} & \\
 & \varepsilon_{51}^{(i)} & & & & \varepsilon_{55}^{(i)} & \varepsilon_{56}^{(i)} \\
\varepsilon_{60}^{(i)} & & \varepsilon_{62}^{(i)} & & & & \varepsilon_{66}^{(i)}
\end{bmatrix}
\begin{bmatrix}
z_{00}^{(i)} & z_{01}^{(i)} & & z_{03}^{(i)} & & & \\
 & z_{11}^{(i)} & z_{12}^{(i)} & & z_{14}^{(i)} & & \\
 & & z_{22}^{(i)} & z_{23}^{(i)} & & z_{25}^{(i)} & \\
 & & & z_{33}^{(i)} & z_{34}^{(i)} & & z_{36}^{(i)} \\
z_{40}^{(i)} & & & & z_{44}^{(i)} & z_{45}^{(i)} & \\
 & z_{51}^{(i)} & & & & z_{55}^{(i)} & z_{56}^{(i)} \\
z_{60}^{(i)} & & z_{62}^{(i)} & & & & z_{66}^{(i)}
\end{bmatrix}
$$

Implementation of shuffled BP

- Backward-forward implementation
- Computation Complexity

Optimality and Convergence Property of Shuffled BP

Given the Tanner Graph of the code is connected and acyclic.

- Shuffled BP is optimal in the sense of MAP
- Shuffled BP converges faster (or at least no slower) than Standard BP

Parallel Shuffled BP

- Divide the N bits into G groups, each group contains N_G bits. (regular partition).

- In each group, the updatings are processed in parallel. The processings of groups are in sequential.

Parallel Shuffled Belief Propagation

(i) Horizontal Step:

$$\varepsilon_{mn}^{(i)} = \log \frac{1 + \prod_{\substack{n' \in N(m)\backslash n \\ n' \le g \cdot N_G - 1}} \tanh(z_{mn'}^{(i)} / 2) \prod_{\substack{n' \in N(m)\backslash n \\ n' \ge g \cdot N_G}} \tanh(z_{mn'}^{(i-1)} / 2)}{1 - \prod_{\substack{n' \in N(m)\backslash n \\ n' \le g \cdot N_G - 1}} \tanh(z_{mn'}^{(i)} / 2) \prod_{\substack{n' \in N(m)\backslash n \\ n' \ge g \cdot N_G}} \tanh(z_{mn'}^{(i-1)} / 2)}$$

(ii)Vertical Step:

$$z_{mn}^{(i)} = F_n + \sum_{m' \in M(n)\backslash m} \varepsilon_{m'n}^{(i)}$$

$$z_n^{(i)} = F_n + \sum_{m \in M(n)} \varepsilon_{mn}^{(i)}$$

Update the belief matrix in i-th iteration with Group shuffled BP decoding

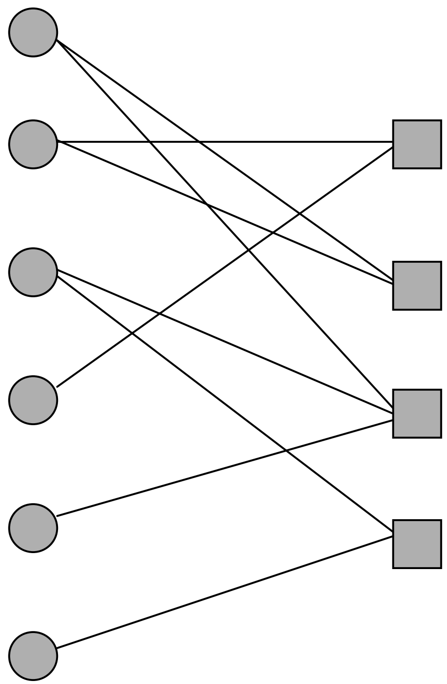
$$
\begin{bmatrix}
\varepsilon_{00}^{(i)} & \varepsilon_{01}^{(i)} & & \varepsilon_{03}^{(i)} & & & \\
& \varepsilon_{11}^{(i)} & \varepsilon_{12}^{(i)} & & \varepsilon_{14}^{(i)} & & \\
& & \varepsilon_{22}^{(i)} & \varepsilon_{23}^{(i)} & & \varepsilon_{25}^{(i)} & \\
& & & \varepsilon_{33}^{(i)} & \varepsilon_{34}^{(i)} & & \varepsilon_{36}^{(i)} \\
\varepsilon_{40}^{(i)} & & & & \varepsilon_{44}^{(i)} & \varepsilon_{45}^{(i)} & \\
& \varepsilon_{51}^{(i)} & & & & \varepsilon_{55}^{(i)} & \varepsilon_{56}^{(i)} \\
\varepsilon_{60}^{(i)} & & \varepsilon_{62}^{(i)} & & & & \varepsilon_{66}^{(i)}
\end{bmatrix}
\begin{bmatrix}
z_{00}^{(i)} & z_{01}^{(i)} & & z_{03}^{(i)} & & & \\
& z_{11}^{(i)} & z_{12}^{(i)} & & z_{14}^{(i)} & & \\
& & z_{22}^{(i)} & z_{23}^{(i)} & & z_{25}^{(i)} & \\
& & & z_{33}^{(i)} & z_{34}^{(i)} & & z_{36}^{(i)} \\
z_{40}^{(i)} & & & & z_{44}^{(i)} & z_{45}^{(i)} & \\
& z_{51}^{(i)} & & & & z_{55}^{(i)} & z_{56}^{(i)} \\
z_{60}^{(i)} & & z_{62}^{(i)} & & & & z_{66}^{(i)}
\end{bmatrix}
$$

A Small Example: LDPC (6,2) code

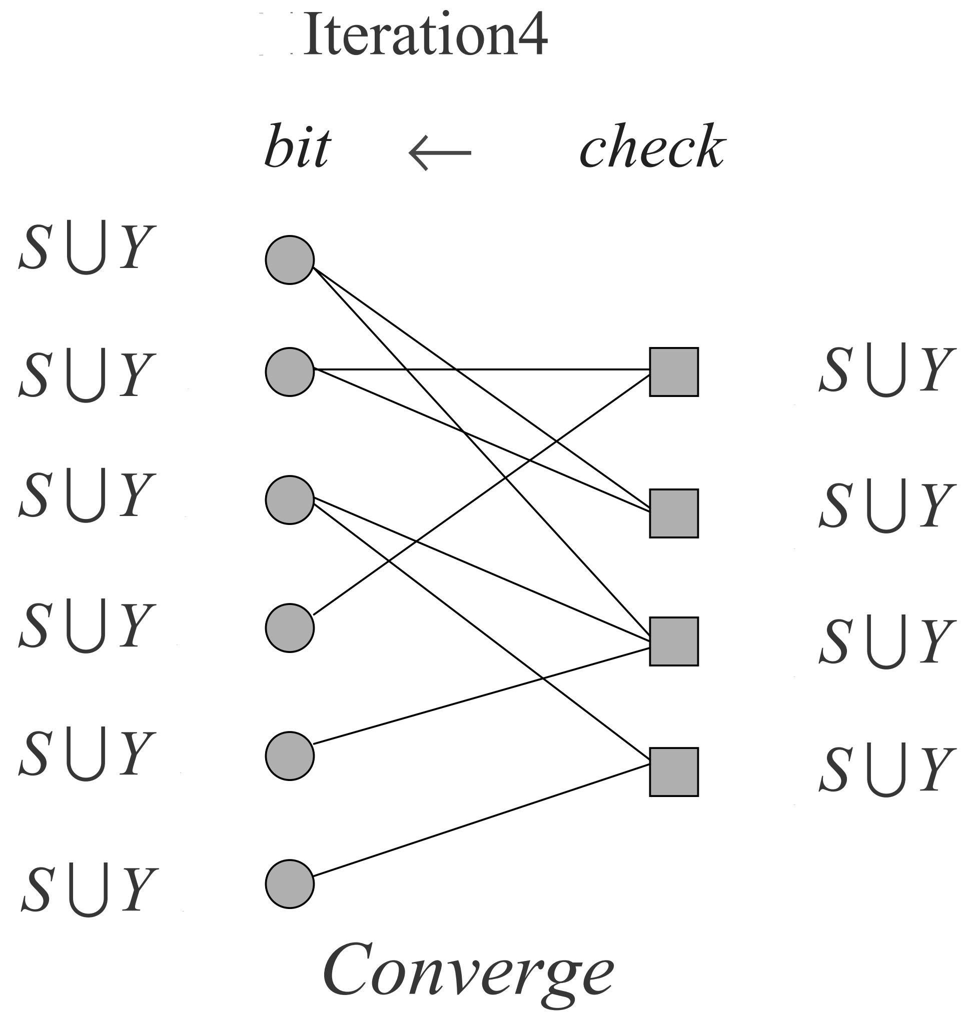
Parity Check Matrix

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Tanner Graph



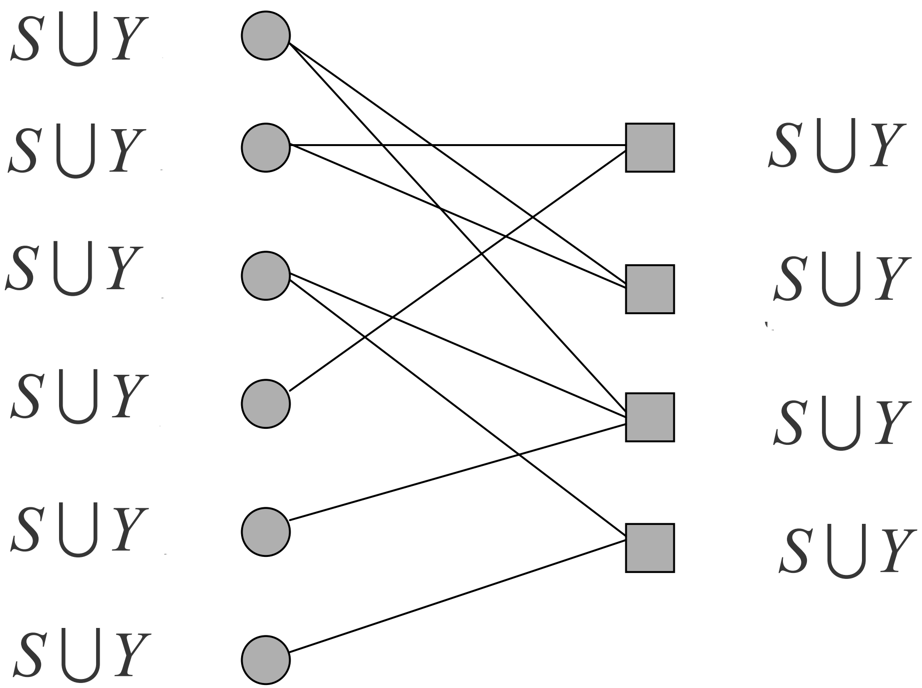
Decoding with Standard BP



Decoding with Shuffled BP

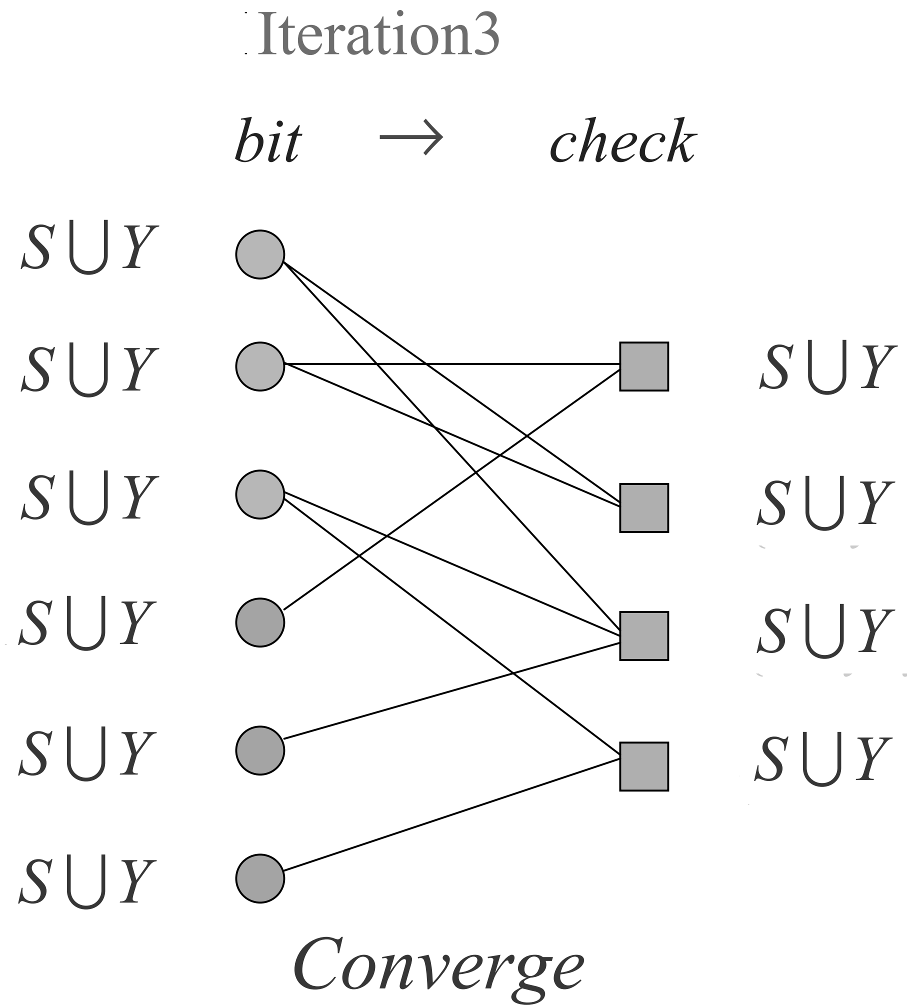
Iteration2

bit $\rightarrow$ *check*



Converge

Decoding with Group Shuffled BP $(G = 2)$



Comparison of Speed of Convergence

- Standard BP

$$I^{G=1} = \begin{bmatrix} 2 & 3 & 3 & 4 & 3 & 4 \end{bmatrix}$$

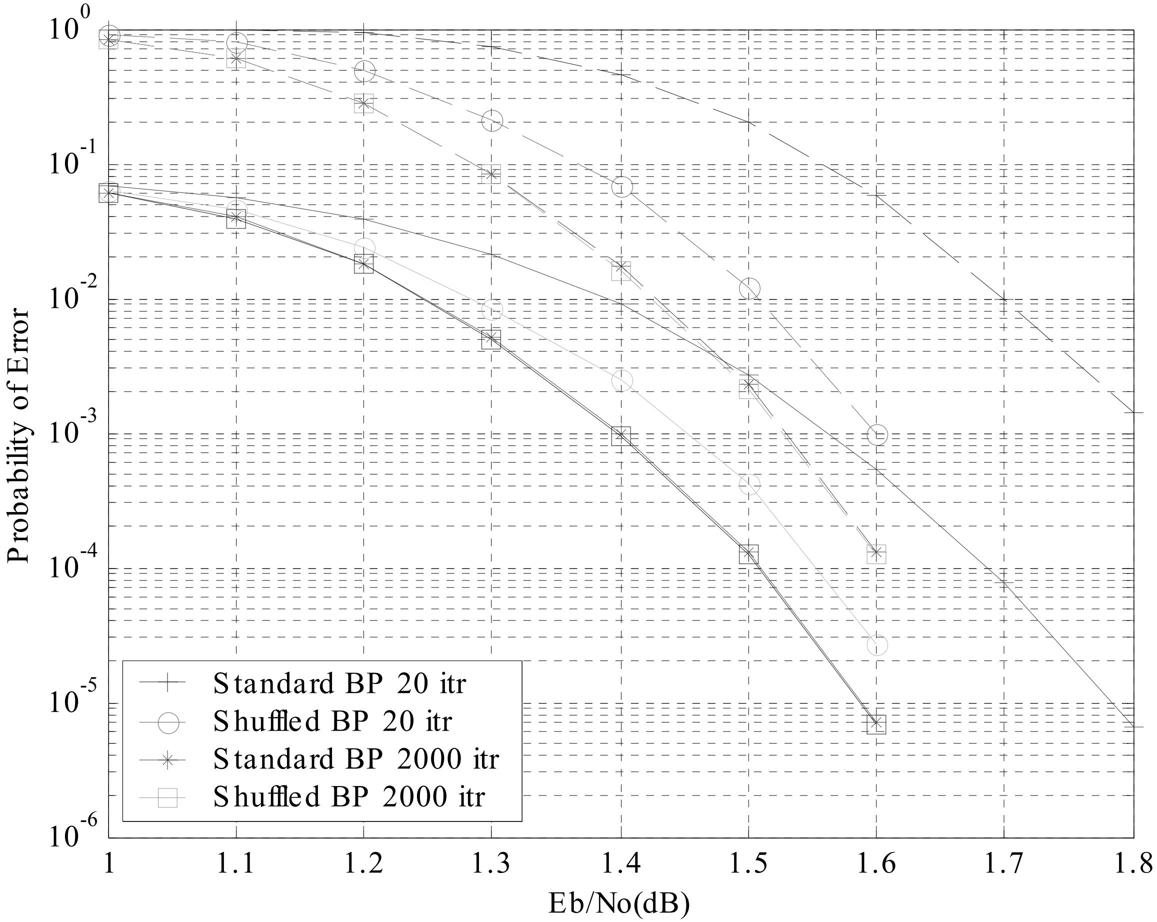
- Shuffled BP

$$I^{G=6} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

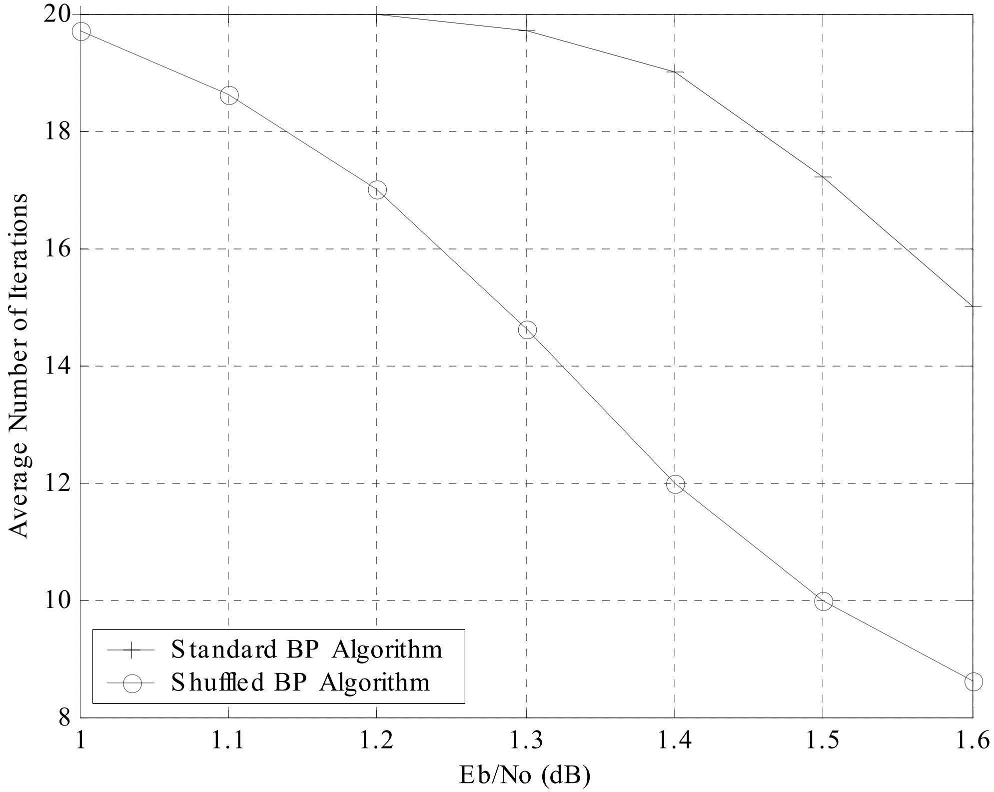
- Group Shuffled BP

$$I^{G=2} = \begin{bmatrix} 2 & 3 & 3 & 3 & 2 & 3 \end{bmatrix}$$

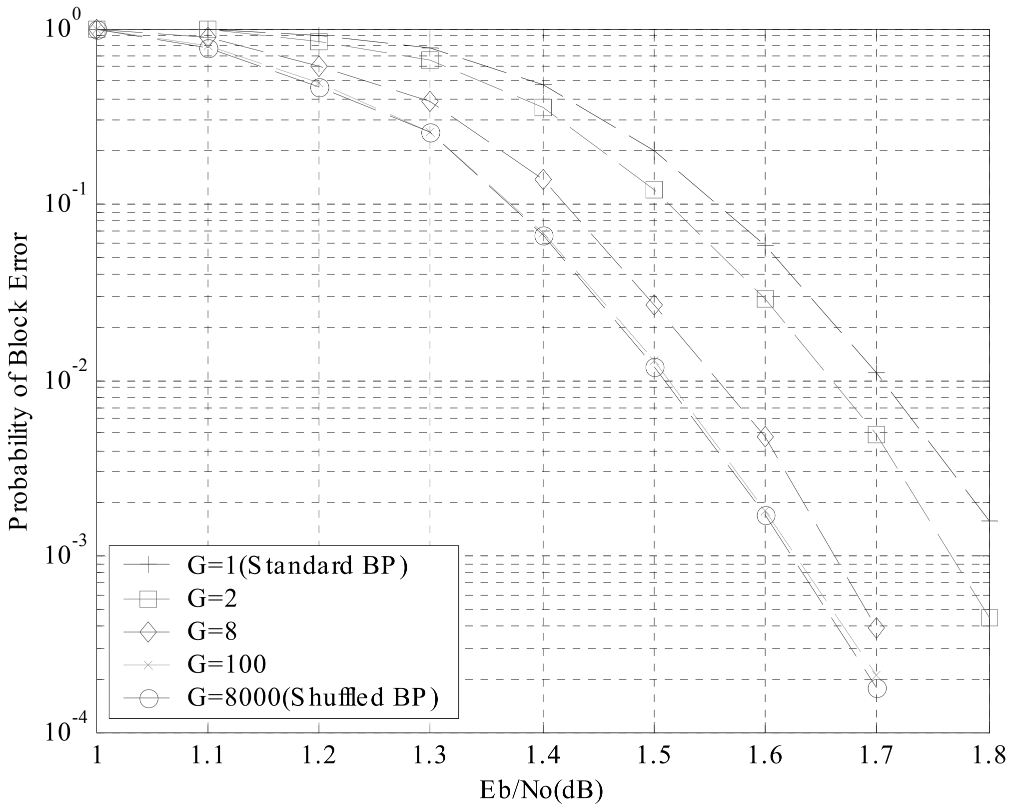
Pe of LDPC (8000,4000)(3,6) code with shuffled and standard BP



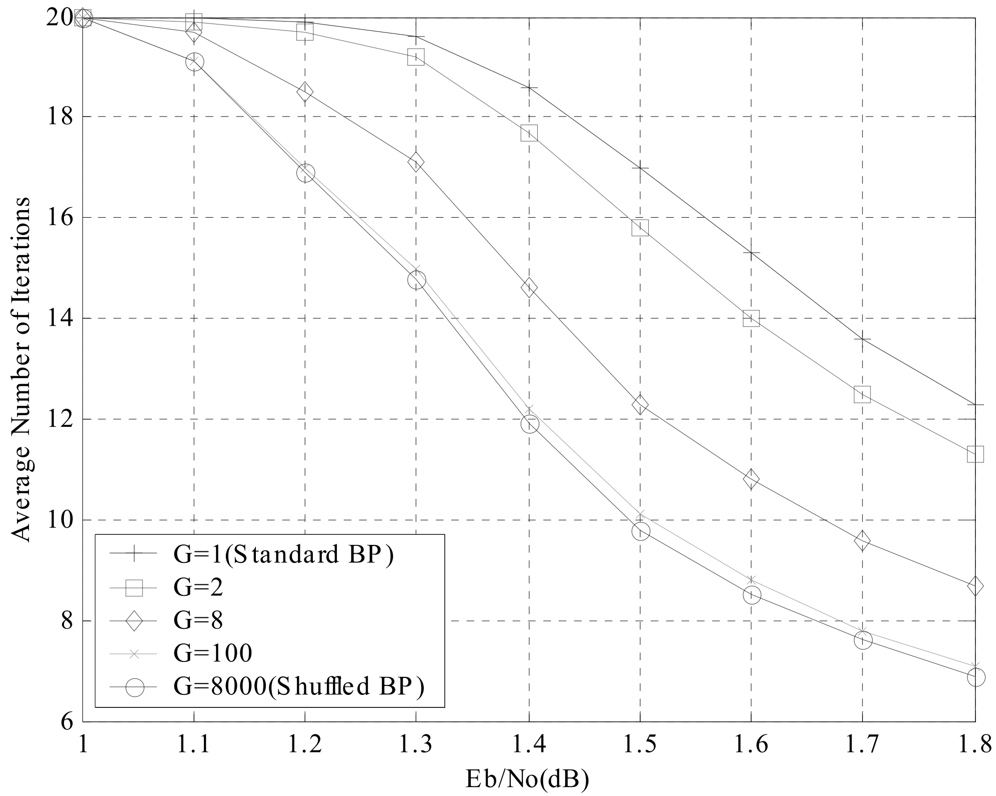
Average Number of Iterations



Pe of LDPC(8000,4000)(3,6) with Group Shuffled BP decoding



Average Number of Iterations



Conclusion

- Shuffled BP achieves a good trade-off between performance and complexity
- Group shuffled BP can decrease decoding delay and is suitable for hardware implementation.