

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

## **Replica Shuffled Belief Propagation Decoding on LDPC Codes**

Juntan Zhang, Yige Wang, Marc Fossorier, Jonathan Yedidia

TR2005-005 March 2005

### **Abstract**

Replica shuffled belief propagation decoders of low-density parity-check codes are presented. The proposed decoders converge faster than standard and shuffled belief propagation decoders. Simulations show that the new decoders offer good performance versus complexity trade-offs.

*Johns Hopkins University Conference on Information Sciences and Systems (CISS)*

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Copyright © Mitsubishi Electric Research Laboratories, Inc., 2005  
201 Broadway, Cambridge, Massachusetts 02139



Presented at the Conference on Information Sciences and Systems (CISS-2005) held at Johns Hopkins University in March, 2005.

# Replica Shuffled Belief Propagation

## Decoding of LDPC Codes

Juntan Zhang, Yige Wang and Marc Fossorier  
Department of Electrical Engineering  
University of Hawaii at Manoa  
Honolulu, HI 96822

Jonathan S. Yedidia  
Mitsubishi Electric Research Laboratories  
201 Broadway, Cambridge, MA 02139

### **Abstract**

Replica shuffled belief propagation decoders of low-density parity-check codes are presented. The proposed decoders converge faster than standard and shuffled belief propagation decoders. Simulations show that the new decoders offer good performance versus complexity trade-offs.

# 1 Introduction

Low-density parity-check (LDPC) codes [1] were first introduced in the 1960's and rediscovered in the 1990's. The iterative belief propagation (BP) algorithm [2] provides a powerful tool for decoding of LDPC codes [3]. LDPC codes with iterative decoding based on BP achieve remarkable performance that is very close to the Shannon limit.

To decode LDPC codes, either soft decision decoding or hard decision decoding can be used. Soft decoding based on BP gives better performance. However, the standard BP decoder of LDPC codes often needs several tens or hundreds of iterations for the iterative decoding process to converge, which is not always realistic in practical scenarios because of high decoding delay. Furthermore, LDPC codes of interest can have large code length in order to achieve a good performance, and it can be difficult to implement the decoding in hardware in a fully parallel way. In [4], a "shuffled" BP algorithm was presented which reduced the required number of iterations compared to standard BP by judicious scheduling which balanced parallel and serial operations. The aim of this paper is to propose a "replica shuffled" BP scheme which further accelerates the decoding process.

This paper is organized as follows. Section 2 and Section 3 briefly review the standard and shuffled BP decoding of LDPC codes, respectively. Section 4 presents the replica shuffled BP decoding method. Section 5 discusses parallel implementation of replica shuffled BP. Section 6 provides simulation results for the various iterative decoding algorithms, and Section 7 presents our conclusions.

## 2 Standard BP

An  $(N, k)$  LDPC code of length  $N$  and dimension  $k$  is specified by an  $M$  by  $N$  parity-check matrix containing mostly zeros and only a small number of ones. The parity check matrix  $\mathbf{H} = [H_{mn}]$  for an LDPC code has rows corresponding to the parity constraints and columns corresponding to the  $N$  bits of the code. As is well known, an LDPC code can equivalently be represented by a bipartite graph with check nodes corresponding to the rows of the parity check matrix and variable nodes corresponding to the columns.

We will 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 use  $\mathcal{N}(m) \setminus n$  to denote the set  $\mathcal{N}(m)$  with bit  $n$  excluded, and  $\mathcal{M}(n) \setminus m$  to denote the set  $\mathcal{M}(n)$  with check  $m$  excluded.

Suppose a regular binary  $(N, k)$  LDPC code  $\mathbf{C}$  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  $\mathbf{c} = (c_1, c_2, \dots, c_N)$  into a transmitted sequence  $\mathbf{x} = (x_1, x_2, \dots, x_N)$ , according to  $x_n = 1 - 2c_n$ , for  $n = 1, 2, \dots, N$ . 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  $y_n = x_n + \eta_n$ , where for  $1 \leq n \leq N$ , the  $\eta_n$  are statistically independent Gaussian random variables with zero mean and variance  $N_0/2$ .

In the following, we consider log-likelihood ratios (LLR's). An LLR passing along the edge connecting bit node  $n$  and check node  $m$  provides information about the decision of bit  $n$  and the reliability of that decision, according to all the information propagated to bit node  $n$  or check node  $m$ . We define the following notations associated with the  $i$ th iteration:

- $F_n$ : The LLR of bit  $n$  which is derived from the received value  $y_n$ . In BP decoding, we initially set  $F_n = \frac{4}{N_0}y_n$ .
- $\varepsilon_{mn}^{(i)}$ : The LLR of bit  $n$  which is sent from check node  $m$  to bit node  $n$ . It is obtained from the information  $\{z_{mn'}^{(i-1)} : n' \in \mathcal{N}(m) \setminus n\}$ , where the notation  $z_{mn}$  is introduced next.
- $z_{mn}^{(i)}$ : The LLR of bit  $n$  which is sent from the bit node  $n$  to check node  $m$ . It is obtained from the *a priori* information  $F_n$  and the information  $\{\varepsilon_{m'n}^{(i)} : m' \in \mathcal{M}(n) \setminus m\}$ .
- $z_n^{(i)}$ : The *a posteriori* LLR of bit  $n$  computed at each iteration. It is obtained from the *a priori* information  $F_n$  and the information  $\{\varepsilon_{mn}^{(i)} : m \in \mathcal{M}(n)\}$ .

For each iteration, we have the *a priori* information  $\{F_n\}$  obtained from the received values  $\{y_n\}$ , and the extrinsic information  $\{z_{mn}\}$  delivered by the previous iteration. We need to update  $\{z_{mn}\}$  as the extrinsic information for the next iteration, and compute  $\{z_n\}$  for the decision of the current iteration. The standard LLR BP algorithm in terms of extrinsic information is carried out as follows:

**Initialization:** Set  $i = 1$ , maximum number of iteration to  $I_{Max}$ . For each  $m, n$ , set  $z_{mn}^{(0)} = F_n$ .

**Step 1: (i)** Horizontal Step, for  $0 \leq n \leq N - 1$  and each  $m \in \mathcal{M}(n)$ , process:

$$\tau_{mn}^{(i)} = \prod_{n' \in \mathcal{N}(m) \setminus n} \tanh(z_{mn'}^{(i-1)}/2) \quad (1)$$

$$\varepsilon_{mn}^{(i)} = \log \frac{1 + \tau_{mn}^{(i)}}{1 - \tau_{mn}^{(i)}} \quad (2)$$

(ii) Vertical Step, for  $0 \leq n \leq N - 1$  and each  $m \in \mathcal{M}(n)$ , process:

$$z_{mn}^{(i)} = F_n + \sum_{m' \in \mathcal{M}(n) \setminus m} \varepsilon_{m'n}^{(i)} \quad (3)$$

$$z_n^{(i)} = F_n + \sum_{m \in \mathcal{M}(n)} \varepsilon_{mn}^{(i)} \quad (4)$$

**Step 2:** Hard decision and stopping criterion test:

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

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

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

### 3 Plain shuffled BP

At the  $i$ th iteration of the standard BP algorithm, first all values of the check-to-bit messages are updated by using the values of the bit-to-check messages obtained at the  $(i - 1)$ th iteration, i.e., each  $\varepsilon_{mn}^{(i)}$  is updated by using  $\{z_{mn'}^{(i-1)} : n' \in \mathcal{N}(m) \setminus n\}$ . Then, all values of the bit-to-check messages are updated by using the values of the check-to-bit messages newly obtained at the  $i$ th iteration, i.e., each  $z_{mn}^{(i)}$  is updated from  $\{\varepsilon_{m'n}^{(i)} : m' \in \mathcal{M}(n) \setminus m\}$ .

In general, for both the check-to-bit messages and bit-to-check messages, the more independent pieces of information that are used to update the messages, the more reliable they become. The  $i$ th iteration of the standard two step implementation of the BP algorithm uses all values  $z_{mn'}^{(i-1)}$  computed at the previous iteration in (1). However certain values  $z_{mn'}^{(i)}$  could already be computed in (3) based on a partial computation of the values  $\varepsilon_{mn}^{(i)}$  obtained from (2), and then be used instead of  $z_{mn'}^{(i-1)}$  in (1) to compute the remaining values  $\varepsilon_{mn}^{(i)}$ . This suggests a shuffling of the horizontal and vertical steps of the standard BP decoding. Hence this new version was referred to as “shuffled” BP decoding [4].

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:

$$\tau_{mn}^{(i)} = \prod_{\substack{n' \in \mathcal{N}(m) \setminus n \\ n' < n}} \tanh(z_{mn'}^{(i)}/2) \prod_{\substack{n' \in \mathcal{N}(m) \setminus n \\ n' > n}} \tanh(z_{mn'}^{(i-1)}/2) \quad (5)$$

## 4 Replica shuffled BP

Shuffled BP decoding is a bit-based sequential approach and the method described in Section 3 is based on a natural increasing order, i.e, the messages concerning bit nodes are updated according to order  $n = 1, 2, \dots, N$ . The larger the value of  $n$ , the more independent pieces of information are used to update the messages concerning bit  $n$  and the more reliable these messages become. Therefore, as the index  $n$  increases, the reliability of the bit decisions increases and the error rate decreases. Of course, the same reasoning applies if shuffled BP decoding is performed in reverse order; thus if shuffled BP decoding is employed using a bit order starting with bit  $N$  and ending with bit 1, then the error rate will increase with the index  $n$ . Figure 1 shows the number of bit errors using standard or shuffled BP decoding (with increasing and decreasing order) for the (273,191) PG-LDPC code [5] at a SNR of 3.0 dB after 10000 random blocks are decoded.

In replica shuffled BP decoding, two or more replica 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 subdecoders. Based on these more reliable messages, all replica subdecoders begin the next iteration. Let  $\overrightarrow{D}$  and  $\overleftarrow{D}$  denote the replica subdecoders with natural increasing and decreasing updating order, respectively. Let  $\overrightarrow{\varepsilon}_{mn}^{(i)}$  and  $\overrightarrow{z}_{mn}^i$  be variables associated with  $\overrightarrow{D}$  at iteration  $i$ . Variables associated with  $\overleftarrow{D}$  are defined in a similar way. The replica shuffled BP decoding with two replica subdecoders is carried out as followings:

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

**Step 1:** Each replica subdecoder process the following two steps simultaneously. For  $0 \leq n \leq N - 1$  and each  $m \in \mathcal{M}(n)$ , process

(i) Horizontal Step

$$\overrightarrow{\tau}_{mn}^{(i)} = \prod_{\substack{n' \in \mathcal{N}^{(m)} \setminus n \\ n' < n}} \tanh(\overrightarrow{z}_{mn'}^{(i)}/2) \bullet \prod_{\substack{n' \in \mathcal{N}^{(m)} \setminus n \\ n' > n}} \tanh(\overrightarrow{z}_{mn'}^{(i-1)}/2) \quad (6)$$

$$\overrightarrow{\varepsilon}_{mn}^{(i)} = \log \frac{1 + \overrightarrow{\tau}_{mn}^{(i)}}{1 - \overrightarrow{\tau}_{mn}^{(i)}} \quad (7)$$

$$\overleftarrow{\tau}_{mn}^{(i)} = \prod_{\substack{n' \in \mathcal{N}^{(m)} \setminus n \\ n' > n}} \tanh(\overleftarrow{z}_{mn'}^{(i)}/2) \bullet \prod_{\substack{n' \in \mathcal{N}^{(m)} \setminus n \\ n' < n}} \tanh(\overleftarrow{z}_{mn'}^{(i-1)}/2) \quad (8)$$

$$\overleftarrow{\varepsilon}_{mn}^{(i)} = \log \frac{1 + \overleftarrow{r}_{mn}^{(i)}}{1 - \overleftarrow{r}_{mn}^{(i)}} \quad (9)$$

(ii) Vertical Step

$$\overrightarrow{z}_{mn}^{(i)} = F_n + \sum_{m' \in \mathcal{M}(n) \setminus m} \overrightarrow{\varepsilon}_{m'n}^{(i)}$$

$$\overleftarrow{z}_{mn}^{(i)} = F_n + \sum_{m' \in \mathcal{M}(n) \setminus m} \overleftarrow{\varepsilon}_{m'n}^{(i)}$$

**Step 2** Exchange of more reliable messages. Set  $\overrightarrow{z}_{mn}^{(i)} = \overleftarrow{z}_{mn}^{(i)}$  for  $0 \leq n < N/2$  and  $\overleftarrow{z}_{mn}^{(i)} = \overrightarrow{z}_{mn}^{(i)}$  for  $N/2 \leq n \leq N - 1$ .

**Step 3:** Hard decision and stopping criterion test:

- (i) Create  $\hat{\mathbf{c}}^{(i)} = [\hat{c}_n^{(i)}]$  such that for  $0 \leq n < N/2$ ,  $\hat{c}_n^{(i)} = 1$  if  $F_n + \sum_{m \in \mathcal{M}(n)} \overleftarrow{\varepsilon}_{mn}^{(i)} < 0$ , and  $\hat{c}_n^{(i)} = 0$  otherwise; for  $N/2 < n \leq N - 1$ ,  $\hat{c}_n^{(i)} = 1$  if  $F_n + \sum_{m \in \mathcal{M}(n)} \overrightarrow{\varepsilon}_{mn}^{(i)} < 0$ , and  $\hat{c}_n^{(i)} = 0$  otherwise.
- (ii) If  $\mathbf{H}\hat{\mathbf{c}}^{(i)} = \mathbf{0}$  or the maximum iteration number  $I_{Max}$  is reached, stop the decoding iteration and go to Step 4. Otherwise set  $i := i + 1$  and go to Step 1.

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

Extension to more than two replica sub-decoders is straightforward, after noticing that the two sub-decoders could perform equally well if they started  $N/2$  bits apart.

## 5 Group replica shuffled BP

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 desirable. However, this scheme is not attractive for hardware implementation due to large decoding delay. A fully parallel implementation is not realistic either for large code lengths.

In [4], 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 make group replica shuffled BP decoders.

## 6 Simulation results

Figure 2 depicts the word error rate (WER) of iterative decoding of a randomly generated regular (8000, 4000) LDPC code with row weight 6 and column weight 3 [6], using standard BP, the ordinary (“plain”) shuffled and the group replica shuffled BP algorithm, for  $G = 2, 4, 8, 16$  and 8000 groups and with four replica subdecoders. The maximum number of iterations for plain and group replica shuffled BP was set to be  $I_{max} = 10$ . We observe that the WER performance of replica shuffled BP decoding with four subdecoders and  $I_{max} = 10$ , and  $G \geq 4$  is approximately the same as that of standard BP with  $I_{max} = 60$ .

We also simulated a (10000, 5000) irregular LDPC code constructed by the progressive edge-growth (PEG) algorithm of [7] with  $d_v = 15$ . The variable node degree distribution was  $\lambda(x) = 0.23802x + 0.20997x^2 + 0.03492x^3 + 0.12015x^4 + 0.01587x^6 + 0.00480x^{13} + 0.37627x^{14}$ , which was chosen from [8]. Four subdecoders were used for the replica shuffled BP algorithm. Figure 3 shows the bit error rate (BER) and word error rate (WER) performance of the standard BP, plain shuffled BP, and replica shuffled BP decoding algorithms. Replica shuffled BP with 10 iterations has almost the same performance as the standard BP with 60 iterations. Figure 4 compares the group and the original algorithms for  $G = 1000$ . As shown in Figure 4, when  $G$  is relatively large, the performance of the group algorithm approximates that of the original one.

Figure 5 depicts the WER of standard and replica shuffled BP decoding of a (16200, 7200) irregular LDPC code which is constructed in a semi-random matter and has been selected in the DVB-S2 standard [9]. The variable node degree distribution is  $\lambda(x) = 0.00006x + 0.57772x^2 + 0.31111x^3 + 0.11111x^8$ . The number of replica subdecoders is four. We observe that the replica shuffled BP with  $I_{max} = 10$  and  $G = 32$  provides a similar performance as that of the standard BP with  $I_{max} = 70$ .

## 7 Conclusion

In this paper, a low latency iterative decoding method for LDPC codes was proposed. A reduction of the number of iterations is achieved by duplicate decoding of the same iteration in parallel and combining the outputs after each iteration. As an example of the kind of speed-up possible using this method, we have shown that for the LDPC code selected in the DVB-S2 standard, 10 iterations of the proposed approach with four sub-decoders achieves the same performance as 70 iterations of the standard BP decoder. Extensions of this approach to other iteratively decodable codes have been considered in [10].

## References

- [1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
- [2] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.
- [3] D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Trans. Inform. Theory*, vol. 45, pp. 399-431, Mar. 1999.
- [4] J. Zhang and M. Fossorier, “Shuffled Belief Propagation Decoding,” *Proceedings of 36th Annual Asilomar Conference on Signals, Systems and Computers*, pp. 8-15, Nov. 2002.
- [5] 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.
- [6] D. J. C. MacKay “Encyclopedia of Sparse Graph Codes,” available at <http://www.inference.phy.cam.ac.uk/mackay/codes/data.html>.
- [7] X. Hu, E. Eleftheriou, and D. Arnold, “Progressive edge-growth Tanner graphs,” *IEEE Global Telecommunications Conference 2001*, vol.2, pp. 995-1001, Nov. 2001.
- [8] T. J. Richardson, M. A. Shokrollahi, and R.L. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Trans. Inf Theory*, vol.47, pp. 619-637, Feb. 2001.
- [9] “Draft DVB-S2 Standard,” available at <http://www.dvb.org>.
- [10] J. Zhang, Y. Wang, M. Fossorier, and J.S. Yedidia, “Replica Shuffled Iterative Decoding,” submitted to the 2005 International Symposium on Information Theory.

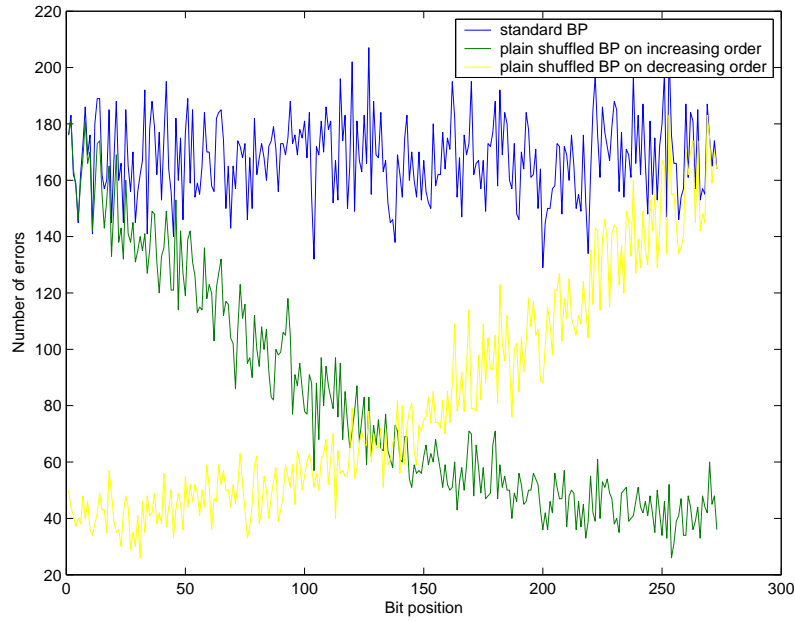


Figure 1: Number of bit errors in the (273,191) PG-LDPC code at a SNR of 3.0 dB.

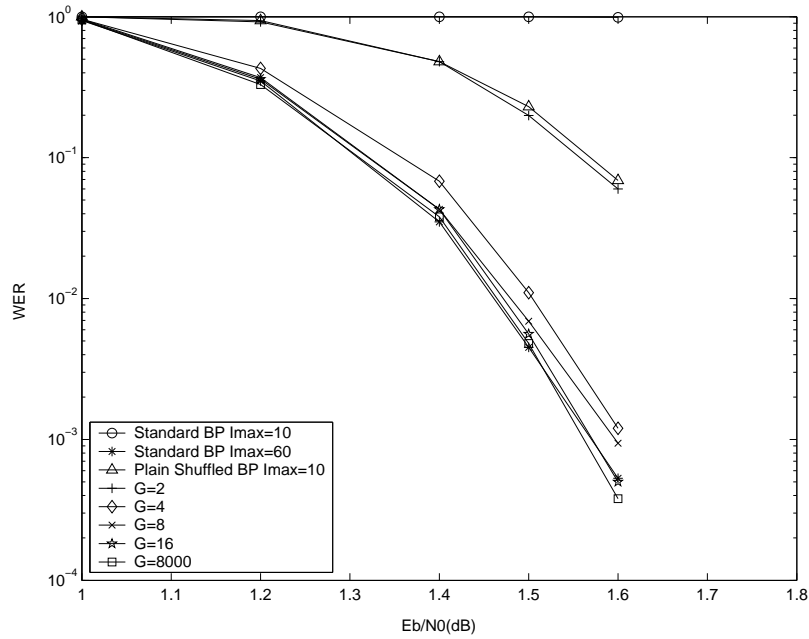


Figure 2: Error performance for iterative decoding of the (8000, 4000)(3, 6) LDPC code with the group replica shuffled BP algorithm, for  $G = 1, 2, 8, 100, 8000$  and at most 10 iterations.

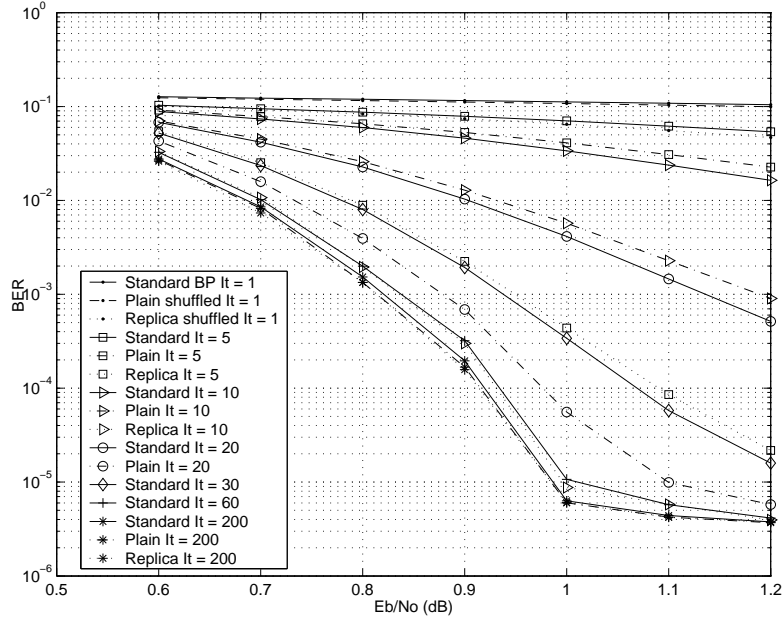


Figure 3: BER for iterative decoding of a (10000, 5000) irregular LDPC code with  $d_v = 15$  and at most 200 iterations.

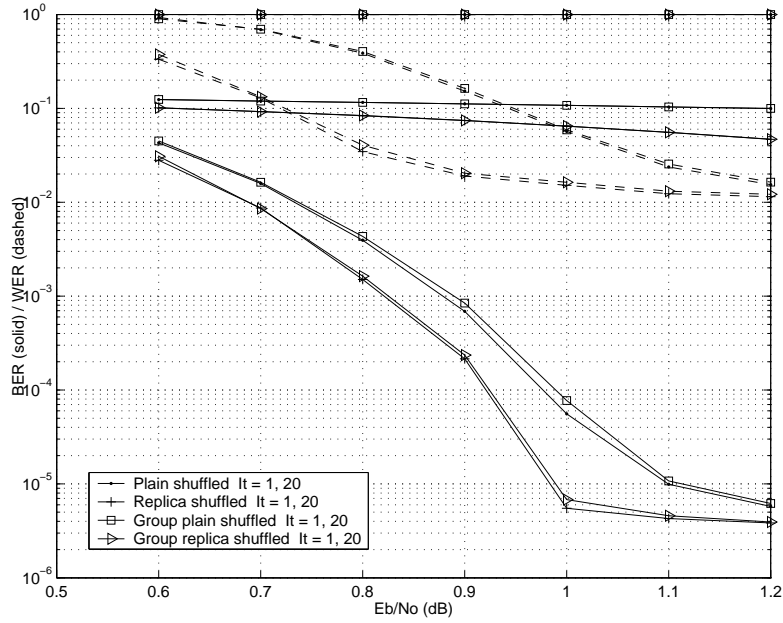


Figure 4: Error rates for iterative decoding of a (10000, 5000) irregular LDPC code with  $d_v = 15$ ,  $G = 1000$ , and at most 200 iterations.

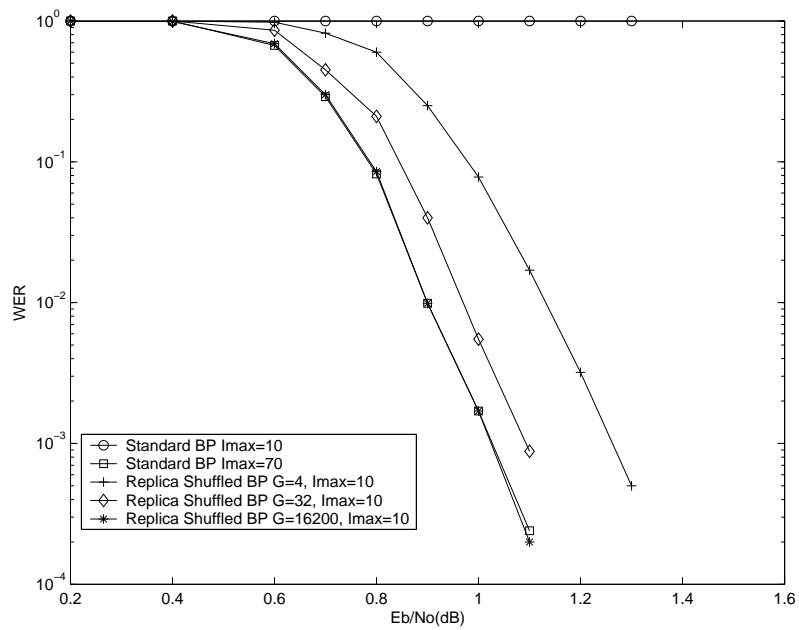


Figure 5: WER for iterative decoding of the (16200, 7200) irregular LDPC code selected in the DVB-S2 standard.