MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Iterative Decoding Using Replicas

Juntan Zhang, Yige Wang, Marc Fossorier, Jonathan S. Yedidia

TR2005-090 August 2005

Abstract

Replica shuffled versions of iterative decoders of low-density parity-check codes and turbo codes are presented in this paper. The proposed schemes can converge faster than standard and plain shuffled approaches. Two methods, density evolution and EXIT charts, are used to analyze the performance of the proposed algorithms. Both theoretical analysis and simulations show that the new schedules offer good trade-offs with respect to performance, complexity, latency and connectivity.

Publication History:–

1. First printing, TR-2005-090, August 2005

Iterative Decoding Using Replicas

Juntan Zhang, Yige Wang, and Marc Fossorier
Department of Electrical Engineering
University of Hawaii at Manoa, Honolulu, HI 96822
juntan,yige,marc@spectra.eng.hawaii.edu

Jonathan S. Yedidia
Mitsubishi Electric Research Laboratories
201 Broadway, Cambridge, MA 02139
yedidia@merl.com

Abstract

Replica shuffled versions of iterative decoders of low-density parity-check codes and turbo codes are presented in this paper. The proposed schemes can converge faster than the standard and plain shuffled approaches. Two methods, density evolution and EXIT charts, are used to analyze the performance of the proposed algorithms. Both theoretical analysis and simulations show that the new schedules offer good trade-offs with respect to performance, complexity, latency and connectivity.

1 Introduction

Iterative decoding has received significant attention recently, mostly due to its near-Shannon-limit error performance for the decoding of low-density parity-check (LDPC) codes [1, 2] and turbo codes [3]. It uses a symbol-by-symbol soft-in/soft-out decoding algorithm like maximum a posteriori probability (MAP) decoding [4] and processes the received symbols recursively to improve the reliability of each symbol based on constraints that specify the code. In the first iteration, the decoder only uses the channel output as input, and generates a soft output for each symbol. Subsequently, the output reliability measures of the decoded symbols at the end of each decoding iteration are used as inputs for the next iteration. The decoding iteration process continues until a certain stopping condition is satisfied. Then hard decisions are made based on the output reliability measures of decoded symbols from the last decoding iteration.

In order to take advantage of the more reliable extrinsic messages available within one iteration, a shuffled turbo decoding method has been proposed in [5]. The shuffled turbo decoding algorithm converges faster and needs approximately the same computational complexity as standard parallel turbo decoding. Schemes using the shuffled principle were also proposed for decoding LDPC codes and have been shown to converge faster than the corresponding standard decoding [6]−[8]. The aim of this work is to develop replica shuffled versions of the standard iterative decoding algorithms for LDPC codes and turbo codes. By using replica subdecoders, this method provides a faster convergence than plain shuffled decoding at the expense of higher complexity. In [9], parallelism within one iteration is achieved by proper interleaver design for the turbo decoder architecture. In this work, iterations themselves are parallelized and consequently, the two approaches

can be combined. The new approach is analyzed by density evolution [10] and EXIT charts [11]−[13]. Both methods show that shuffled belief propagation (BP) converges about twice as fast as the standard BP and replica shuffled BP converges faster than the plain shuffled BP. The convergence speed of the replica shuffled BP is determined by the number of subdecoders and the information updating schemes. For turbo decoding, replica shuffled turbo decoding converges faster than both plain shuffled turbo decoding and standard parallel turbo decoding. It is worth mentioning that the proposed schemes are sequential in nature. Therefore they are mainly interesting when the structure of a code makes it difficult to implement the decoding in hardware in a fully parallel way (e.g., long LDPC codes, LDPC codes with relatively dense connectivity such as finite geometry LDPC codes or turbo codes).

2 Iterative decoding of LDPC codes

In general, LDPC codes can be categorized into regular LDPC codes and irregular LDPC codes. Both can be represented by a bipartite graph with N variable nodes on the left and M check nodes on the right. This bipartite graph can be specified by the sequences $(\lambda_1, \lambda_2, \cdots, \lambda_{d_v})$ and $(\rho_1, \rho_2, \cdots, \rho_{d_c})$, where $\lambda_i(\rho_i)$ represents the fraction of edges with left (right) degree i, and d_v and d_c are the maximum variable degree and check degree, respectively.

2.1 Algorithms

Following the definitions in [14], deterministic schedulings can be implemented either based on horizontal [15, 16] or vertical partitioning [6, 7] of the parity check matrix. In [15, 16] a horizontal partitioning was proposed to serialize the decoding of LDPC codes and in the process, speed-up of the convergence was achieved. The algorithms of [6, 7] directly intend to speed up BP or its simplified versions by combining the bit node and check node processings in their scheduling. In this work, we consider replica approaches based on a vertical partitioning to speed up the decoding. The similar replica principle can be applied to the horizontal partitioning in a straightforward way and similar gains are observed for both partitioning schedules.

2.1.1 Standard BP decoding of LDPC codes

Suppose a regular binary $(N, K)(d_v, d_c)$ LDPC code $\mathbf{C}$ of length N and dimension K is used for error control over an AWGN channel with zero mean and power spectral density $N_0/2$. Assume BPSK signaling with unit energy, which maps a codeword $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ into a transmitted sequence $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, according to $x_n = 1 - 2c_n$, for $n = 1, 2, \ldots, N$. If $\mathbf{c} = [c_n]$ is a codeword in $\mathbf{C}$ and $\mathbf{x} = [x_n]$ is the corresponding transmitted sequence, then the received sequence is $\mathbf{x} + \mathbf{n} = \mathbf{y} = [y_n]$, with $y_n = x_n + n_n$, where for $1 \leq n \leq N$, n_n's are statistically independent Gaussian random variables $\mathcal{N}(0, N_0/2)'s$ with zero mean and variance $N_0/2$. Let $\mathbf{H} = [H_{mn}]$ be the parity check matrix which defines the LDPC code. We denote the set of bits that participate in check m by $\mathcal{N}(m) = \{n : H_{mn} = 1\}$ and the set of checks in which bit n participates as $\mathcal{M}(n)$

$= \{m : H_{mn} = 1\}$. We also denote $\mathcal{N}(m)\backslash n$ as the set $\mathcal{N}(m)$ with bit n excluded, and $\mathcal{M}(n)\backslash m$ as the set $\mathcal{M}(n)$ with check m excluded. We define the following notations associated with the ith iteration:

- $U_{ch,n}$: The log-likelihood ratio (LLR) of bit n which is derived from the channel output y_n. In BP decoding, we initially set $U_{ch,n} = \frac{4}{N_0} y_n$.
- $U_{mn}^{(i)}$: The LLR of bit n which is sent from the check node m to bit node n.
- $V_{mn}^{(i)}$: The LLR of bit n which is sent from the bit node n to check node m.
- $V_n^{(i)}$: The *a posteriori* LLR of bit n.

The standard BP algorithm is carried out as follows [2]:

Initialization: Set $i = 1$, and the maximum number of iteration to I_{Max}. For each m,n, set $V_{mn}^{(0)} = U_{ch,n}$.

Step 1: **(i)** Horizontal Step, for $1 \leq n \leq N$ and each $m \in \mathcal{M}(n)$, process:

$$U_{mn}^{(i)} = 2 \tanh^{-1} \left(\prod_{n' \in \mathcal{N}_{(m)\backslash n}} \tanh \frac{V_{mn'}^{(i-1)}}{2} \right). \tag{1}$$

(ii) Vertical Step, for $1 \leq n \leq N$ and each $m \in \mathcal{M}(n)$, process:

$$V_{mn}^{(i)} = U_{ch,n} + \sum_{m' \in \mathcal{M}(n)\backslash m} U_{m'n}^{(i)} \tag{2}$$

$$V_n^{(i)} = U_{ch,n} + \sum_{m \in \mathcal{M}(n)} U_{mn}^{(i)}.$$

Step 2: Hard decision and stopping criterion test:

(i) Create $\hat{\boldsymbol{c}}^{(i)} = [\hat{c}_n^{(i)}]$ such that $\hat{c}_n^{(i)} = 1$ if $V_n^{(i)} < 0$, and $\hat{c}_n^{(i)} = 0$ if $V_n^{(i)} \geq 0$.

(ii) If $\boldsymbol{H}\hat{\boldsymbol{c}}^{(i)} = \boldsymbol{0}$ or I_{Max} is reached, stop the decoding iteration and go to Step 3. Otherwise set $i := i + 1$ and go to Step 1.

Step 3: Output $\hat{\boldsymbol{c}}^{(i)}$ as the decoded codeword.

2.1.2 Plain shuffled BP decoding of LDPC codes

In general, for both check-to-bit messages and bit-to-check messages, the more independent information is used to update the messages, the more reliable they become. Iteration-i of the standard two-step implementation of the BP algorithm uses all values $V_{mn'}^{(i-1)}$ computed at the previous iteration in (1). However certain values $V_{mn'}^{(i)}$ could already be computed based on a partial computation of the values $U_{mn}^{(i)}$ obtained from (2),

and then be used instead of $V_{mn'}^{(i-1)}$ in (1) to compute the remaining values $U_{mn}^{(i)}$. This suggests a shuffling of the horizontal and vertical steps of standard BP decoding. This is referred to as shuffled BP decoding.

In the shuffled BP algorithm, the initialization, stopping criterion test and output steps remain the same as in the standard BP algorithm. The only difference between the two algorithms lies in the updating procedure. Step 1 of the shuffled BP algorithm is modified as: for $1 \leq n \leq N$ and each $m \in \mathcal{M}(n)$, process the horizontal step and vertical step **jointly**, with (1) modified as [5]:

$$U_{mn}^{(i)} = 2\tanh^{-1}\Bigg(\prod_{\substack{n' \in \mathcal{N}_{(m)\backslash n} \\ n'<n}} \tanh\frac{V_{mn'}^{(i)}}{2} \prod_{\substack{n' \in \mathcal{N}_{(m)\backslash n} \\ n'>n}} \tanh\frac{V_{mn'}^{(i-1)}}{2}\Bigg). \tag{3}$$

Note that (3) has a similar form as the forward-backward algorithm of [4].

2.1.3 Replica shuffled BP decoding of LDPC codes

Shuffled BP decoding is a bit-based sequential approach and the method described in Section 2.1.2 is based on a natural increasing order, i.e, the messages at bit nodes are updated according to the order $n = 1, 2, \ldots, N$. The larger the value of n, the more independent pieces of information are used to update the messages at bit n and the more reliable these messages become. Therefore, as the index n increases, the reliability of the bit decisions increases and the corresponding error rate decreases. Indeed, the same reasoning applies if shuffled BP decoding is performed in reverse order; hence if shuffled BP decoding is employed using a bit order starting with bit N and ending with bit 1, the error rate increases with the index n. As illustration, Figure 1 depicts the number of bit errors using standard and shuffled BP decodings (with increasing and decreasing order) for the (273,191) PG-LDPC code [17] at the SNR of 3.0 dB and after the second iteration. A total of 10000 random blocks were decoded. From Figure 1, we observe that in plain shuffled BP decoding, the later a bit is processed, the more reliable it is. If more decoders are used, they can exchange their most reliable messages (bit-to-check beliefs associated with bits corresponding to the lower part of shuffled decoding curve) with one another and achieve faster convergence. Based on this observation, replica shuffled BP decoding is developed next.

In replica shuffled BP decoding, several shuffled subdecoders based on different updating orders operate simultaneously and cooperatively. After each iteration, each subdecoder receives more reliable messages from and sends more reliable messages to other subdecoders. Based on these more reliable messages, all replica subdecoders begin the next iteration. Hence replica decoding can be viewed as a way to parallelize iterations. For two replicas, let $\overrightarrow{D}$ and $\overleftarrow{D}$ denote the subdecoders with natural increasing and decreasing updating orders, respectively. Let $\overrightarrow{U}_{mn}^{(i)}$ and $\overrightarrow{V}_{mn}^{(i)}$ be the variables associated with $\overrightarrow{D}$ at iteration i. The variables associated with $\overleftarrow{D}$ are defined in a similar way. The replica shuffled BP decoding with two replica subdecoders is carried out as follows:

Initialization: Set $i = 1$, and the maximum number of iteration to I_{Max}. For each m,n, set $\overrightarrow{V}_{mn}^{(0)} = \overleftarrow{V}_{mn}^{(0)} = U_{ch,n}$.

Step 1: Each replica subdecoder processes the following two steps simultaneously. For $1 \leq n \leq N$ and each $m \in \mathcal{M}(n)$, process

(i) Horizontal Step

$$\overrightarrow{U}_{mn}^{(i)} = 2\tanh^{-1}\left(\prod_{\substack{n' \in \mathcal{N}_{(m)\backslash n} \\ n'<n}} \tanh\frac{\overrightarrow{V}_{mn'}^{(i)}}{2} \prod_{\substack{n' \in \mathcal{N}_{(m)\backslash n} \\ n'>n}} \tanh\frac{\overrightarrow{V}_{mn'}^{(i-1)}}{2}\right)$$

$$\overleftarrow{U}_{mn}^{(i)} = 2\tanh^{-1}\left(\prod_{\substack{n' \in \mathcal{N}_{(m)\backslash n} \\ n'>n}} \tanh\frac{\overleftarrow{V}_{mn'}^{(i)}}{2} \prod_{\substack{n' \in \mathcal{N}_{(m)\backslash n} \\ n'<n}} \tanh\frac{\overleftarrow{V}_{mn'}^{(i-1)}}{2}\right)$$

(ii) Vertical Step

$$\overrightarrow{V}_{mn}^{(i)} = U_{ch,n} + \sum_{m' \in \mathcal{M}(n)\backslash m} \overrightarrow{U}_{m'n}^{(i)}$$

$$\overleftarrow{V}_{mn}^{(i)} = U_{ch,n} + \sum_{m' \in \mathcal{M}(n)\backslash m} \overleftarrow{U}_{m'n}^{(i)}$$

Step 2: Set $\overrightarrow{V}_{mn}^{(i)} = \overleftarrow{V}_{mn}^{(i)}$ for $1 \leq n \leq N/2$ and $\overleftarrow{V}_{mn}^{(i)} = \overrightarrow{V}_{mn}^{(i)}$ for $N/2 < n \leq N$.

Step 3: Hard decision and stopping criterion test:

(i) Create $\hat{\boldsymbol{c}}^{(i)} = [\hat{c}_n^{(i)}]$ such that for $1 \leq n \leq N/2$, $\hat{c}_n^{(i)} = 1$ if $U_{ch,n} + \sum_{m \in \mathcal{M}(n)} \overleftarrow{U}_{mn}^{(i)} < 0$, and $\hat{c}_n^{(i)} = 0$ otherwise; for $N/2 < n \leq N$, $\hat{c}_n^{(i)} = 1$ if $U_{ch,n} + \sum_{m \in \mathcal{M}(n)} \overrightarrow{U}_{mn}^{(i)} < 0$, and $\hat{c}_n^{(i)} = 0$ otherwise.

(ii) If $\boldsymbol{H}\hat{\boldsymbol{c}}^{(i)} = \boldsymbol{0}$ or I_{Max} is reached, stop the decoding iteration and go to Step 4. Otherwise set $i := i + 1$ and go to Step 1.

Step 4: Output $\hat{\boldsymbol{c}}^{(i)}$ as the decoded codeword.

With respect to Figure 1, note that Step 2 is equivalent to keep the lower parts of the two shuffled BP curves.

Another possible implementation is that these two subdecoders exchange more reliable messages synchronously with each other during the decoding process. Define $R(n) = \{n' | n \leq n' \leq N - n\}$, and $\overline{R}(n) = \{n' | 1 \leq n' \leq N, n' \notin R(n)\}$, for $1 \leq n \leq N$. In synchronous scheme, the updating and exchanging procedures operate simultaneously as follows:

Step 1: For $1 \leq n \leq N$ and each $m \in \mathcal{M}(n)$, for $p = N - n$ and each $q \in \mathcal{M}(p)$, two replica subdecoders process the following two steps simultaneously

(i) Horizontal Step

$$U_{mn}^{(i)} = 2\tanh^{-1}\left(\prod_{\substack{n'\in\mathcal{N}_{(m)\backslash n}\\ n'\in\overline{R}(n)}}\tanh\frac{V_{mn'}^{(i)}}{2}\prod_{\substack{n'\in\mathcal{N}_{(m)\backslash n}\\ n'\in R(n)}}\tanh\frac{V_{mn'}^{(i-1)}}{2}\right)$$

$$U_{qp}^{(i)} = 2\tanh^{-1}\left(\prod_{\substack{p'\in\mathcal{N}_{(q)\backslash p}\\ p'\in\overline{R}(N-p)}}\tanh\frac{V_{qp'}^{(i)}}{2}\prod_{\substack{p'\in\mathcal{N}_{(q)\backslash p}\\ p'\in R(N-p)}}\tanh\frac{V_{qp'}^{(i-1)}}{2}\right).$$

(ii) Vertical Step

$$V_{mn}^{(i)} = U_{ch,n} + \sum_{m'\in\mathcal{M}(n)\backslash m} U_{m'n}^{(i)}$$

$$V_{qp}^{(i)} = U_{ch,p} + \sum_{q'\in\mathcal{M}(p)\backslash q} U_{q'p}^{(i)}.$$

Notice that in this case the two replica subdecoders use the same set of bit-to-check LLR values. It is also straightforward to extend the replica shuffled BP decoding to the cases in which more than two replica subdecoders are used.

2.1.4 Group replica shuffled BP decoding of LDPC codes

To take advantage of as many newly delivered messages as possible and therefore to achieve the best performance, a fully serial replica shuffled BP is necessary. However, this scheme is not attractive for hardware implementation due to its serial nature. A totally parallel implementation is not realistic either for large code lengths, or codes with highly connected graph.

In [5], a method called "group shuffled" BP was presented. In group shuffled BP, the bits of a codeword are processed in groups in a semi-parallel manner. The groups are processed serially while the bits within a group are processed in parallel. This approach can be extended in a straightforward way to the design of group replica shuffled BP decoders. Assume the N bits of a codeword are divided into G groups and each group contains $\frac{N}{G} = B$ bits (assuming $N \bmod G = 0$ for simplicity). Step 1 of the non-synchronous group replica shuffled BP algorithm is carried out as follows:

Step 1: For $1 \le g \le G$, each replica subdecoder processes jointly the following two steps

(i) Horizontal step: for $(g-1)\cdot B+1 \le n \le g\cdot B$ and each $m \in \mathcal{M}(n)$, process:

$$\overrightarrow{U}_{mn}^{(i)} = 2\tanh^{-1}\left(\prod_{\substack{n'\in\mathcal{N}_{(m)\backslash n}\\ n'\le(g-1)\cdot B}}\tanh\frac{\overrightarrow{V}_{mn'}^{(i)}}{2}\prod_{\substack{n'\in\mathcal{N}_{(m)\backslash n}\\ n'\ge(g-1)\cdot B+1}}\tanh\frac{\overrightarrow{V}_{mn'}^{(i-1)}}{2}\right)$$

$$\overleftarrow{U}_{mn}^{(i)} = 2\tanh^{-1}\left(\prod_{\substack{n'\in\mathcal{N}_{(m)\backslash n}\\ n'\geq(G-g+1)\cdot B+1}} \tanh\frac{\overleftarrow{V}_{mn'}^{(i)}}{2}\prod_{\substack{n'\in\mathcal{N}_{(m)\backslash n}\\ n'\leq(G-g+1)\cdot B}} \tanh\frac{\overleftarrow{V}_{mn'}^{(i-1)}}{2}\right)$$

(ii) Vertical Step: for $(g-1)\cdot B+1 \leq n \leq g\cdot B$ and each $m\in\mathcal{M}(n)$, process:

$$\overrightarrow{V}_{mn}^{(i)} = U_{ch,n} + \sum_{m'\in\mathcal{M}(n)\backslash m} \overrightarrow{U}_{m'n}^{(i)}$$

$$\overleftarrow{V}_{mn}^{(i)} = U_{ch,n} + \sum_{m'\in\mathcal{M}(n)\backslash m} \overleftarrow{U}_{m'n}^{(i)}$$

Synchronous group replica shuffled decoding is defined in a similar way.

2.2 Analysis by density evolution

2.2.1 Density evolution of shuffled BP

Density evolution [10] is an effective numerical method to analyze the performance of message passing iterative decoding algorithms based on graph. It has been shown that for a given message-passing decoding, if the channel and the decoder satisfy the symmetry conditions [10], then the decoding bit error rate is independent of the transmitted sequence. The process of density evolution therefore can be greatly simplified by assuming the all-zero sequence is transmitted. It is straightforward to verify that shuffled and replica shuffled BP decoder satisfy the symmetry condition, so that the all-zero transmitted codeword assumption is valid. In density evolution of shuffled and replica shuffled BP, a cycle-free structure of the LDPC code graph is assumed as in [10]. In this case, the incoming messages to any bit or check node are independent, which also simplifies the derivation of the probability density functions (pdf's) of the outgoing messages. Next we present density evolution for shuffled and replica shuffled BP decoding. Density evolution results for serial BP decoding of LDPC codes can also be found in [8].

In shuffled and replica shuffled BP decoding, the pdf's of outgoing and incoming messages of bit nodes depend on the bit index number n. Let $f_{U_n}^{(i)}(u)$ and $f_{V_n}^{(i)}(v)$ be the pdf's of the incoming and outgoing messages of bit node n at iteration i, respectively. In standard BP, infinite codeword length is assumed while in shuffled BP we consider a large enough codeword length N and assume the cycle-free condition still holds.

For the bit node processor of shuffled BP, the density evolution is the same as that of standard BP, so that for $n = 1, 2, \ldots, N$,

$$f_{V_n}^{(i)} = \mathcal{F}^{-1}\left(\mathcal{F}(f_{U_{ch}})\cdot\left(\mathcal{F}(f_{U_n}^{(i)})\right)^{dc-1}\right) \tag{4}$$

where $\mathcal{F}$ denotes the Fourier transform operator.

As observed from (3), $U_n^{(i)}$ depends on both $V_{n'}^{(i)}$ for $n' < n$ and $V_{n'}^{(i-1)}$ for $n' > n$. To avoid a brute force calculation of all possible combinatorial formats of $V_{n'}^{(i)}$ and $V_{n'}^{(i-1)}$,

we let the average pdf of the newly delivered incoming messages to check nodes adjacent to bit node n at iteration i be

$$f^{(i)}_{\overline{V}_{n'<n}}(v) = \frac{1}{n-1}\sum_{n'=1}^{n-1} f^{(i)}_{V_{n'}}(v). \tag{5}$$

Similarly, we let the average pdf of the incoming messages from bit nodes $\{b_{n'}|n' > n\}$ to check nodes adjacent to bit node n be

$$f^{(i-1)}_{\overline{V}_{n'>n}}(v) = \frac{1}{N-n}\sum_{n'=n+1}^{N} f^{(i-1)}_{V_{n'}}(v). \tag{6}$$

The check node processing can be implemented in a recursive way [18]. Define a core operation as

$$\Psi(V_1, V_2) \;=\; 2\tanh^{-1}\left(\tanh\left(\frac{V_1}{2}\right)\tanh\left(\frac{V_2}{2}\right)\right). \tag{7}$$

Then (1) can be calculated by applying (7) recursively as

$$U \;=\; \Psi(\ldots\Psi(\Psi(V_1, V_2), V_3), \ldots, V_{d_c-1}). \tag{8}$$

If the incoming messages are i.i.d. random variables with pdf $f_V(v)$, the pdf of the outgoing message can be efficiently computed as [18]

$$f_U \;=\; \Psi^{d_c-1} f_V. \tag{9}$$

Let us consider plain shuffled BP with natural increasing ordering. For a belief message incoming to bit node n, the incoming messages to the check node adjacent to bit node n have in total

$$\binom{N-1}{d_c-1} \tag{10}$$

possible formats. For each $j = 0, 1, \ldots, d_c - 1$, there are

$$\binom{n-1}{j} \cdot \binom{N-n}{d_c-1-j} \tag{11}$$

possible formats which contain j newly delivered bit-to-check messages at the current iteration and $d_c - 1 - j$ bit-to-check messages delivered at the previous iteration. The average pdf incoming to bit node n at iteration i becomes

$$f^{(i)}_{U_n} \;=\; \sum_{j=0}^{d_c-1} \frac{\binom{n-1}{j}\cdot\binom{N-n}{d_c-1-j}}{\binom{N-1}{d_c-1}} \cdot \Psi^j f^{(i)}_{\overline{V}_{n'<n}} \cdot \Psi^{d_c-1-j} f^{(i-1)}_{\overline{V}_{n'>n}}. \tag{12}$$

Theorem 3.2.2 in [8] also provides a recursion for density evolution of a serial schedule. In [8], the variable nodes are divided into m_v sets of equal size. Based on the assumption that no two variable nodes in a set are connected to the same check node, density evolution is simplified and only m_v recursions are needed. In our method, every variable node is processed and the average pdf's are computed based on a combinatorial analysis, thus no specific assumption of the graphical structure is required. Although our approach needs more calculations, it is independent of the code structure.

2.2.2 Density evolution of replica shuffled BP

It is straightforward to extend these updating rules of pdf's for shuffled BP to replica shuffled BP. For instance, in non-synchronous replica shuffled BP with two subdecoders, the updating rule of the pdf's of the outgoing belief messages from bit nodes is the same as that in plain shuffled BP, while the pdf's of incoming belief messages to bit nodes are modified as

$$f_{V_{N+1-n}}^{(i)} \leftarrow f_{V_n}^{(i)} \tag{13}$$

for $N/2 \leq n \leq N$. Density evolution of synchronous replica shuffled BP operates in the same way while updating pdf's of incoming belief messages to bit nodes synchronously, i.e.,

$$f_{V_{N+1-n}}^{(i-1)} \leftarrow f_{V_n}^{(i)} \tag{14}$$

for $1 \leq n \leq N/2$, and

$$f_{V_{N+1-n}}^{(i)} \leftarrow f_{V_n}^{(i)} \tag{15}$$

for $N/2 < n \leq N$. The density evolution of replica shuffled BP with more than two subdecoders can be obtained in a similar way.

The extension of density evolution of shuffled and replica shuffled BP for decoding irregular LDPC codes is also straightforward. Consider an irregular LDPC code with degree distributions $\lambda(x) = \sum_{l=1}^{dv} \lambda_l x^{l-1}$ and $\rho(x) = \sum_{l=1}^{dc} \rho_l x^{l-1}$. Consider plain shuffled BP decoding in natural increasing order. From (12), at iteration i, the pdf of incoming messages to bit node n from a check node with degree l is

$$f_{U_{n,l}}^{(i)} = \sum_{j=0}^{l-1} \frac{\binom{n-1}{j} \cdot \binom{N-n}{l-1-j}}{\binom{N-1}{l-1}} \cdot \Psi^j f_{\overline{V}_{n'<n}}^{(i)} \cdot \Psi^{l-1-j} f_{\overline{V}_{n'>n}}^{(i-1)}. \tag{16}$$

Since the pdf's of the outgoing messages of check nodes with different degree are distinct, the expectation of these pdf's is the overall pdf of the messages incoming to bit node n

$$f_{U_n}^{(i)} = \sum_{l=1}^{dc} \rho_l \sum_{j=0}^{l-1} \frac{\binom{n-1}{j} \cdot \binom{N-n}{l-1-j}}{\binom{N-1}{l-1}} \cdot \Psi^j f_{\overline{V}_{n'<n}}^{(i)} \cdot \Psi^{l-1-j} f_{\overline{V}_{n'>n}}^{(i-1)}.$$

Similarly, the pdf of outgoing messages from bit node n at iteration i becomes

$$f_{V_n}^{(i)} = \sum_{l=1}^{dv} \lambda_l \mathcal{F}^{-1}\left(\mathcal{F}(f_{U_{ch}}) \cdot \left(\mathcal{F}(f_{U_n}^{(i)})\right)^{l-1}\right). \tag{17}$$

2.2.3 Simulation results

Figure 2 depicts the bit error rate (BER) as a function of the numbers of decoding iterations predicted by density evolution with standard BP, shuffled BP, replica shuffled BP with two and four subdecoders (synchronous exchanging) methods, for decoding rate-1/2 $(3, 6)$ regular LDPC codes with $E_b/N_o = 1.111$ dB. We observe that shuffled BP converges about twice as fast as the standard BP decoding while replica shuffled

BP converges faster than plain shuffled BP. As expected, we observe that the larger the number of subdecoders in replica shuffled BP, the faster the convergence of decoding.

Figure 3 depicts the BER versus the number of iterations predicted by density evolution with replica shuffled BP decoder of two subdecoders using non-synchronous and synchronous exchanging schemes, for a $(3, 6)$ regular LDPC code. We observe that replica shuffled BP under the synchronous exchanging scheme converges faster than under the non-synchronous exchanging schedule. It is also worth mentioning that the synchronous scheme requires less memory than the non-synchronous scheme, but more frequent memory access.

Figure 4 depicts the BER as a function of the numbers of decoding iterations predicted by density evolution with standard BP, shuffled BP, replica shuffled BP with two and four subdecoders (synchronous exchanging) methods, for decoding a rate-1/2 irregular LDPC code over an AWGN channel with $E_b/N_o = 0.409$dB. The check and bit nodes distributions of this code are $\rho(x) = 0.63676x^6 + 0.36324x^7$ and $\lambda(x) = 0.25105x + 0.30938x^2 + 0.00104x^3 + 0.43853x^9$, respectively [19]. We observe a similar behavior as in the case of regular LDPC codes.

Figure 5 depicts the BER versus the decrease in BER predicted by density evolution with standard BP and replica shuffled BP with four subdecoders, for decoding the above irregular LDPC code at the SNR 0.409dB. We observe that at a given probability of error, the decrease of the probability of error with replica shuffled BP is always larger than that of standard BP, which illustrates the faster convergence property of replica shuffled BP from another perspective. We also observe that density evolution of replica shuffled BP with four subdecoders has three fixed points, which is the same as that of standard BP. We observe a similar behavior for plain shuffled BP and replica shuffled BP with two subdecoders.

2.3 Analysis by EXIT chart

EXIT chart [11]-[13] is another effective technique to study the convergence behavior of iterative decoding. It is easy to visualize, to program and it is a good complement to density evolution. Both the variable node and check node EXIT curves can be computed in closed form [20] for the standard BP decoding. Let I_U be the average mutual information between the bits on the edges of the graph and the a priori (extrinsic) LLRs of the variable (check) nodes. Similarly let I_V be that between the bits on the edges of the graph and the extrinsic (a priori) LLRs of the variable (check) nodes. Then the EXIT functions of a degree$-d_v$ variable node and a degree$-d_c$ check node are respectively

$$I_{V,STD}\left(I_U, d_v, \frac{E_b}{N_0}, R\right) = J\left(\sqrt{(d_v - 1)[J^{-1}(I_U)]^2 + \sigma_{ch}^2}\right) \tag{18}$$

$$I_{U,STD}\left(I_V, d_c\right) \approx 1 - J\left(\sqrt{d_c - 1} \cdot J^{-1}(1 - I_V)\right) \tag{19}$$

where $\sigma_{ch}^2 = 8R \cdot \frac{E_b}{N_0}$ and the functions $J(\cdot)$ and $J^{-1}(\cdot)$ are given in the Appendix of [20].

2.3.1 EXIT chart of plain shuffled BP

In order to find a closed form for the shuffled BP decoding, the following ideal model is constructed for a regular LDPC code. Suppose the variable nodes can be divided into d_c sets and those in the ith set only connect to the ith edge of the check nodes. For example, this kind of structure can be approximately obtained when the progressive edge-growth (PEG) method [21] is used to construct the code. Since all the edges of the variable nodes in the same set connect to different check nodes, they can not benefit from one another. However they can equally make use of the updated information of the previous edges. The processing of each check node also becomes identical. Let the mutual information between the bits on any edge connected to a check node and their corresponding a priori LLRs be equal to the average input mutual information I_V. Let I'_{V_i} be the updated mutual information between the bit on the ith edge of the same check node and its a priori LLRs. Denote I_{U_i} as the mutual information between the bit on the ith edge of this check node and its extrinsic LLRs. Then the EXIT function for a check node of a (d_v, d_c) regular LDPC code decoded with shuffled BP decoding is

$$I_{U,SHF}\left(I_V, d_c\right) = \frac{1}{d_c}\sum_{i=1}^{d_c} I_{U_i} \tag{20}$$

$$I_{U_i} = I_{U,STD}\left(\frac{(d_c - i)I_V + \sum_{k=1}^{i-1} I'_{V_k}}{d_c - 1}, d_c\right) \tag{21}$$

$$I'_{V_i} = I_{V,STD}\left(I_{U_i}, d_v, \frac{E_b}{N_0}, R\right). \tag{22}$$

Since the input mutual information of the variable nodes in different sets are different, denote them as $I_{U_1}, \ldots, I_{U_{d_c}}$, respectively. Then the average input mutual information of all the variable nodes is $I_{U_{av}} = \sum_{i=1}^{d_c} I_{U_i}/d_c$ and the average output mutual information is $I_{V_{av}} = \sum_{i=1}^{d_c} I_{V,STD}(I_{U_i}, d_v, \frac{E_b}{N_0}, R)/d_c$. The EXIT function for a variable node in the shuffled BP decoding is given by

$$I_{V,SHF}\left(I_{U_{av}}, d_v, \frac{E_b}{N_0}, R\right) = I_{V_{av}}. \tag{23}$$

Next, we compare $I_{V,STD}$ and $I_{V,SHF}$. Let $J_1(\sigma^2) = J(\sigma)$ and $I_{U_i} = J_1(\sigma_i^2)$. Since $J_1(\sigma^2)$ is approximately linear with σ^2 when σ^2 is within a small range, we obtain in that case $I_{U_{av}} = \sum_{i=1}^{d_c} I_{U_i}/d_c = \sum_{i=1}^{d_c} J_1(\sigma_i^2)/d_c \approx J_1(\frac{1}{d_c}\sum_{i=1}^{d_c}\sigma_i^2)$. Therefore, it follows

$$\begin{aligned} I_{V,STD}\left(I_{U_{av}}, d_v, \frac{E_b}{N_0}, R\right) &= J_1\left((d_v - 1)J_1^{-1}(I_{U_{av}}) + \sigma_{ch}^2\right) \\ &\approx J_1\left((d_v - 1)\left(\frac{1}{d_c}\sum_{i=1}^{d_c}\sigma_i^2\right) + \sigma_{ch}^2\right) \\ &= J_1\left(\frac{1}{d_c}\sum_{i=1}^{d_c}\left((d_v - 1)\sigma_i^2 + \sigma_{ch}^2\right)\right) \end{aligned}$$

$$\approx \frac{1}{d_c}\sum_{i=1}^{d_c} J_1\left((d_v-1)\sigma_i^2+\sigma_{ch}^2\right)$$

$$= \frac{1}{d_c}\sum_{i=1}^{d_c} I_{V,STD}\left(I_{U_i}, d_v, \frac{E_b}{N_0}, R\right)$$

$$= I_{V,SHF}\left(I_{U_{av}}, d_v, \frac{E_b}{N_0}, R\right).$$

From simulations, we observe that the variances σ_i^2 of the a priori inputs to different variable nodes at one iteration vary within a small range. Hence the EXIT function for a variable node in shuffled BP decoding is almost the same as that in standard BP decoding.

2.3.2 EXIT chart of replica shuffled BP

It is straightforward to extend this method to replica shuffled BP. Using a similar approach, we can prove that the EXIT function for a variable node in replica shuffled BP decoding is also almost the same as that in standard BP decoding. Since in the non-synchronous scheme, subdecoders only exchange information at the end of each iteration, the EXIT function for a check node in replica shuffled BP with two subdecoders and the non-synchronous updating can be written as

$$I_{U,REP_2,NS}\left(I_V, d_c\right) = \frac{1}{d_c}\sum_{i=d_c/2}^{d_c} 2I_{U_i} \quad (even\ d_c) \tag{24}$$

$$I_{U,REP_2,NS}\left(I_V, d_c\right) = \frac{1}{d_c}\left(\sum_{i=\lceil d_c/2\rceil+1}^{d_c} 2I_{U_i} + I_{U_{\lceil d_c/2\rceil}}\right) (odd\ d_c). \tag{25}$$

The EXIT function for a check node in replica shuffled BP with more than two subdecoders can be obtained in a similar way.

In the synchronous scheme, subdecoders exchange information immediately. Suppose D subdecoders are used. Then we can divide each of the d_c sets of the ideal model into D subsets. Each subdecoder processes the variable nodes in a distinct subset of the same set at the same time. After all the variable nodes have been processed once, the subdecoders go back to the first set and process a subset different from those they have already processed. Then the replica shuffled BP can be regarded as applying the shuffled BP D times. Therefore the EXIT function for a check node in the synchronous scheme with D subdecoders is given by

$$I_{U,REP_D,S}\left(I_V, d_c\right) = I_{U,SHF}\left(I_{V_D}, d_c\right) \tag{26}$$

$$I_{V_i} = I_{V,SHF}\left(I_{U,SHF}\left(I_{V_{i-1}}, d_c\right), d_v, \frac{E_b}{N_0}, R\right) \quad i = 2, 3, \ldots, D \tag{27}$$

with $I_{V_1} = I_V$.

While these derivations allow to model the convergence of each method, the following theorem shows that the threshold value remains the same for all methods.

Theorem 1. *Based on EXIT chart analysis, the threshold of a code decoded by BP is not improved by shuffled BP or replica shuffled BP.*

Proof. Let γ be the threshold in standard BP decoding. When $E_b/N_0 \leq \gamma$, the EXIT curves of variable and check nodes cross each other at some point, say A. If $I_E = I_{V,STD}\left(I_A, d_v, \frac{E_b}{N_0}, R\right)$, then $I_A = I_{U,STD}\left(I_E, d_c\right)$. In (20)−(22), $I_V = I_E$, $I_{U_i} \equiv I_A$ and $I'_{V_i} \equiv I_E$. So $I_{U,SHF}\left(I_E, d_c\right) = I_A$. Since I_{U_i} is constant, $I_{V,STD} = I_{V,SHF}$ at point A. Then $I_E = I_{V,SHF}\left(I_A, d_v, \frac{E_b}{N_0}, R\right)$. Therefore the EXIT curves of variable and check nodes in shuffled BP also cross each other at point A . The same result can be proved for replica shuffled BP. □

This theorem provides a formal proof of the observations made in [22]. Indeed it is expected that the threshold derived on a tree can not be changed by modifying the scheduling of the algorithm only.

In general the actual graph does not satisfy all the constraints of this ideal model, but the convergence behavior of the corresponding code can still be well approximated by the ideal model as shown next. Figure 6 compares the EXIT functions obtained from the simulation method of [13] and the proposed closed forms. Both methods assume the input LLRs have a Gaussian distribution. We observe that the EXIT functions of these two methods are almost the same, which validates the EXIT functions derived in this paper.

We also verified by EXIT chart that the non-synchronous scheduling converges slower than the synchronous one, as shown in Figure 3. Figure 7 depicts the EXIT charts of five decoding methods. We observe that replica shuffled BP with four subdecoders using the synchronous scheme converges much faster than the other methods. Figure 8 depicts EXIT curves superimposed to constant-BER curves [28, Chapter 9]. For the same BER, the iteration number of standard BP is twice that of shuffled BP and 8 times that of replica shuffled BP with four subdecoders and synchronous updating.

Figure 9 depicts the EXIT curves of different decoding methods at the SNR 1.11 dB, which is the threshold of the $(3, 6)$ regular LDPC code. We observe that the EXIT curves of variable and check nodes cross each other at the same point for all the methods. Hence they have the same threshold as expected from Theorem 1.

These results can be readily extended to irregular LDPC codes.

2.3.3 EXIT chart of group plain shuffled BP

Based on the analysis of plain shuffled BP, we deduce the following theorem.

Theorem 2. *When decoding a regular LDPC code, group plain shuffled BP should have at least d_c groups in order to have at any given iteration the same performance as plain shuffled BP based on the ideal model.*

Simulation results presented in the next section confirm that this value is a good estimate of the least number of groups necessary to achieve the same performance as plain shuffled BP on real Tanner graphs. Consequently Theorem 2 indicates that the speed-up obtained by shuffled BP over standard BP can still be achieved with a high level of parallelism since in general d_c is quite small. For completeness, we develop the remaining case next.

When the group number is less than d_c, the EXIT function of group plain shuffled BP is easily obtained if the check node degree is divisible by the group number, but it becomes cumbersome otherwise. Let G be the number of groups. Suppose the check node degree d_c is divisible by G with $S_G = d_c/G$. Then the EXIT function of group plain shuffled BP can be described as

$$I_{U,SHF,GR_G}\left(I_V, d_c\right) = \frac{1}{d_c}\sum_{i=1}^{d_c} I_{U_i}. \tag{28}$$

If $i \bmod S_G = 1$, then

$$I_{U_i} = I_{U,STD}\left(\frac{(d_c - i)I_V + \sum_{k=1}^{i-1} I'_{V_k}}{d_c - 1}, d_c\right) \tag{29}$$

$$I'_{V_i} = I_{V,STD}\left(I_{U_i}, d_v, \frac{E_b}{N_0}, R\right). \tag{30}$$

Otherwise,

$$I_{U_i} = I_{U_m} \tag{31}$$

$$I'_{V_i} = I'_{V_m} \tag{32}$$

where $m = \lfloor (i-1)/S_G \rfloor \cdot S_G + 1$.

2.3.4 EXIT chart of group replica shuffled BP

The EXIT function of group replica shuffled BP with non-synchronous updating is almost the same as that of replica shuffled BP (i.e. $G = N$) except that I_{U_i}'s in (24) and (25) are obtained from (29) and (31).

For the synchronous scheme, when $G \leq D$, group replica shuffled BP can be regarded as applying standard BP G times. Therefore the corresponding EXIT function is

$$I_{U,REP_D,S,GR_G}\left(I_V, d_c\right) = I_{U,STD}\left(I_{V_G}, d_c\right) \tag{33}$$

$$I_{V_i} = I_{V,STD}\left(I_{U,STD}\left(I_{V_{i-1}}, d_c\right), d_v, \frac{E_b}{N_0}, R\right) \quad i = 2, 3, \ldots, G \tag{34}$$

where $I_{V_1} = I_V$.

When $D \cdot d_c > G > D$, if G is divisible by D and d_c is divisible by $\frac{G}{D}$, group replica shuffled BP is equivalent to applying group shuffled BP with $\frac{G}{D}$ groups D times. Let $T = \frac{G}{D}$. Then the EXIT function becomes

$$I_{U,REP_D,S,GR_G}\left(I_V, d_c\right) = I_{U,SHF,GR_T}\left(I_{V_D}, d_c\right) \tag{35}$$

$$I_{V_i} = I_{V,SHF,GR_T}\left(I_{U,SHF,GR_T}\left(I_{V_{i-1}}, d_c\right), d_v, \frac{E_b}{N_0}, R\right) \quad i = 2, 3, \ldots, D \tag{36}$$

where $I_{V_1} = I_V$.

When $G \geq D \cdot d_c$, the EXIT function of group replica shuffled BP with synchronous updating is the same as for $G = N$. Hence we have the following theorem.

Theorem 3. *When decoding a regular LDPC code, group replica shuffled BP should have at least $D \cdot d_c$ groups in order to have at any given iteration the same performance as replica shuffled BP based on the ideal model.*

Figure 10 depicts the EXIT curves obtained from the simulation method of [13] and the proposed closed forms for group shuffled BP and group replica shuffled BP with synchronous updating. We observe that the curves obtained with these two methods match each other well, which again validates our derived EXIT functions.

Figure 11 depicts the error performance of shuffled BP, group shuffled BP with 6 groups, replica and group replica shuffled BP with 24 groups with four subdecoders and synchronous updating for decoding a $(8000, 4000)$ $(3, 6)$ regular LDPC code, whose Tanner graph was constructed by the PEG method [21]. Since the number of the bit nodes, 8000, cannot be divided by 6 or 24, the remaining bit nodes are assigned to the corresponding last group. From this figure, we observe that the group methods with the smallest group number G derived theoretically in Theorem 2 and 3 have almost the same performance as their corresponding non-group counterparts.

2.4 Simulation results

Figure 12 depicts the word error rate (WER) of iterative decoding of a $(8000, 4000)(3, 6)$ LDPC code, with the standard BP, plain shuffled and group replica shuffled BP algorithms, for $G = 2$, 4, 8, 16 and 8000, with four replica subdecoders and synchronous updating. The maximum number of iterations I_{Max} for plain and group replica shuffled BP was set to 10. We observe that the WER performances of replica shuffled BP decoding with four subdecoders and $I_{Max} = 10$, and a group number larger or equal to four, are approximately the same as that of standard BP with $I_{Max} = 60$.

Figure 13 depicts the WER of standard and replica shuffled BP decoding of a $(16200, 7200)$ irregular LDPC code which was constructed in a semi-random manner [25]. The variable node and check node degree distributions are $\lambda(x) = 0.00006x + 0.57772x^2 + 0.3111x^3 + 0.11111x^8$ and $\rho(x) = 0.00006x^2 + 0.14917x^3 + 0.29851x^4 + 0.44777x^5 + 0.10449x^6$, respectively. The number of replica subdecoders was four and updating was synchronous. We observe that replica shuffled BP with $I_{Max} = 10$ and $G = 16$ provides a similar performance as that of standard BP with $I_{Max} = 70$.

3 Iterative decoding of turbo codes

A turbo code [3] encoder is formed by the concatenation of two (or more) convolutional encoders, and its decoder consists of two (or more) soft-in/soft-out convolutional decoders

which feed reliability information back and forth to each other. For simplicity, we consider a turbo code that consists of two rate-$1/n$ systematic convolutional codes with encoders in feedback form. Let $\mathbf{u} = (u_1, u_2, \ldots, u_K)$ be an information block of length K and $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K)$ be the corresponding coded sequence, where $\mathbf{c}_k = (c_{k,1}, c_{k,2}, \ldots, c_{k,n})$, for $k = 1, 2, \ldots, K$, is the output code block at time k. Suppose BPSK transmission over an AWGN channel, with u_k and $c_{k,j}$ all taking values in {+1, -1} for $k = 1, 2, \ldots, K$ and $j = 1, 2, \ldots, n$. Let $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K)$ be the received sequence, where $\mathbf{y}_k = (y_{k,1}, y_{k,2}, \ldots, y_{k,n})$ is the received block at time k. Let $\hat{\mathbf{u}} = \{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_K\}$ denote the estimate of $\mathbf{u}$. Let s_k denote the encoder state at time k. Following [4], define $\alpha_k(s) = p(s_k = s, \mathbf{y}_1^k)$, $\gamma_k(s', s) = p(s_k = s, y_k | s_{k-1} = s')$, $\beta_k(s) = p(\mathbf{y}_{k+1}^K | s_k = s)$, where $\mathbf{y}_a^b = (\mathbf{y}_a, \mathbf{y}_{a+1}, \ldots, \mathbf{y}_b)$, and let $\alpha_k^{(m)}(s)$, $\gamma_k^{(m)}(s', s)$, $\beta_k^{(m)}(s)$ represent the corresponding values computed by component decoder m, with $m = 1, 2$. Let $L_{em}^{(i)}(\hat{u}_k)$ denote the extrinsic value of the estimated information bit $\hat{u}_k$ delivered by component decoder m at the ith iteration [23].

3.1 Algorithms

3.1.1 Standard serial and parallel turbo decoding

The decoding approach proposed in [3] operates in serial mode, i.e., the component decoders take turns in generating the extrinsic values of the estimated information symbols, and each component decoder uses the most recent extrinsic messages delivered by the other component decoder as a priori values of the information symbols. The disadvantage of this scheme is its decoding delay. In the parallel turbo decoding algorithm [24], both component decoders operate in parallel at any given time. After each iteration, each component decoder delivers its extrinsic messages to the other decoder, which uses these messages as a priori values at the next iteration.

3.1.2 Plain shuffled turbo decoding

Although the parallel turbo decoding reduces the decoding delay of serial decoding by half, the extrinsic messages are not taken advantage of as soon as they become available, because the extrinsic messages are delivered to component decoders only after each iteration is completed. The aim of the shuffled turbo decoding is to use the more reliable extrinsic messages at each time. Let $\tilde{\mathbf{u}} = (\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_K)$ be the sequence permuted by the interleaver corresponding to the original information sequence $\mathbf{u} = (u_1, u_2, \ldots, u_K)$, according to the mapping $\tilde{u}_k = u_{\pi(k)}$, for $k = 1, 2, \ldots, K$. We assume that $k \neq \pi(k), \forall k$. There is a unique corresponding reverse mapping $u_k = \tilde{u}_{\pi^-(k)}$, for $k = 1, 2, \ldots, K$ and $k \neq \pi^-(k), \forall k$. In shuffled turbo decoding, first α's of the two component decoders are computed in parallel and then β's and γ's are calculated partially based on the most recent updates at the current iteration. Although the two component decoders operate simultaneously as in parallel turbo decoding scheme, the messages are updated during each iteration based on $\pi(k)$ and $\pi^-(k)$ [5]. Correspondingly it provides a faster decoding convergence.

3.1.3 Replica shuffled turbo decoding

In the plain shuffled turbo decoding summarized in Section 3.1.2, we assume all the component decoders compute α's followed by β's. Let us refer to the two component decoders as $\overrightarrow{D}_1$ and $\overrightarrow{D}_2$. Another possible scheme is to operate in the reverse order, i.e, all the component decoders compute β's followed by α's and we refer to them as $\overleftarrow{D}_1$ and $\overleftarrow{D}_2$. In terms of error performance, there is no difference between these two approaches. However, the reliabilities of the extrinsic messages associated with a certain information bit delivered by these two shuffled turbo decoders differ. In general, the more independent information is used, the more reliable the delivered messages become. For the extrinsic messages delivered by component decoder $\overrightarrow{D}_1$, which are denoted as $\overrightarrow{L}_{e1}^{(i)}(\hat{u}_k)$, the larger k is, the more reliable this message is. Similarly, for the extrinsic message $\overleftarrow{L}_{e1}^{(i)}(\hat{u}_k)$ delivered by $\overleftarrow{D}_1$, the smaller k is, the more reliable this message is. It is natural to expect a faster decoding convergence if these two shuffled turbo decoders operate cooperatively instead of independently. Because in this approach two sets of shuffled component decoders are used to decode the same sequence of information bits, we refer to it as replica shuffled turbo decoding. In replica shuffled turbo decoding, two plain shuffled turbo decoders (processing recursions in opposite directions) $\overrightarrow{D}_1$, $\overrightarrow{D}_2$ and $\overleftarrow{D}_1$, $\overleftarrow{D}_2$ operate simultaneously and exchange more reliable extrinsic messages during each iteration. We assume that the component decoders deliver extrinsic messages synchronously, i.e., $\overrightarrow{T}_k^1 = \overrightarrow{T}_k^2 = \overleftarrow{T}_k^1 = \overleftarrow{T}_k^2$, where the $\overrightarrow{T}_k^1$ ($\overleftarrow{T}_k^1$) and $\overrightarrow{T}_k^2$ ($\overleftarrow{T}_k^2$) denote the times at which $\overrightarrow{D}_1$ ($\overleftarrow{D}_1$) and $\overrightarrow{D}_2$ ($\overleftarrow{D}_2$) deliver the extrinsic values of the kth ($(K+1-k)$th) estimated symbol of the original information sequence $\mathbf{u}$ and of the interleaved sequence $\tilde{\mathbf{u}}$, respectively. As a result, each value is available as soon as computed or four new values become available at same time instant.

Let us first consider the processing of component decoder $\overrightarrow{D}_1$ at the ith iteration. After time $\overrightarrow{T}_{k-1}^1$, the values of $\overrightarrow{\alpha}_k^{(1)}(s)$ should be updated and the values of $\overrightarrow{\gamma}_k^{(1)}(s)$ are needed. There are two possible cases. The first case is $k > \pi^-(k)$, which means the extrinsic value $\overrightarrow{L}_{e2}^{(i)}(\hat{u}_k)$ of the information bit $\hat{u}_k$ has already been delivered by decoder $\overrightarrow{D}_2$. As in plain shuffled turbo decoding, this newly available $\overrightarrow{L}_{e2}^{(i)}(\hat{u}_k)$ is used to compute the values $\overrightarrow{\gamma}_k^{(1)}(s)$, $\overrightarrow{\alpha}_k^{(1)}(s)$, and $\overrightarrow{L}_{e1}^{(i)}(\hat{u}_k)$. The second case is $k < \pi^-(k)$, which implies the extrinsic value $\overrightarrow{L}_{e2}^{(i)}(\hat{u}_k)$ of the information bit $\hat{u}_k$ has not been delivered yet by $\overrightarrow{D}_2$. Then in plain shuffled turbo decoding, the values $\alpha_k^{(1)}(s)$ and $L_{e1}^{(i)}(\hat{u}_k)$ are updated based on the extrinsic messages delivered at last iteration. In replica shuffled turbo decoding, however, there are two further subcases. The first subcase is $K+1-k < \pi^-(k)$, which implies the extrinsic value $\overleftarrow{L}_{e2}^{(i)}(\hat{u}_k)$ of the information bit $\hat{u}_k$ has already been delivered by decoder $\overleftarrow{D}_2$. Then this newly available $\overleftarrow{L}_{e2}^{(i)}(\hat{u}_k)$, instead of $\overrightarrow{L}_{e2}^{(i-1)}(\hat{u}_k)$ is used to compute the values $\overrightarrow{\gamma}_k^{(1)}(s)$, $\overrightarrow{\alpha}_k^{(1)}(s)$, and $\overrightarrow{L}_{e1}^{(i)}(\hat{u}_k)$. The second subcase is $K+1-k < \pi^-(k)$, which implies both extrinsic messages of the information bit $\hat{u}_k$, i.e, $\overleftarrow{L}_{e2}^{(i)}(\hat{u}_k)$ and $\overrightarrow{L}_{e2}^{(i)}(\hat{u}_k)$ are not available yet. In this subcase, the values of $\overrightarrow{\alpha}_k^{(1)}(s)$ and $\overrightarrow{L}_{e1}^{(i)}(\hat{u}_k)$ are updated based on the extrinsic messages delivered at the $(i-1)$th iteration. The recursions of component decoders $\overrightarrow{D}_2$, $\overleftarrow{D}_1$ and $\overleftarrow{D}_2$ are realized based on the same principle. After I_{Max} iterations, the shuffled turbo decoding algorithm outputs $\hat{\mathbf{u}} = (\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_K)$, where $\hat{u}_k = \text{sgn}[(\overrightarrow{L}_{e1}^{(i)}(\hat{u}_k) + \overleftarrow{L}_{e1}^{(i)}(\hat{u}_k))/2 + (\overrightarrow{L}_{e2}^{(i)}(\hat{u}_k) + \overleftarrow{L}_{e2}^{(i)}(\hat{u}_k))/2 + \frac{4}{N_0} y_{k,1}]$, which is different from the estimate in standard turbo decoding [3] and plain shuffled turbo decoding.

Figure 14 (a) and (b) illustrate the decoding processes of plain and replica shuffled turbo decoding, respectively, with $K = 8$. In Figure 14 (a), when bit-1 of decoder D_1 is processed, the new extrinsic information from decoder D_2 is not available yet, and the extrinsic information from the previous iteration is used as a priori information; when bit-3 of decoder D_1 is processed, the new extrinsic information from the current iteration is used as it is already available. In Figure 14 (b), when bit-1 of decoder $\overrightarrow{D}_1$ is processed, no new extrinsic information from decoders $\overrightarrow{D}_2$ and $\overleftarrow{D}_2$ is available, so the information from the previous iteration is used; when bit-3 is processed, only the new extrinsic information from $\overrightarrow{D}_2$ is available, and this new value is used; when bit-7 is processed, information from decoder $\overrightarrow{D}_2$ is not available yet, but that from decoder $\overleftarrow{D}_2$ is; when bit-8 is processed, new extrinsic information from both $\overrightarrow{D}_2$ and $\overleftarrow{D}_2$ is available, and the most recently updated value is used. These two last cases illustrate the advantage of using replica decoders.

It is straightforward to generalize replica shuffled turbo decoding to multiple turbo codes which consist of more than two component codes. Also group of bits can be updated periodically only to reduce information exchanges between replicas. Based on the above descriptions with two replicas, the total computational complexity of the replica shuffled turbo decoding for multiple turbo codes at each decoding iteration is about twice that of the parallel turbo decoding.

The proposed approach can be generalized to more than two replicas of each decoder but in that case, termination issues have to be considered, unless the convolutional code is in tail-biting form. Furthermore, while complete forward and backward recursions have been considered, additional speed up seems achievable with the finite window implementation proposed in [26].

3.2 Analysis by EXIT chart

In this section, we first review the results obtained in [11, 13, 28]. Both channel observations and a priori knowledge can be modeled as conditional Gaussian random variables [11]. Denote L_o, L_a, and L_e the LLRs of channel observations, a priori and extrinsic messages, respectively. Since we assume an AWGN channel, each received signal $y = c + n$ with $n \sim \mathcal{N}(0, \sigma_n^2)$. Then $L_o = \ln \frac{p(y|c=+1)}{p(y|c=-1)} = \frac{2}{\sigma_n^2}(c + n)$. It follows

$$L_o|c \sim \mathcal{N}(\mu_o, \sigma_o^2) \tag{37}$$

where $\sigma_o^2 = 4/\sigma_n^2$ and $\mu_o = c\sigma_o^2/2$. Hence the consistency condition [27] is satisfied.

Consider the a priori input $A = \mu_A \cdot u + n_A$, with $\mu_A = \sigma_a^2/2$ and $n_A \sim \mathcal{N}(0, \sigma_a^2)$. Using a similar analysis, we obtain

$$L_a|u \sim \mathcal{N}(u\sigma_a^2/2, \sigma_a^2) \tag{38}$$

and the consistency condition is also satisfied. Denote I_a as the mutual information exchanged between L_a and u, and I_e as that exchanged between L_e and u. Since L_a is conditionally Gaussian and the consistency condition is satisfied, I_a is independent of the value of u. Therefore I_a can be written as a function of σ_a, say $J(\sigma_a)$ where [11, 28]

$$J(\sigma_a) = 1 - \int_{-\infty}^{\infty} \frac{\mathrm{e}^{-[(\xi - \sigma_a^2/2)^2/2\sigma_a^2]}}{\sqrt{2\pi}\sigma_a} \log_2(1 + \mathrm{e}^{-\xi}) d\xi. \tag{39}$$

Since we do not impose a Gaussian assumption on L_e, I_e is approximated based on the observation of N samples of L_e, so that [13, 28]

$$I_e \approx 1 - \frac{1}{N}\sum_{i=1}^{N} \log_2[1 + \mathrm{e}^{-u_i L_{ei}}]. \tag{40}$$

The transfer function is defined as $I_e = T(I_a, E_b/N_0)$ and for a fixed value E_b/N_0, it is just $I_e = T(I_a)$. The transfer functions of both decoders are plotted on a single chart. Since in turbo decoding the extrinsic messages of the first decoder serve as the a priori messages of the second decoder, the axes are swapped for the transfer function of decoder-2.

3.2.1 Analysis of plain shuffled turbo decoding

In [28, Chapter 9], a Monte Carlo model is used to derive the EXIT chart for a given turbo code. Its structure is shown in Figure 15, with two Gaussian random noise generator outputs L_o and L_a whose distributions satisfy (37) and (38), respectively. Then L_o and L_a are sent to the SISO decoder, which outputs L_e. Based on (39) and (40) I_a and I_e can be calculated. The transfer functions are obtained accordingly.

In plain shuffled turbo decoding, each decoder sends the newly updated extrinsic messages to the other decoder immediately after updating. Hence we adopt three Gaussian random noise generators in the model to compute the transfer function, as shown in Figure 16. The first two generators are identical to those in Figure 15, while the third one takes the interleaved sequence $\tilde{\mathbf{u}}$ as input. The outputs of all these generators, L_o, L_{a1} and L_{a2}, are sent to the plain shuffled turbo decoders, where L_{a1} and L_{a2} are used as the a priori messages of decoder-1 and decoder-2, respectively. Then L_{e1} and L_{e2} are obtained and both of them are used to calculate I_e in (40).

3.2.2 Analysis of replica shuffled turbo decoding

For replica shuffled turbo decoding, the model to compute the transfer function is depicted in Figure 17. Since the four decoders, $\overrightarrow{D}1$, $\overrightarrow{D}2$, $\overleftarrow{D}1$ and $\overleftarrow{D}2$, exchange information synchronously, the newly updated a priori messages of $\overrightarrow{D}1$ and $\overleftarrow{D}1$ are the same after each iteration and so are those of $\overrightarrow{D}2$ and $\overleftarrow{D}2$. Therefore we still use three Gaussian random noise generators, but send L_{a1} to $\overrightarrow{D}1$ and $\overleftarrow{D}1$, and L_{a2} to $\overrightarrow{D}2$ and $\overleftarrow{D}2$, respectively. Since each decoder takes the extrinsic messages from two other decoders as its a priori messages, only the most recently updated extrinsic messages serve as the a priori messages in the next iteration. Hence it is more convenient to use the a priori LLRs for the next iteration, say L'_{a1} and L'_{a2}, to calculate I_e. Therefore in Figure 17, we have the replica shuffled turbo decoder output L'_{a1} and L'_{a2} instead of $\overleftarrow{L}_{e1}$, $\overleftarrow{L}_{e2}$, $\overrightarrow{L}_{e1}$ and $\overrightarrow{L}_{e2}$. The values I_a and I_e are then calculated using the same formulas as before and the transfer functions follow.

3.3 Simulation results

Figure 18 depicts the EXIT charts of a rate-1/3 turbo code with two component codes and interleaver size 16384, for standard parallel, plain shuffled, and replica shuffled turbo

decoding at the SNR 0.15dB. We observe that the replica shuffled turbo decoding converges faster than both the parallel and plain shuffled turbo decoding.

Figure 19 depicts the BER of the same turbo code, with standard parallel, plain shuffled and replica shuffled decoding. After five iterations, the replica shuffled turbo decoder outperforms its parallel and plain counterparts by several tenths of a dB. Furthermore, at the SNR value 0.15dB, the BER of replica shuffled turbo decoding after five iterations is slightly worse than that of standard parallel turbo decoding after ten iterations, as predicted from the EXIT charts in Figure 18.

4 Conclusion

Replica shuffled iterative methods have been proposed to decode LDPC codes and turbo codes with reduced latency. The faster convergence property of the presented algorithms has been verified by density evolution and EXIT charts. Both theoretical analysis and simulation results show that replica shuffled decoding provides good tradeoffs with respect to performance, complexity and latency. Although not explored in this work, connectivity in the decoder realization can also benefit from the replica approach.

In general, the proposed replica approach can be viewed as several processing elements updating the same memory unit, each element corresponding to one iteration of the underlying algorithm. The global scheduling of the memory accesses can be determined from the convergence analysis by density evolution or EXIT charts. This analysis is also useful to design codes suitable for replica decoding.

References

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: M.I.T. Press, 1963.

[2] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399-431, Mar. 1999.

[3] C. Berrou and A. Glavieux, "Near-optimum error-correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, pp. 1261-1271, Oct. 1996.

[4] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284-287, Mar. 1974.

[5] J. Zhang and M. Fossorier, "Shuffled Belief Propagation Decoding," *IEEE Trans. Commun.*, vol. 53, pp. 209-213, Feb. 2005.

[6] H. Kfir and I. Kanter, "Parallel versus sequential updating for belief propagation decoding," *Physica A*, vol. 330, pp. 259-270, 2003.

[7] J. Zhang and M. Fossorier, "Shuffled Belief Propagation Decoding," *Proc. 36th Annual Asilomar Conf. on Signals, Systems and Computers*, CA, US, pp. 8-15, Nov. 2002.

[8] E. Sharon, S. Litsyn, and J. Goldberger, "An efficient message-passing schedule for LDPC decoding," *Electrical and Electronics Engineers in Israel, 2004. Proceedings*, pp. 223-226, Sept. 2004.

[9] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, and M. Jézéquel, "Designing good permutations for turbo codes: towards a single model," *Proc. 2004 IEEE Int. Conf. Commun.*, Paris, France, pp. 341-345, Jun. 2004.

[10] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599-618, Feb. 2001.

[11] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Inform. Theory*, vol. 49, pp. 1727-1737, Oct. 2001.

[12] M. Tüchler, S. ten Brink, and J. Hagenauer, "Measures for tracing convergence of iterative decoding algorithms," *Proc. 4th IEEE/ITG Conf. on Source and Channel Coding*, Berlin, Germany, pp.53-60, Jan. 2002.

[13] M. Tüchler and J. Hagenauer, "EXIT charts of irregular codes," *Proc. 2002 Conf. Information Sciences and Systems*, Princeton, NJ, pp. 748-753, Mar. 2002.

[14] F. Guilloud, *Generic architecture for LDPC codes decoding*, Ph.D. thesis, ENST Paris, France, 2004.

[15] E. Yeo, P. Pakzad, B. Nikolic and V. Anantharam, "High throughput low-density parity-check decoder architectures," *Proc. 2001 IEEE Global Telecommun. Conf.*, TX, US, pp. 3019-3024, Nov. 2001.

[16] M. M. Mansour and N. R. Shanbhag, "Turbo decoder architecture for low-density parity-check codes" *Proc. 2002 IEEE Global Telecommun. Conf.*, pp. 1383-1388, Nov. 2002.

[17] Y. Kou, S. Lin, and M. Fossorier, "Low density parity check codes based on finite geometries: a rediscovery and more," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711-2736, Nov. 2001.

[18] S. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of Low-Density Parity-Check codes within 0.0045dB of the Shannon Limit," *IEEE Commun. Lett.*, vol. 5, pp. 58-60, Feb. 2001.

[19] T. J. Richardson, M. A. Shokrollahi, and R.L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619-637, Feb. 2001.

[20] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. Commun.*, vol. 52, pp. 670-678, Apr. 2004.

[21] X. Hu, E. Eleftheriou, and D. Arnold, "Progressive edge-growth Tanner graphs," *Proc. 2001 IEEE Global Telecommun. Conf.*, TX, US, pp. 995-1001, Nov. 2001.

[22] S. Tong and X. Wang, "Convergence analysis of Gallager codes under different message-passing schedules," *IEEE Commun. Lett.*, vol. 9, pp. 249-251, Mar. 2005.

[23] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of block and convolutional codes," *IEEE Trans. on Inform. theory*, vol. 42, pp. 429-445, Mar. 1996.

[24] D. Divsalar and F. Pollara, "Multiple turbo codes for deep-space communications," *JPL TDA Progress Report*, pp. 66-77, May 1995.

[25] "Draft DVB-S2 Standard," available at *http://www.dvb.org*.

[26] A. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes ," *IEEE J. on Select. Areas in Commun.*, Vol. 12, pp. 260-264, Feb. 1998.

[27] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity-check codes," *Proc. 2000 IEEE Int. Symp. Inform. Theory*, Sorrento, Italy, p. 199, Jun. 2000.

[28] E. Biglieri, *Coding of Wireless Channels*, Springer-Verlag, preprint.

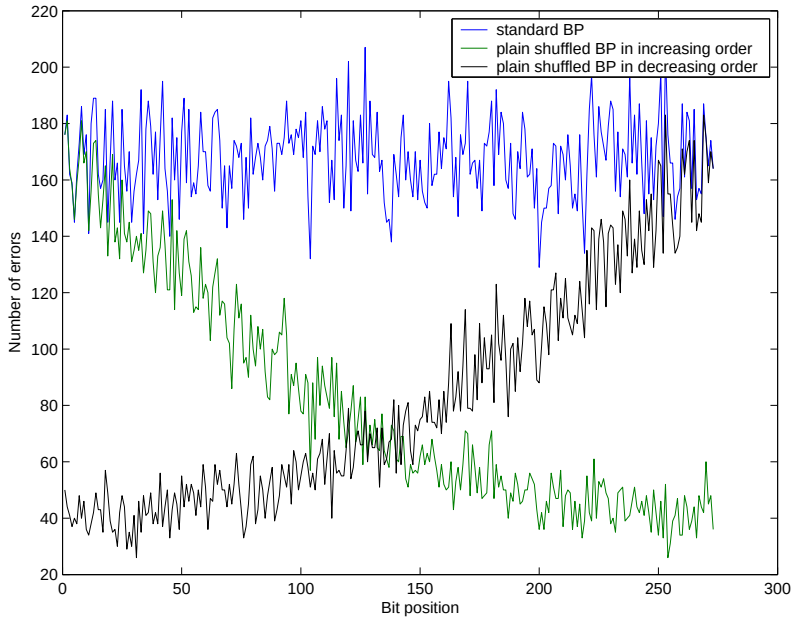


Figure 1: Number of bit errors versus bit position in the (273,191) PG-LDPC code at SNR 3.0 dB.

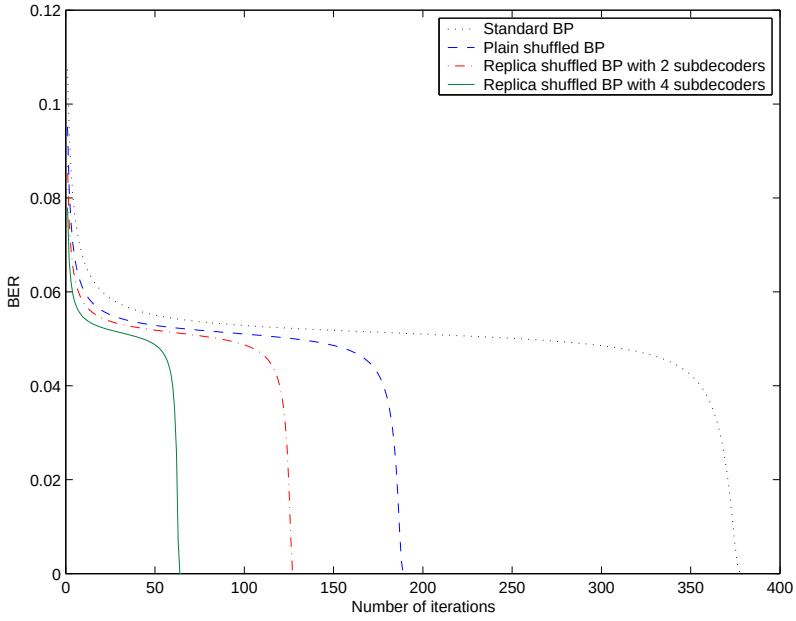


Figure 2: BER versus number of iterations predicted by density evolution with the standard BP, plain shuffled BP, replica shuffled BP with two and four subdecoders (synchronous scheme), for decoding a $(3, 6)$ regular LDPC code at the SNR 1.111 dB.

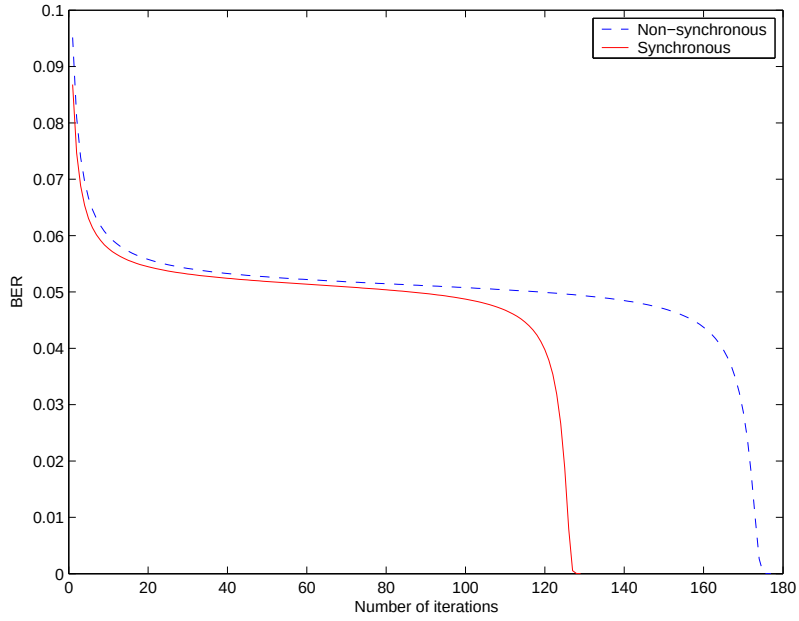


Figure 3: BER versus number of iterations predicted by density evolution with replica shuffled BP with two subdecoders under non-synchronous and synchronous updating schemes, for decoding a $(3, 6)$ regular LDPC code.

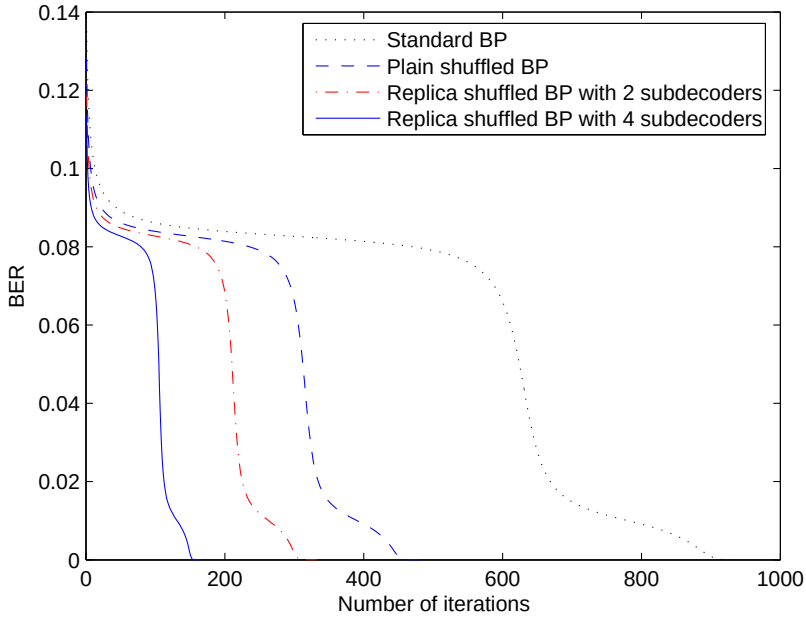


Figure 4: BER versus number of iterations predicted by density evolution with the standard BP, plain shuffled BP, replica shuffled BP with two and four subdecoders (synchronous scheme), for decoding an irregular LDPC code at the SNR 0.409 dB.

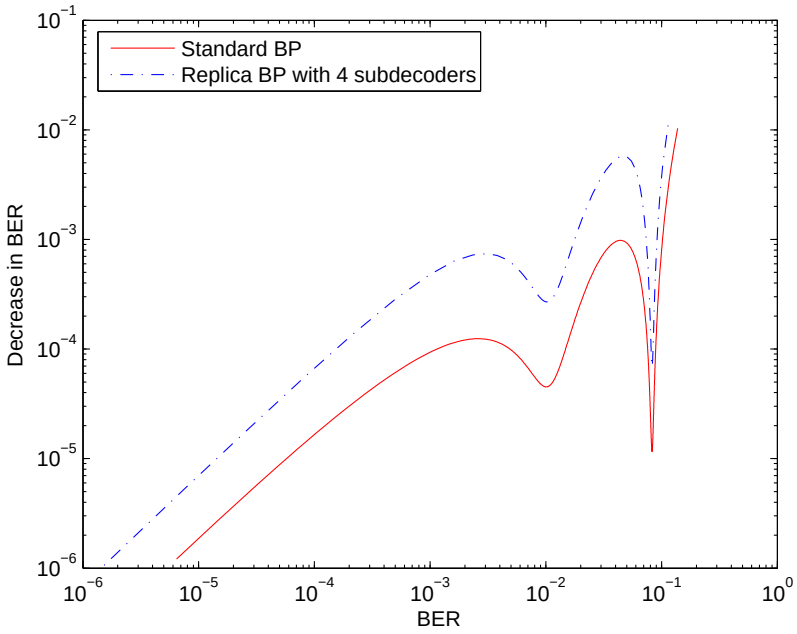


Figure 5: BER versus decrease in BER predicted by density evolution with the standard BP and replica shuffled BP with four subdecoders and synchronous updating, for decoding an irregular LDPC code at the SNR 0.409 dB.

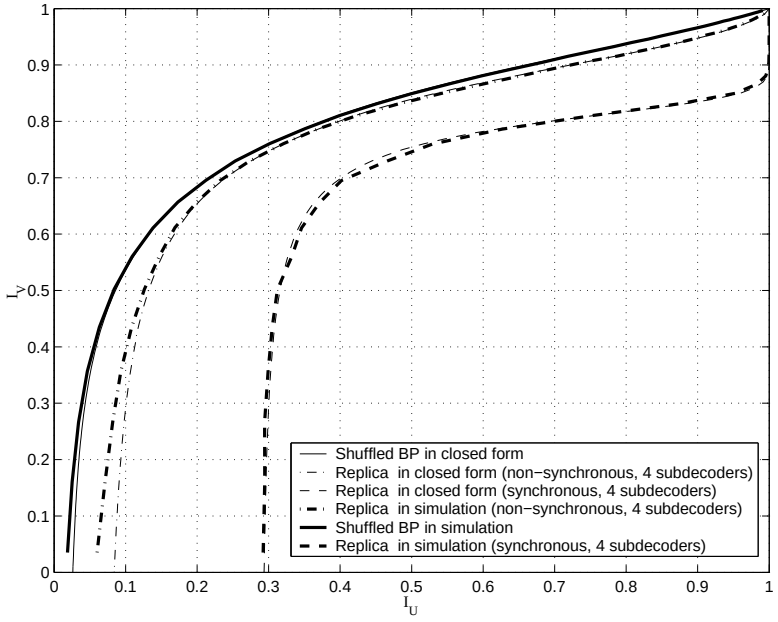


Figure 6: Comparison between the EXIT curves obtained from the simulation method of [13] and the proposed closed forms for a $(3, 6)$ regular LDPC code at the SNR 1.5 dB.

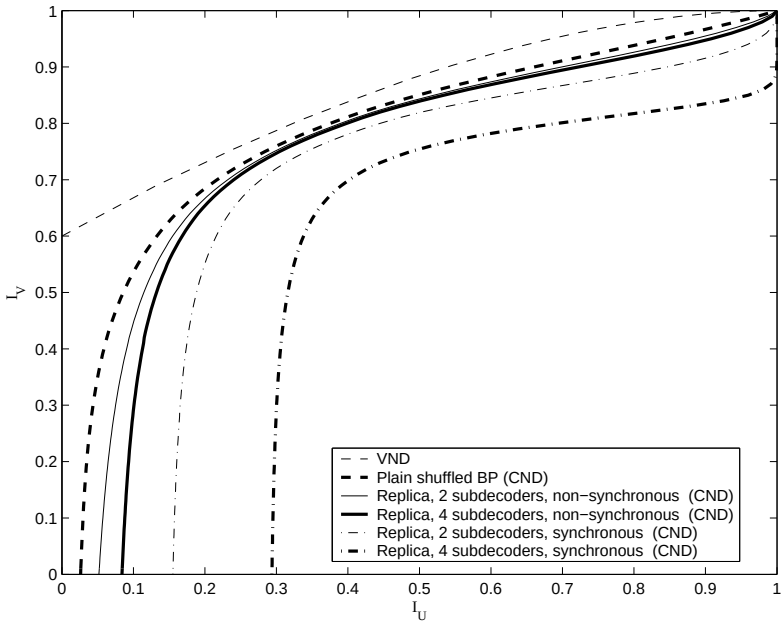


Figure 7: EXIT curves (in closed form) for shuffled BP and four types of replica shuffled BP decodings at the SNR 1.5 dB (variable nodes (VND) and check nodes (CND)).

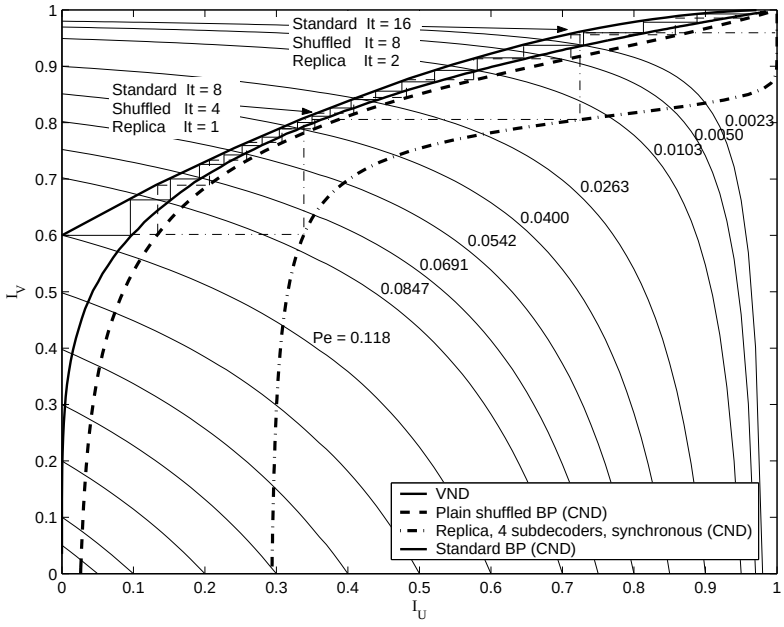


Figure 8: EXIT curves (in closed form) for standard BP, shuffled BP and replica shuffled BP with four subdecoders with synchronous updating at the SNR 1.5 dB, superimposed to constant-BER curves.

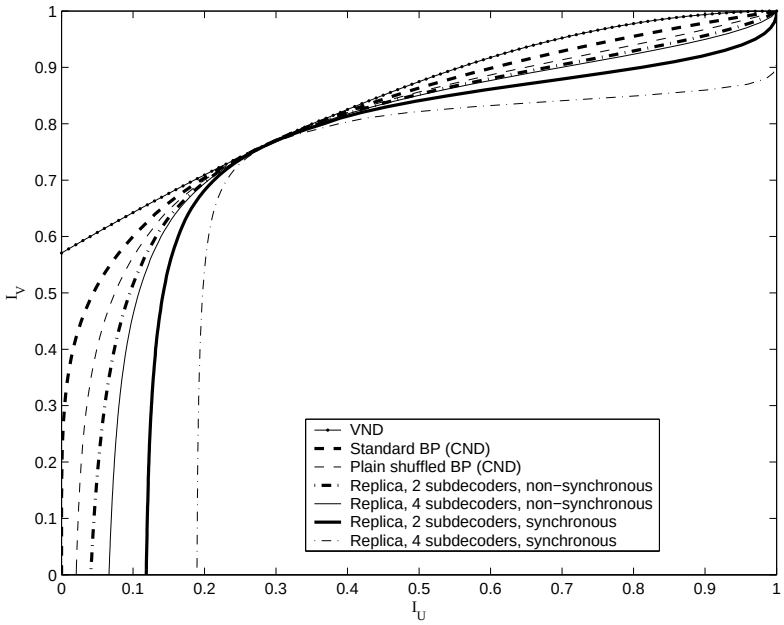


Figure 9: EXIT curves (in closed form) for standard BP, shuffled BP and four types of replica shuffled BP at the SNR 1.11 dB.

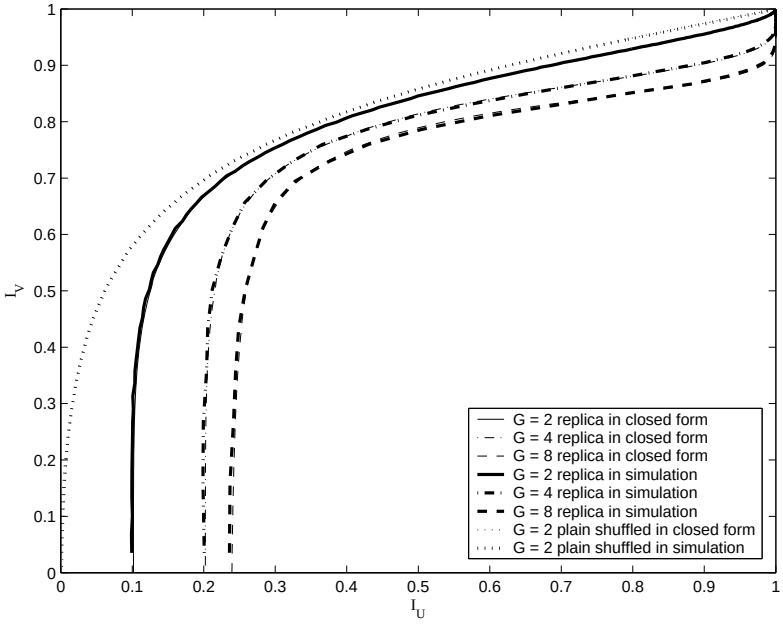


Figure 10: Comparison between the EXIT curves obtained from the simulation method of [13] and the proposed closed forms for group shuffled BP and group replica shuffled BP with four subdecoders and synchronous updating, for decoding a $(3, 6)$ regular LDPC code at the SNR 1.5 dB.

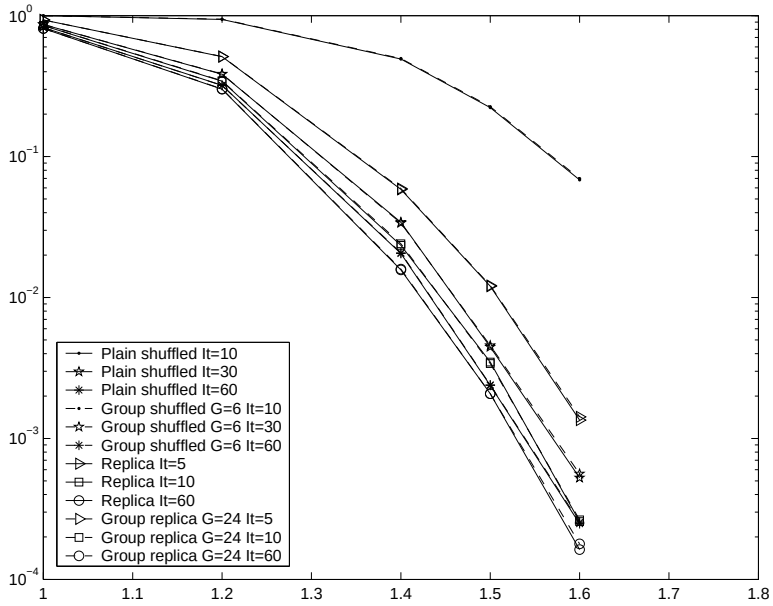


Figure 11: WER of shuffled BP, group shuffled BP with 6 groups, replica shuffled BP with four subdecoders and synchronous updating and its group version with 24 groups, for decoding a $(8000, 4000)$ $(3, 6)$ regular LDPC code.

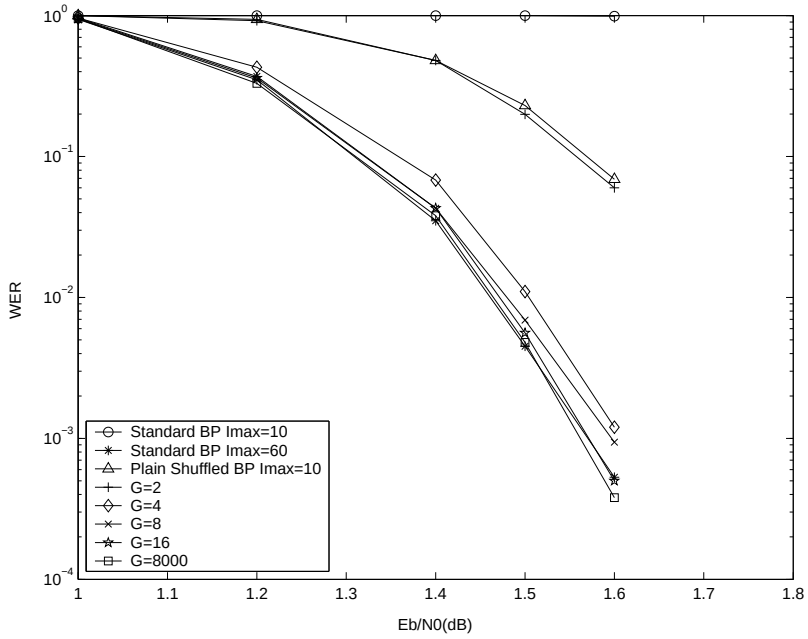


Figure 12: WER of a $(8000, 4000)(3, 6)$ LDPC code with group shuffled BP algorithm, for $G = 2, 4, 8, 16, 8000$ and at most 10 iterations.

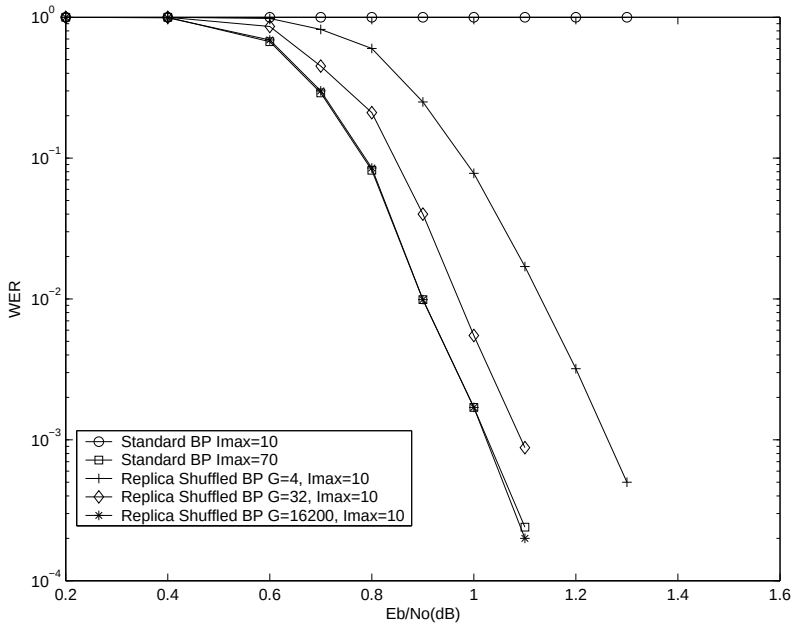
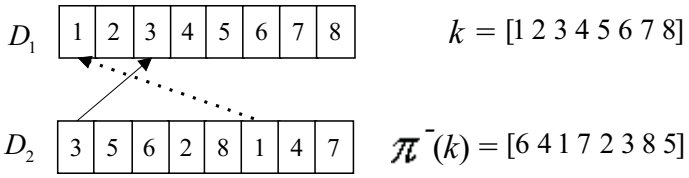
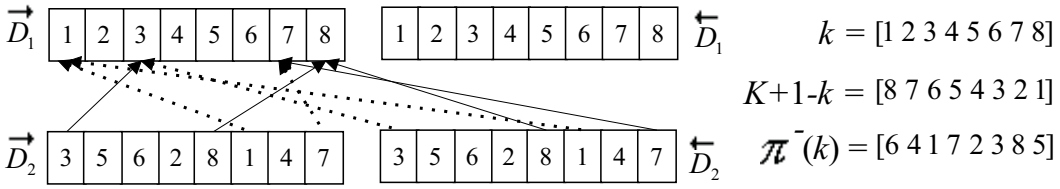


Figure 13: Error performance for iterative decoding of a $(16200, 7200)$ irregular LDPC code.



(a) Example of plain shuffled turbo decoding with K=8.



(b) Example of replica shuffled turbo decoding with $K = 8$.

Figure 14: Examples for illustrating the processing of plain and replica shuffled turbo decodings.

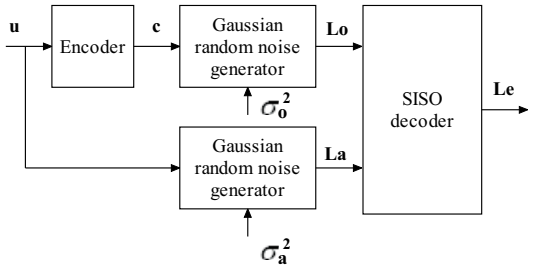


Figure 15: Monte Carlo model for computing the transfer function of a given turbo code with conventional turbo decoding.

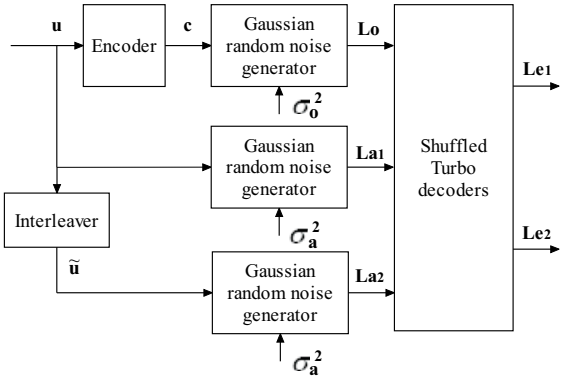


Figure 16: Monte Carlo model for computing the transfer function of plain shuffled turbo decoding.

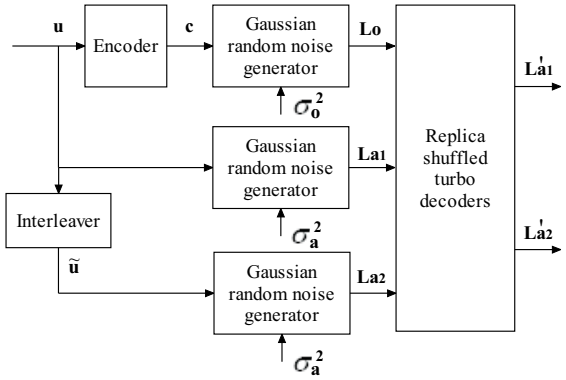


Figure 17: Monte Carlo model for computing the transfer function of replica shuffled turbo decoding.

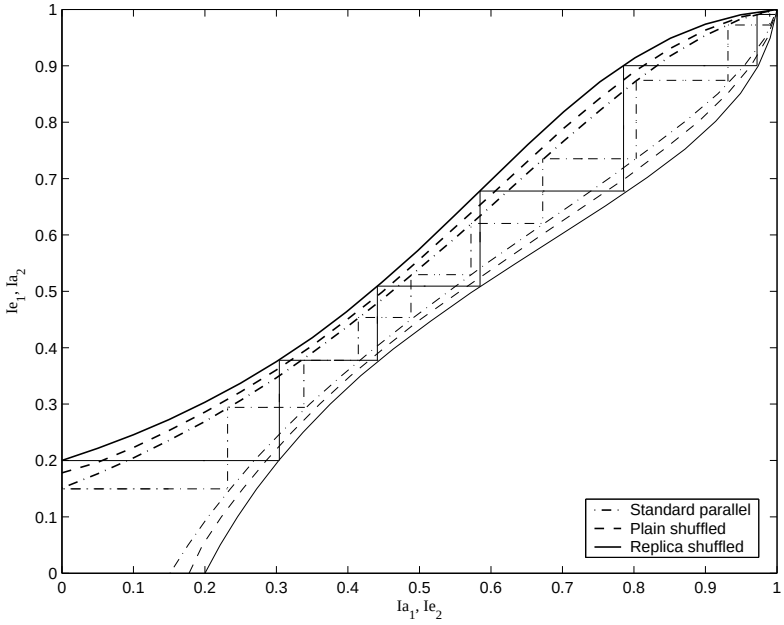


Figure 18: EXIT charts of a 2-component turbo code with interleaver size 16384, for standard parallel, plain shuffled, and replica shuffled turbo decoding, E_b/N_0=0.15 dB.

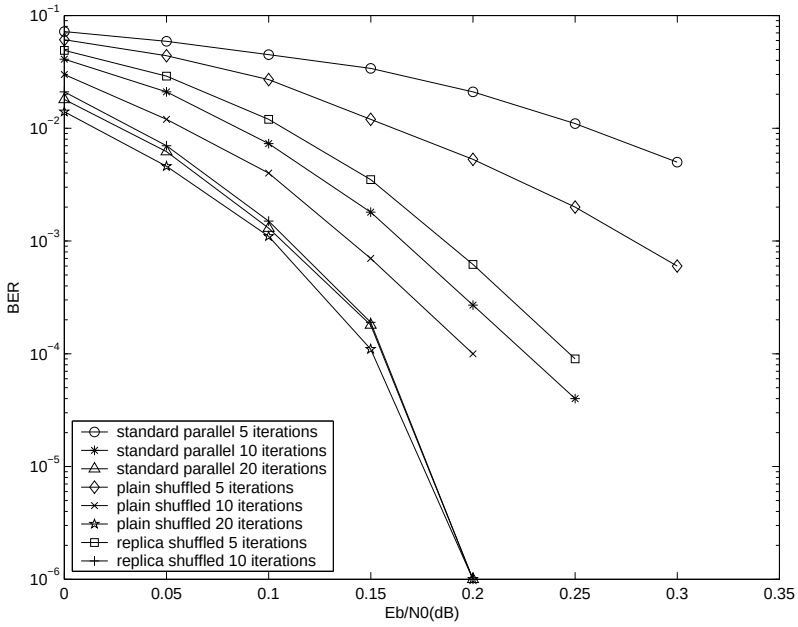


Figure 19: Bit error performance of a 2-component turbo code with interleaver size 16384, for standard parallel, plain shuffled and replica shuffled decodings.