Performance Analysis of Iterative Decoding Algorithms with Memory over Memoryless Channels

Authors: Emil Janulewics, Amir H. Banihashemi

Publication: IEEE T. Comm, Dec 2012
Speaker: Jeong-Min Ryu

Short summary:

In this work, they propose a model for iterative decoding algorithms with **memory** which covers successive relaxation (SR) version of belief propagation and differential decoding with binary message passing (DD-BMP) algorithms as special cases. Based on this model, they derive a Bayesian network for iterative algorithms with memory over memoryless channels and use this representation to analyze the performance of the algorithms using density evolution.

I. INTRODUCTION

Iterative decoding algorithm → Decoding algorithm of LDPC codes

Low-density parity-check (LDPC) codes are known to have good performance when decoded with iterative decoding algorithms, also known as message-passing algorithms.

Density Evolution → An analytical tool of LDPC codes

An analytical tool called *density evolution* can be used to find the threshold of a particular code ensemble under a given iterative decoding algorithm. The threshold is an asymptotic measure of performance and is defined as the worst channel parameter (e.g., largest noise variance) for which the probability of error still converges to zero as the number of iterations tends to infinity

Density Evolution → A technique for constructing irregular LDPC codes

Density evolution is also a powerful technique for constructing irregular LDPC codes through the optimization of the degree distributions.

All message-passing algorithms analyzed by density Evolution → Memoryless

To the best of our knowledge however, all the message-passing algorithms analyzed by density evolution in the literature are memoryless, i.e., the output message of a variable node (check node) at iteration l is only a function of the input messages to that node at iteration l (l -1) and also of the initial message of the channel in the case of variable nodes.

Iterative decoding algorithms with memory → Exist

There exist however a number of iterative decoding algorithms, such as successive relaxation (SR) variants of BP and MS and DD-BMP (differential decoding with binary message-passing), that have memory.

The presence of memory in algorithms → Improves the performance but makes the density evolution analysis much more complex.

In this paper, they develop the framework for the density evolution analysis of iterative extrinsic message-passing algorithms with memory which includes DD-BMP and SR algorithms.

They employ the *Bayesian network representation* via a directed acyclic graph (DAG), to capture the dependences among different messages and memory contents in a space with two dimensions: iteration l and the depth of the decoding tree d.

Independent

Incoming messages to a node along different edges

Dependencies

A message passed along a given edge at iteration l

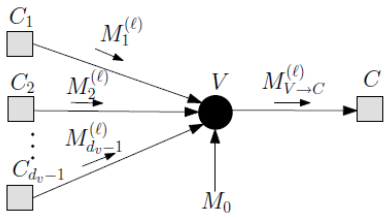
All the messages passed along that edge at previous iteration $l'<l$.

Such dependencies cause the complexity of density evolution to grow at least exponentially with l. They derive the density evolution equations and use techniques to make them tractable.

II. ITERATIVE DECODING ALGORITHMS WITH MEMORY

A. General Model

1) Memoryless decoding algorithm



(a) Memoryless

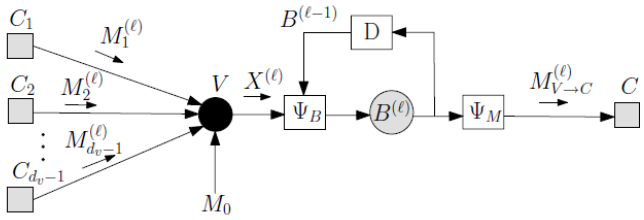
The figure (a) shows a snapshot of the Tanner graph of an LDPC code at iteration l for a memoryless decoding algorithm, where variable and check nodes are represented by circles and squares, respectively.

Under cycle-free assumption and based on the principle of extrinsic message passing,

- Incoming messages $M_1^{(l)},\ldots,M_{d_v-1}^{(l)}$ to node V are independent of each other and of the channel message M_0.
- The outgoing message $M_{V\to C}^{(l)}$ of node V to node C at iteration l is a function of d_v-1 i.i.d. random variables and the channel message.
- The outgoing message of a check node is a function of d_c-1 i.i.d. random variables corresponding to the extrinsic incoming message.

→ The distribution of a function of independent random variables is relatively easy to find since the joint distribution of these variables is the product of their marginal distribution. In this case, one can recursively derive the distribution of messages at iteration l as a deterministic function of the distribution at iteration l-1 with a complexity that is independent of l.

2) Memory decoding algorithm



(b) With Memory

Similar to figure (a), we have a set of i.i.d. extrinsic incoming messages to a variable node *V*.

The outgoing message from V: $X^{(l)}$

Memory units: Ψ_B, Ψ_M, $B^{(l)}$, D.

$B^{(l)}$ is updated by $\Psi_B\left(X^{(l)}, B^{(l-1)}\right)$, where Ψ_B is a deterministic function of $X^{(l)}$ and the content of the memory at iteration *l*-1.

The incoming message $M_{V\to C}^{(l)}$ to node *C* from *V* is obtained by $\Psi_M\left(B^{(l)}\right)$

Note that while the message $X^{(l)}$ is a function of independent random variables, the outgoing messages, $M_{V\to C}^{(l)}$, is a function of dependent random variables $X^{(l)}$ and $B^{(l-1)}$.

Our focus will be on the link from variable nodes to check nodes and on finding the distribution of $B^{(l)}$ and $M_{V\to C}^{(l)}$.

B. SR and DD-BMP Algorithms

1) SR Algorithms: Any standard memoryless iterative algorithm, such as BP or MS, can be turned into an SR algorithm by proper introduction of memory. SR algorithms can be performed in different message domains. In this work, we assume log-likelihood ratio (LLR) domain for messages (SRLLR). Based on the model of Fig. 1(b), the SR version is defined by the following variable node map

$$\begin{aligned} B^{(l)} &= \Psi_B\left(B^{(l-1)}, X^{(l)}\right) = (1-\beta)B^{(l-1)} + \beta X^{(l)}, \\ M_{V\to C}^{(l)} &= \Psi_M\left(B^{(l)}\right) = B^{(l)}, \end{aligned} \tag{1}$$

where $X^{(l)} = \Psi_V\left(M_0, M_1^{(l)}, ..., M_{d_v-1}^{(l)}\right)$. In (1), β is called the *relaxation factor*, and can be optimized for the best performance. The optimal value of β is usually in the interval (0, 1).

2) DD-BMP: Differential decoding with binary message passing (DD-BMP) was introduced as an attractive alternative to purely hard-decision algorithms. This algorithm combines the simplicity of binary message-passing with the good performance of soft-decision algorithms, where the soft information is stored in edge- or node-based memories. In the former case, studied in this paper, the variable node map, following the model of Fig. 1(b), is defined by

$$\begin{aligned} B^{(l)} &= \Psi_B\left(B^{(l-1)}, X^{(l)}\right) = B^{(l-1)} + X^{(l)}, \\ M_{V\to C}^{(l)} &= \Psi_M\left(B^{(l)}\right) = \mathrm{sgn}_r\left(B^{(l)}\right), \end{aligned} \tag{2}$$

where $\mathrm{sgn}_r(x) = 1$ for $x > 0$, and $= -1$ for $x < 0$. For $x = 0$, $\mathrm{sgn}_r(x)$ takes +1 or -1 randomly with equal probability. In (2), $X^{(l)} = \Psi_V\left(M_0, M_1^{(l)}, ..., M_{d_v-1}^{(l)}\right)$, which for the BIAWGN channel reduces to $X^{(l)} = \sum_{i=1}^{d_v-1} M_i^{(l)}$.

Both the variable and the check node operations (particularly the latter) are simpler for DD-BMP compared to BP and MS algorithms.

C. Symmetry of the Decoder and Error Probability

The analysis of iterative decoders is greatly simplified assuming that both the channel and the decoder are symmetric.

In particular, the variable node symmetry condition has some implications on the choices of the mappings Ψ_B and Ψ_M: Ψ_B should be sign inversion invariant, and $\Psi_M(-x) = -\Psi_M(x)$. As it can be seen in (1) and (2), both conditions are satisfied for SRLLR and DD-BMP algorithms.

With both the channel and the decoder being symmetric, we can assume, without loss of generality, that the all-zero codeword is transmitted. In this case, the average fraction of incorrect messages passed at iteration *l* from variable nodes to check nodes is calculated by

$$P_e^{(l)} = P\left(B^{(l)} < 0\right) + \frac{1}{2} P\left(B^{(l)} = 0\right). \quad (3)$$

We refer to $P_e^{(l)}$ in (3) as the probability of bit error at iteration *l*.

III. Bayesian Network Representation of Iterative Decoding Algorithms with Memory

A. Bayesian Networks and Conditional Independence

In this work, they use a Bayesian network to represent the dependencies among different messages and memory contents of an iterative algorithm with memory.

The *conditional independence* between two sets of random variables $\mathcal{X}$ and $\mathcal{Y}$ given a third set $\mathcal{Z}$ is defined by

$$\mathcal{X} \perp \mathcal{Y} \mid \mathcal{Z} \Leftrightarrow \mathbb{P}\left(\mathcal{X}, \mathcal{Y} \mid \mathcal{Z}\right) = \mathbb{P}\left(\mathcal{X} \mid \mathcal{Z}\right) \mathbb{P}\left(\mathcal{Y} \mid \mathcal{Z}\right)$$

where $\mathbb{P}\left(\mathcal{X}\right)$ and $\mathbb{P}\left(\mathcal{X} \mid \mathcal{Z}\right)$ are the marginal distribution of $\mathcal{X}$ and the conditional distribution of $\mathcal{X}$ given $\mathcal{Z}$, respectively.

B. Bayesian Networks of Iterative Decoders with Memory

Based on the principle of extrinsic message-passing, one can see that $X^{(l)}$ is a deterministic function of $B_1^{(l-1)}$ and M_0. Moreover, as it can be seen in Fig. 1(b), $B^{(l)}$ is a function of $B^{(l-1)}$ and $X^{(l)}$. In addition, $B_i^{(l)}$ depends on $B_i^{(l-1)}$ and $B_{i+1}^{(l-1)}$.

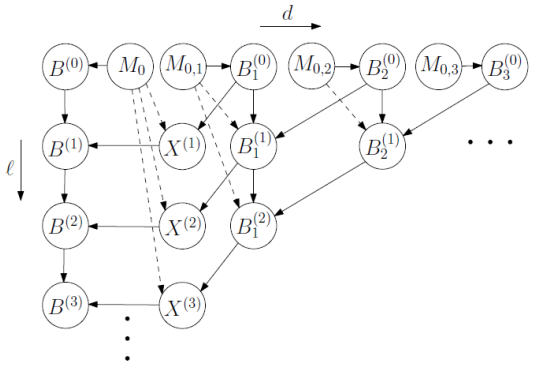


Fig. 3. Bayesian network based on the model of Fig. 1(b).

IV. DENSITY EVOLUTION

Based on (3), to obtain the error probability $P_e^{(l)}$ at each iteration l, we need to compute $\mathbb{P}\left(B^{(k)}\right)$.

A efficient approach is to compute $\mathbb{P}\left(B^{(k)}\right)$ is:

$$P\left(B^{(l)}=b\right)=\sum_{(b',x)\in S_B\times S_X : b=\Psi_B(b',x)} P\left(B^{(l-1)}=b', X^{(l)}=x\right),\ \forall b\in S_B\ .$$

where S_B is sample space of $B^{(l)}$ and $\mathbb{P}\left(B^{(l-1)}, X^{(l)}\right)$ is

$$\begin{aligned}\mathbb{P}\left(B^{(l-1)}, X^{(l)}\right)&=\sum_{x^{(1\to l-1)}}\mathbb{P}\left(B^{(l-1)}\middle| X^{(1\to l-1)}\right)\mathbb{P}\left(X^{(l)}\middle| X^{(1\to l-1)}\right)\mathbb{P}\left(X^{(1\to l-1)}\right)\\&=\sum_{x^{(1\to l-1)}}\mathbb{P}\left(B^{(l-1)}\middle| X^{(1\to l-1)}\right)\mathbb{P}\left(X^{(1\to l)}\right)\end{aligned}$$

where $X^{(1\to k)}=\left\{X^{(1)}, X^{(2)},\cdots, X^{(k)}\right\}$ for $k\geq 1$.

- Calculation of $\mathbb{P}\left(B^{(l)}\middle| X^{(1\to l)}\right)$

$$\begin{aligned}\mathbb{P}\left(B^{(l)}\middle| X^{(1\to l)}\right)&=\sum_{b^{(l-1)}}\mathbb{P}\left(B^{(l)}\middle| X^{(1\to l)}, B^{(l-1)}\right)\mathbb{P}\left(B^{(l-1)}\middle| X^{(1\to l)}\right)\\&=\sum_{b^{(l-1)}}\mathbb{P}\left(B^{(l)}\middle| X^{(l)}, B^{(l-1)}\right)\mathbb{P}\left(B^{(l-1)}\middle| X^{(1\to l-1)}\right),\ \ l\geq 2\end{aligned}$$

- Calculation of $\mathbb{P}\left(X^{(1\to l)}\right)$

The variables $M_i^{(l)}$ are i.i.d.. Since $X^{(l)}=\Psi_v\left(M_1^{(l)}, M_2^{(l)},\ldots, M_{d_v-1}^{(l)}\right)$, $X^{(l)}$ is conditionally independent of all other random variable given $M_{1\to d_v-1}^{(l)}$. We thus have

$$\begin{aligned}\mathbb{P}\left(X^{(1\to l)}\right)&=\sum_{m_{1\to d_v-1}^{(1\to l)}}\mathbb{P}\left(X^{(1\to l)}\middle| M_{1\to d_v-1}^{(1\to l)}\right)\mathbb{P}\left(M_{1\to d_v-1}^{(1\to l)}\right)\\&=\sum_{m_{1\to d_v-1}^{(1\to l)}}\prod_{i=1}^{l}\mathbb{P}\left(X^{(i)}\middle| M_{1\to d_v-1}^{(i)}\right)\prod_{j=1}^{d_v-1}\mathbb{P}\left(M_j^{(1\to l)}\right)\end{aligned}\qquad (4)$$

V. Memory Truncation

A. Main Idea

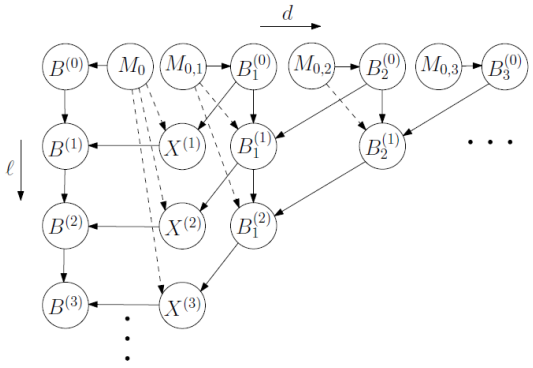


Fig. 3. Bayesian network based on the model of Fig. 1(b).

To explain the approximation, we consider the calculation of the joint distribution $\mathbb{P}\left(B^{(l-1)}\middle|X^{(l)}\right)$. This computation uses the fact that $B^{(l-1)} \perp X^{(l)} \mid X^{(1\to l-1)}$. The problem lies in the fact that the size of the sample space of the conditioning set $X^{(1\to l-1)}$ grows exponentially with *l*. Now consider making the following approximation:

$$B^{(l-1)} \perp X^{(l)} \mid X^{(l-n+1\to l-1)}, n \geq 2 . \quad (5)$$

Regardless of *l*, the conditioning set in (5) will always have a sample space with size $\left|S_X\right|^{n-1}$. Consider the sequence $B^{(0\to l)}$. This sequence, in general, is not a Markov process of some finite order. The memory truncation approximates $B^{(0\to l)}$ by a Markov process of order *n*, $n \in \mathbb{N}$. This is represented by the following:

$$\mathbb{P}\left(B^{(n+k)}\middle|B^{(n+k-1)},\ldots,B^{(0)}\right) \approx \mathbb{P}\left(B^{(n+k)}\middle|B^{(n+k-1)},\ldots,B^{(k)}\right) (6)$$

and corresponds to removing the edges between $X^{(i)}$ and $B^{(i)}$ for $i = 1,...,k$, in the Bayesian network of Fig.3.

We refer to the approximation of (6) as memory truncation of order $n\left(MT^n\right)$.

B. Analysis

We consider a memory truncation of order n, and assume that we have already calculated (approximated) $\mathbb{P}\left(B^{(k)}\right)$, $k \geq n$. For $k = 1, ..., n$, we have the following distributions available:

- $\mathbb{P}\left(B^{(k-1)}, X^{(k)}\right)$
- $\mathbb{P}\left(B^{(k-1)} \middle| X^{(k-n+1 \to k-1)}\right)$
- $\mathbb{P}\left(X^{(k-n+1 \to k)}\right)$

We now derive $\mathbb{P}\left(B^{(k+1)}\right)$. To perform this, we will use the calculation of the joint distribution $\mathbb{P}\left(B^{(k)}, X^{(k+1)}\right)$.

$$\begin{aligned}\mathbb{P}\left(B^{(k)}, X^{(k+1)}\right) &= \sum_{x^{(k-n+2 \to k)}} \mathbb{P}\left(B^{(k)} \mid X^{(k-n+2 \to k)}\right) \mathbb{P}\left(X^{(k+1)} \mid X^{(k-n+2 \to k)}\right) \mathbb{P}\left(X^{(k-n+2 \to k)}\right) \\ &= \sum_{x^{(k-n+2 \to k)}} \mathbb{P}\left(B^{(k)} \mid X^{(k-n+2 \to k)}\right) \mathbb{P}\left(X^{(k-n+2 \to k+1)}\right)\end{aligned} \tag{7}$$

In (7), the distribution $\mathbb{P}\left(B^{(k)} \mid X^{(k-n+2 \to k)}\right)$ is calculated by

$$\mathbb{P}\left(B^{(k)} \mid X^{(k-n+2 \to k)}\right) = \sum_{b^{(k-1)}} \mathbb{P}\left(B^{(k)} \mid X^{(k)}, B^{(k-1)}\right) \mathbb{P}\left(B^{(k-1)} \mid X^{(k-n+2 \to k)}\right),$$

where

$$\begin{aligned}&\mathbb{P}\left(B^{(k-1)} \mid X^{(k-n+2 \to k)}\right) \\ &= \sum_{x^{(k-n+1)}} \mathbb{P}\left(B^{(k-1)} \mid X^{(k-n+1 \to k)}\right) \mathbb{P}\left(X^{(k-n+1)} \mid X^{(k-n+2 \to k)}\right) \\ &= \frac{1}{\mathbb{P}\left(X^{(k-n+2 \to k)}\right)} \sum_{x^{(k-n+1)}} \mathbb{P}\left(B^{(k-1)} \mid X^{(k-n+1 \to k-1)}\right) \mathbb{P}\left(X^{(k-n+1 \to k)}\right) \quad \because \mathbb{P}\left(X^{(k-n+1)} \mid X^{(k-n+2 \to k)}\right) = \frac{\mathbb{P}\left(X^{(k-n+1 \to k)}\right)}{\mathbb{P}\left(X^{(k-n+2 \to k)}\right)}\end{aligned}$$

VI. Simulation Results

In general, the accuracy of $\mathbb{P}\left(B^{(l)}\right)$ increases with increasing the memory truncation order *n*, and so does the complexity. It is however expected that after increasing *n* beyond a certain order n_0, the accuracy improvement would be negligible. The goal is thus to find n_0.

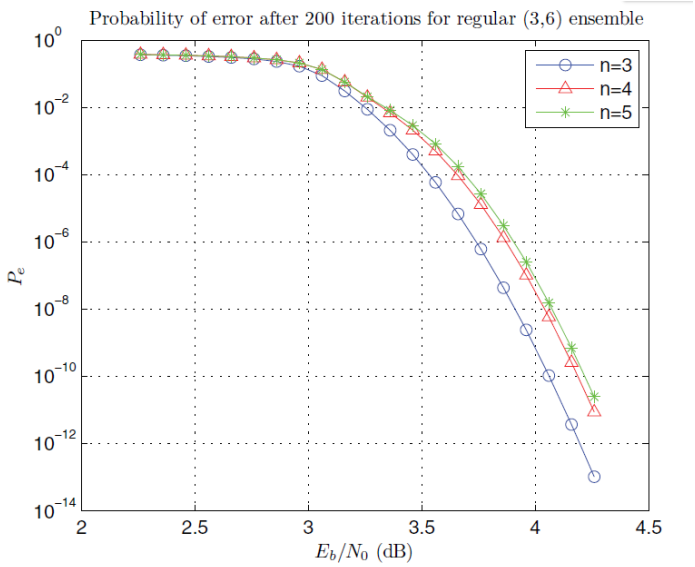


Fig. 4. P_e after 200 iterations vs. E_b/N_0 for the (3,6) ensemble with memory truncation orders 3, 4, and 5.

In Fig. 4, we have shown $P_e^{(l)}$ of the (3, 6) LDPC code ensemble for $l = 200$ vs. E_b / N_0 for different values of memory truncation order *n*. The curves demonstrate a convergence behavior as *n* is increased. In particular, the two curves for *n* = 4 and *n* = 5 are very close. We have also tried a number of other ensembles and observed a similar trend

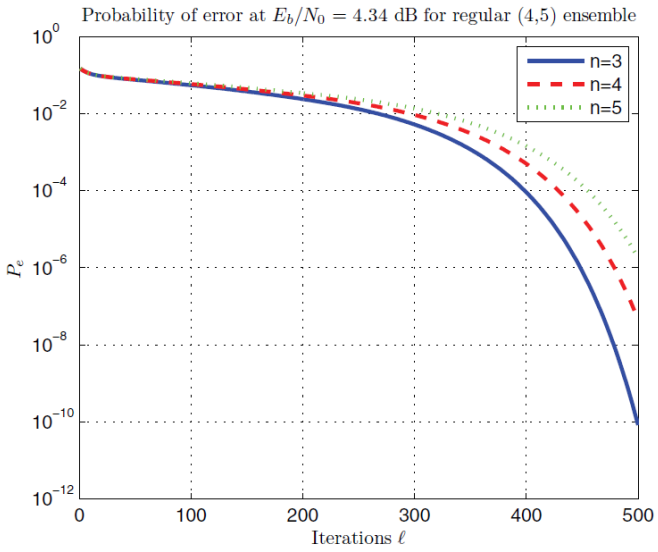


Fig. 5. $P_e^{(\ell)}$ of the (4,5) ensemble vs. ℓ at $E_b/N_0 = 4.34$ dB (above threshold) for memory truncation orders 3, 4, and 5.

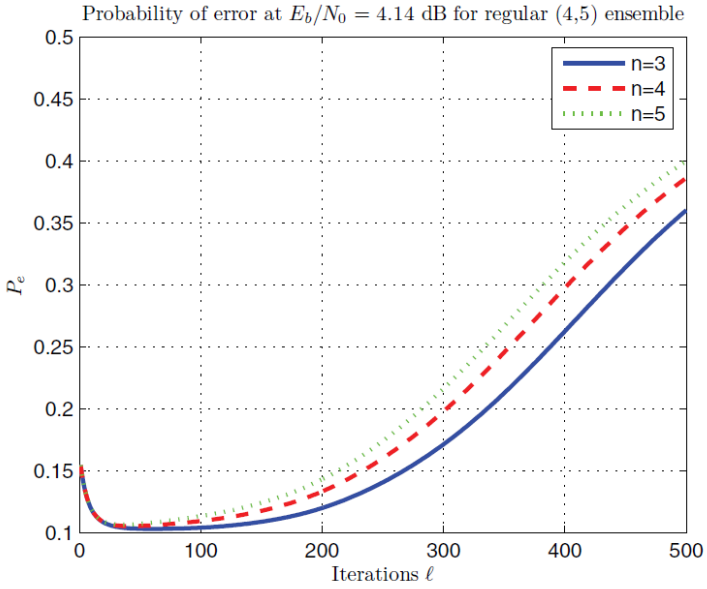


Fig. 6. $P_e^{(\ell)}$ of the (4,5) ensemble vs. ℓ at $E_b/N_0 = 4.14$ dB (below threshold) for memory truncation orders 3, 4, and 5.

For the ensemble of (4, 5) codes, we have plotted $P_e^{(l)}$ vs. l for truncation orders 3, 4 and 5, and for E_b / N_0 values 4.34 dB and 4.14 dB in Figures 5 and 6, respectively. The figures suggest that the ensemble threshold is between the two SNR values.

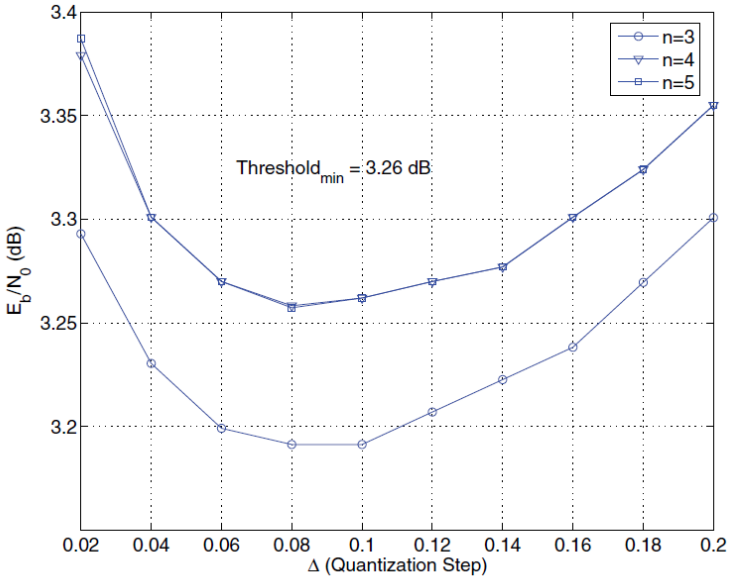


Fig. 7. Threshold values of the $(3, 6)$ ensemble vs. Δ for different memory truncation orders $(q = 8)$.

To clearly see the effect of memory truncation on the calculated thresholds, in Fig. 7, we show the threshold values of the (3, 6) ensemble for different memory truncation orders *n*. The thresholds for each truncation order are plotted versus the quantization step Δ for q = 8.the calculated thresholds for n = 4 and n = 5 are practically identical for different values of Δ. From Fig. 7, the optimal threshold of the (3, 6) ensemble (as a function of Δ) is seen to be about 3.26 dB. Based on the above results, in the following, we use n_0 = 4 to derive the thresholds. In all cases, we use q = 8 and the optimal value of Δ that minimizes the threshold.

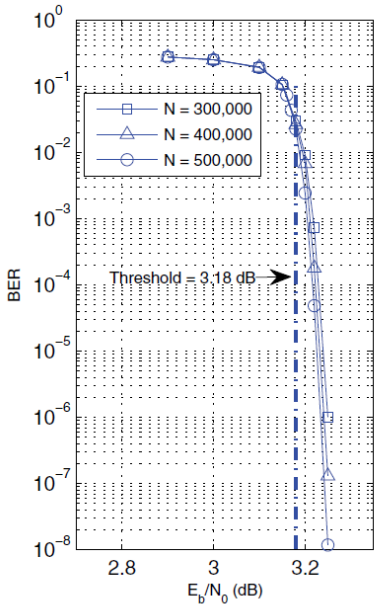


Fig. 8. BER curves of $(5, 10)$ LDPC codes with block lengths 300,000; 400,000; and 500,000 decoded by DD-BMP, and the corresponding threshold $(q = 8)$.

To verify the calculated threshold, for the (5, 10) ensemble, we compare the performance of randomly constructed (5, 10) codes of large block length (N = 300, 000; 400, 000 and 500, 000) with the threshold value of the ensemble (3.18 dB) in Fig. 8.

TABLE I
THRESHOLDS OF BP AND DD-BMP FOR REGULAR ENSEMBLES WITH $d_v = 4$ AND $5 \leq d_c \leq 18$

d_c	$rate$	$\left(\frac{E_b}{N_0}\right)_{BP}$ (dB)	$\left(\frac{E_b}{N_0}\right)_{DD-BMP}$ (dB)	Gap (dB)
5	1/5	2.57	4.24	1.67
6	1/3	1.73	3.18	1.45
7	3/7	1.27	2.88	1.61
8	1/2	1.59	2.80	1.21
9	5/9	1.67	2.82	1.15
10	3/5	1.78	2.87	1.09
11	7/11	1.89	2.93	1.04
12	4/6	1.99	2.99	1.00
13	9/13	2.10	3.06	0.96
14	5/7	2.20	3.13	0.93
15	11/15	2.29	3.20	0.91
16	3/4	2.39	3.27	0.88
17	13/17	2.47	3.33	0.86
18	7/8	2.55	3.40	0.85

TABLE II
THRESHOLDS OF BP AND DD-BMP FOR REGULAR ENSEMBLES WITH $d_v = 6$ AND $7 \leq d_c \leq 18$

d_c	$rate$	$\left(\frac{E_b}{N_0}\right)_{BP}$ (dB)	$\left(\frac{E_b}{N_0}\right)_{DD-BMP}$ (dB)	Gap (dB)
7	1/7	5.14	6.46	1.32
8	1/4	3.52	4.71	1.19
9	1/3	2.90	4.00	1.10
10	2/5	2.61	3.63	1.02
11	5/11	2.47	3.45	0.98
12	1/2	2.41	3.34	0.93
13	7/13	2.39	3.26	0.87
14	6/7	2.40	3.24	0.84
15	3/5	2.43	3.24	0.81
16	5/8	2.46	3.24	0.78
17	11/17	2.50	3.26	0.76
18	2/3	2.54	3.28	0.74

These results show that for a fixed dv, the threshold gap between DD-BMP and BP decreases with increasing the rate. As it can be seen, at higher rates the performance gap is less than 1 dB. In comparison with MS, for codes with larger degrees, DD-BMP outperforms MS

TABLE III
THRESHOLDS OF MS AND DD-BMP FOR RATE-1/2 ENSEMBLES

d_v	d_c	$\left(\frac{E_b}{N_0}\right)_{MS}$ (dB)	$\left(\frac{E_b}{N_0}\right)_{DD-BMP}$ (dB)	Gap (dB)
3	6	1.70	3.26	1.56
4	8	2.49	2.80	0.31
5	10	3.09	3.18	0.09
6	12	3.54	3.34	-0.21
7	14	3.91	3.61	-0.30

These results show that by increasing the degrees, the performance gap between MS and DD-BMP, which is to the advantage of MS for smaller degrees, disappears and then reverses to the advantage of DD-BMP.

These performance results for DD-BMP are impressive considering that both the check node operations and the message-passing for DDBMP are much simpler than those of BP and MS. They also demonstrate the potential of iterative decoding algorithms with memory in achieving better performance/complexity tradeoffs compared to memoryless algorithms.