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

Iterative Decoding With Replicas

Juntan Zhang, Yige Wang, Marc P.C. Fossorier, and Jonathan S. Yedidia

TR2008-001 January 2008

Abstract

Replica shuffled versions of iterative decodes for low-density parity-check (LDPC)codes and turbo codes are presented. The proposed schemes can converge faster than standard and plain shuffled approaches. Two methods, density evolution and extrinsic information transfer (EXIT) charts, are used to analyze the performance of the proposed algorithms. Both theoretical analysis and simulations show that the new schedules offer good tradeoffs with respect to performance, complexity, latency, and connectivity.

IEEE Transactions on Information Theory, May 2007

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

Iterative Decoding With Replicas

Juntan Zhang, Yige Wang, Marc P.C. Fossorier, and Jonathan S. Yedidia

TR-2008-001 January 2008

Abstract

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

Published in IEEE Transactions on Information Theory, vol. 53, number 5, May 2007, pp. 1644-1663.

Iterative Decoding With Replicas

Juntan Zhang, Yige Wang, Marc P. C. Fossorier, *Fellow, IEEE*, and Jonathan S. Yedidia, *Member, IEEE*

***Abstract*—Replica shuffled versions of iterative decoders for low-density parity-check (LDPC) codes and turbo codes are presented. The proposed schemes can converge faster than standard and plain shuffled approaches. Two methods, density evolution and extrinsic information transfer (EXIT) charts, are used to analyze the performance of the proposed algorithms. Both theoretical analysis and simulations show that the new schedules offer good tradeoffs with respect to performance, complexity, latency, and connectivity.**

***Index Terms*—Belief propagation decoding, density evolution, extrinsic information transfer (EXIT) charts, low-density parity-check (LDPC) codes, turbo codes.**

I. 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. Standard iterative decoders of LDPC codes and turbo codes often require several tens of iterations for the iterative decoding process to converge. Hence, methods to accelerate the decoding convergence without sacrificing performance are needed.

A "shuffled" turbo decoding method was previously proposed [5] that takes account of the different reliabilities of extrinsic messages that are available during an iteration of a turbo decoder. The shuffled turbo decoding algorithm converges faster and only needs approximately the same computational complexity as standard parallel turbo decoding. Scheduling schemes using the "shuffled" idea have also been 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 replicated 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.

Our new approach is analyzed by density evolution [10] and extrinsic information transfer (EXIT) charts [11]–[13]. Both methods show that shuffled belief propagation (BP) converges about twice as fast as standard BP and replica shuffled BP converges faster than plain shuffled BP. The convergence speed of 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).

II. Iterative Decoding of LDPC Codes

LDPC codes 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 $(\lambda_1, \lambda_2, \ldots, \lambda_{d_v})$ and $(\rho_1, \rho_2, \ldots, \rho_{d_c})$, 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. For a regular (d_v, d_c) LDPC code, $\lambda_{d_v} = \rho_{d_c} = 1$.

A. 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 simplified versions of BP 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 replica principle can also be applied to a horizontal partitioning in a straightforward way and similar gains have been observed for both partitioning schedules.

Manuscript received June 23, 2005; revised November 10, 2006. This work was supported in part by the National Science Foundation under Grant CCF-04-30576.

J. Zhang was with the Department of Electrical Engineering, University of Hawai'i at Manoa, Honolulu, HI 96822 USA. He is now with Availink, Germantown, MD 20874 USA (e-mail: jzhang@availink.com).

Y. Wang and M. P. C. Fossorier are with the Department of Electrical Engineering, University of Hawai'i at Manoa, Honolulu, HI 96822 USA (e-mail: yige@spectra.eng.hawaii.edu; marc@spectra.eng.hawaii.edu).

J. S. Yedidia is with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139 USA (e-mail: yedidia@merl.com).

Communicated by Ø. Ytrehus, Associate Editor for Coding Techniques.

Color versions of Figures 1, 3–6, 8, and 11 in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2007.894683

1) Standard BP Decoding of LDPC Codes: Suppose a regular binary $(N,K)(d_v,d_c)$ LDPC code $\boldsymbol{C}$ of length N and dimension K is used for error control over an additive white Gaussian noise (AWGN) channel with zero mean and power spectral density $N_0/2$. Assume binary phase-shift keying (BPSK) signaling with unit energy, which maps a codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_N)$ into a transmitted sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, according to $x_n = 1 - 2c_n$, for $n = 1, 2, \ldots, N$. If $\boldsymbol{c} = [c_n]$ is a codeword in $\boldsymbol{C}$ and $\boldsymbol{x} = [x_n]$ is the corresponding transmitted sequence, then the received sequence is $\boldsymbol{x} + \boldsymbol{n} = \boldsymbol{y} = [y_n]$, with $y_n = x_n + n_n$, where for $1 \le n \le N$, the n_n's are statistically independent Gaussian random variables with zero mean and variance $N_0/2$. Let $\boldsymbol{H} = [H_{mn}]$ 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 using $\mathcal{N}(m)\backslash n$ the set $\mathcal{N}(m)$ with bit n excluded, and $\mathcal{M}(n)\backslash m$ the set $\mathcal{M}(n)$ with check m excluded. We define the following notations associated with the ith iteration.

- $U_{\mathrm{ch},n}$: The log-likelihood ratio (LLR) of bit n which is derived from the channel output y_n. In BP decoding, we initially set $U_{\mathrm{ch},n} = \frac{4}{N_0} y_n$.
- $U_{mn}^{(i)}$: 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 $V_{mn}^{(0)} = U_{\mathrm{ch},n}$.

Step 1:

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

$$\begin{aligned} V_{mn}^{(i)} &= U_{\mathrm{ch},n} + \sum_{m' \in \mathcal{M}(n)\backslash m} U_{m'n}^{(i)} \\ V_n^{(i)} &= U_{\mathrm{ch},n} + \sum_{m \in \mathcal{M}(n)} U_{mn}^{(i)}. \end{aligned} \tag{2}$$

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 $V_n^{(i)} \ge 0$.

(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) Plain Shuffled BP Decoding of LDPC Codes: In general, for both check-to-bit messages and bit-to-check messages, the more independent information that 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 decoding is referred to as shuffled BP decoding.

In the shuffled BP algorithm [5], 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 $1 \le n \le N$ and each $m \in \mathcal{M}(n)$, process the horizontal step and vertical step **jointly**, with (1) modified as

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

3) Replica Shuffled BP Decoding of LDPC Codes: Shuffled BP decoding is a bit-based sequential approach and the method described in Section II-A2 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 an illustration, Fig. 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 signal-to-noise ratio (SNR) of 3.0 dB and after the second iteration. A total of 10000 random blocks were decoded. From Fig. 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 messages 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

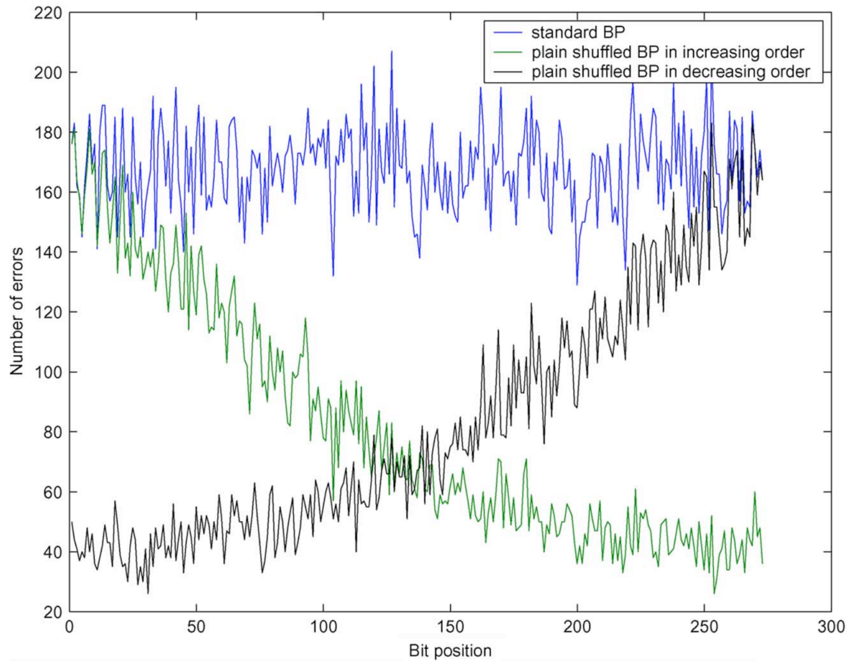

Fig. 1. Number of bit errors versus bit position in the $(273, 191)$ PG-LDPC code at SNR of 3.0 dB.

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 $\overrightarrow{V}_{mn}^{(0)} = \overleftarrow{V}_{mn}^{(0)} = U_{\text{ch},n}$.

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

(i) Horizontal Step

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

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

(ii) Vertical Step

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

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

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

Step 3: Hard decision and stopping criterion test:

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

(ii) If $\boldsymbol{H}\hat{\boldsymbol{c}}^{(i)} = \mathbf{0}$ 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 Fig. 1, note that Step 2 is equivalent to keeping 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 $R(n) = \{n' \mid n \le n' \le N-n\}$, and $\overline{R}(n) = \{n' \mid 1 \le n' \le N, n' \notin R(n)\}$, for $1 \le n \le N$. In synchronous scheme, the updating and exchanging procedures operate simultaneously as follows:

Step 1: For $1 \le n \le N$ 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

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

$$\times \prod_{\substack{n' \in \mathcal{N}(m)\backslash n \\ n' \in R(n)}} \tanh \frac{V_{mn'}^{(i-1)}}{2} \Bigg)$$

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

$$\times \prod_{\substack{p' \in \mathcal{N}(q)\backslash p \\ p' \in R(N-p)}} \tanh \frac{V_{qp'}^{(i-1)}}{2} \Bigg).$$

(ii) Vertical Step

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

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

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.

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 $N \mathrm{mod} G = 0$ for simplicity). Step 1 of the nonsynchronous group replica shuffled BP algorithm is carried out as follows:

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

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

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

$$\times \prod_{\substack{n' \in \mathcal{N}(m)\backslash n \\ n' \geq (g-1) \cdot B + 1}} \tanh \frac{\overrightarrow{V}_{mn'}^{(i-1)}}{2} \Bigg)$$

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

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

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

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

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

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

Replica shuffled BP can also update messages in groups based on unnatural increasing or decreasing orders. Suppose the updating order of one replica is $\boldsymbol{O} = \{O_1, O_2, \ldots, O_G\}$, where $O_g \in \{1, 2, \ldots, G\}$. Assume the updating orders of $\overrightarrow{D}$ and $\overleftarrow{D}$ are $\overrightarrow{\boldsymbol{O}}$ and $\overleftarrow{\boldsymbol{O}}$, respectively. Then replica shuffled BP with unnatural updating ordering can be described with the above updating rules by replacing $(g-1) \cdot B$ and $(G-g+1) \cdot B$ with $\overrightarrow{O}_g \cdot B$ and $\overleftarrow{O}_g \cdot B$, respectively.

Replica shuffled BP can be further generalized to various forms. One example is that in the unnatural updating scheme, some groups of bit nodes may be updated more than once at one iteration while other groups of bit nodes are updated only once. The updating of LLR values at the ith iteration is now based on the LLR values delivered at the $(i-1)$th or $(i-2)$th iteration (as in the example of Section II–A5).

5) Relationship Between Group Plain and Group Replica Shuffled BP: Group plain shuffled BP can be viewed as a special case of synchronous group replica shuffled BP. Assume in group plain shuffled BP decoding, the N bits in a codeword are divided into $G = \frac{N}{B}$ groups and each group contains B bits. Consider a group replica shuffled BP decoder with two subdecoders D_1 and D_2. For both D_1 and D_2, the N bits in a codeword are divided into $2G = \frac{N}{(B/2)}$ groups and each group contains $B/2$ bits. For $1 \leq g \leq G$, let bits in group-$(2g-1)$ in D_1 and bits in group-$2g$ in D_2 compose group-g in group shuffled BP decoding. In synchronous group replica shuffled BP decoding, if subdecoder D_1 updates group-$(2g-1)$ and subdecoder D_2 updates group-$2g$ simultaneously, group replica shuffled BP decoding with two subdecoders becomes group plain shuffled BP decoding.

Since each subdecoder in group replica shuffled BP decoder can take any updating order, group replica shuffled BP decoding provides more flexibility than group plain shuffled BP decoding. Hence, we can find some scheduling for group replica shuffled BP decoder that has better performance than group plain shuffled BP using the same decoding time and the same hardware resources, i.e., the same number of subdecoders. For example, consider 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

$$\rho(x) = 0.00006x^2 + 0.14917x^3 + 0.29851x^4 + 0.44777x^5 + 0.10449x^6$$

TABLE I
PERFORMANCE COMPARISON OF GROUP PLAIN AND GROUP REPLICA SHUFFLED BP DECODING

SNR (dB)	group plain shuffled BP (It=18)	group replica shuffled BP (It=6)
1.1	9.8e-4	8.6e-4
1.2	2.6e-4	2.0e-4
1.3	6.2e-5	4.0e-5
1.4	1.5e-5	1.0e-5
1.5	4.3e-6	1.8e-6

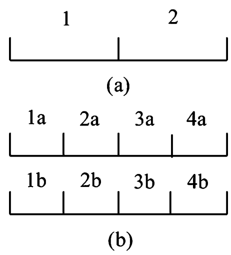

Fig. 2. Illustration of the scheduling of group plain shuffled BP decoding with two groups and group replica shuffled BP decoding with two subdecoders and four groups.

respectively. We compare group plain shuffled BP decoding with two groups and group replica shuffled BP decoding with two subdecoders and four groups. Fig. 2(a) illustrates the scheduling of group plain shuffled BP decoding. The variable nodes are divided into two parts, 1 and 2. The processing follows the order: $1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow \cdots$. Fig. 2(b) illustrates the scheduling of group replica shuffled BP decoding. With respect to the first subdecoder, the variable nodes are divided into four parts, $1a, 2a, 3a$, and $4a$. The processing follows the order: $1a \rightarrow 2a \rightarrow 1a \rightarrow 2a \rightarrow 3a \rightarrow 4a \rightarrow 1a \rightarrow 2a \rightarrow 1a \rightarrow 2a \rightarrow 3a \rightarrow 4a \cdots$. With respect to the second subdecoder, the variable nodes are divided into $1b, 2b, 3b$, and $4b$. The processing follows the order: $2b \rightarrow 1b \rightarrow 4b \rightarrow 3b \rightarrow 2b \rightarrow 1b \rightarrow 2b \rightarrow 1b \rightarrow 4b \rightarrow 3b \rightarrow 2b \rightarrow 1b \cdots$. Since the decoding time for one iteration in group replica shuffled BP triples that in group plain shuffled BP decoding, we compare their performance after 6 and 18 iterations, respectively, in Table I. We observe that with this particular scheduling and the same number of subdecoders, group replica shuffled BP outperforms group plain shuffled BP.

B. Analysis by Density Evolution

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 decoder, if the channel and the decoder satisfy the symmetry conditions [10], then the decoding bit error rate (BER) 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 (pdfs) of the outgoing messages.

In shuffled and replica shuffled BP decoding, the pdfs of outgoing and incoming messages of bit nodes depend on the bit index number. Let $f^{(i)}_{U_g}(u)$ and $f^{(i)}_{V_g}(v)$ be the pdfs of the incoming and outgoing messages of bit nodes in the gth group at iteration i, respectively. We assume the bits of an LDPC codeword are divided into G groups and for simplicity we assume given any check, the number of adjacent bits from any group is at most one.

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

$$f^{(i)}_{V_g} = \mathcal{F}^{-1}\left(\mathcal{F}\left(f_{U_{\mathrm{ch}}}\right) \cdot \left(\mathcal{F}\left(f^{(i)}_{U_g}\right)\right)^{dc-1}\right) \tag{4}$$

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

As observed from (3), $U^{(i)}_g$ depends on both $V^{(i)}_{g'}$ for $g' < g$ and $V^{(i-1)}_{g'}$ for $g' > g$. To avoid a brute-force calculation of all possible combinatorial formats of $V^{(i)}_{g'}$ and $V^{(i-1)}_{g'}$, we let the average pdf of the newly delivered outgoing messages from bit nodes in group $\{g'|g' < g\}$ at iteration i be

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

Similarly, we let the average pdf of the outgoing messages from bit nodes in group $g'|g' > g\}$ be

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

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 independent and identically distributed (i.i.d.) random variables with pdf $f_V(v)$, the pdf of the outgoing message can be efficiently computed as [18]

$$f_U = \Psi(\ldots\Psi(\Psi(f_V, f_V), f_V), \ldots, f_V) = \Psi^{d_c-1} f_V \tag{9}$$

where Ψ denotes the operation of pdfs of bit-to-check messages based on check node processing (7). Let us consider group shuffled BP with natural increasing ordering. The incoming messages to the check nodes adjacent to bit nodes in the gth group have in total

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

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

$$\binom{g-1}{j} \cdot \binom{G-g}{d_c-1-j} \tag{11}$$

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 of message incoming to bit nodes in the gth group at iteration i becomes

$$f_{U_g}^{(i)} = \sum_{j=0}^{d_c-1} \frac{\binom{g-1}{j} \cdot \binom{G-g}{d_c-1-j}}{\binom{G-1}{d_c-1}} \cdot \Psi\left(\Psi^j f_{\overline{V}_{g'<g}}^{(i)}, \Psi^{d_c-1-j} f_{\overline{V}_{g'>g}}^{(i-1)}\right). \tag{12}$$

Theorem 3.2.2 in [8] 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. For the bit nodes associated with group g, the updating rule in our paper is the same as that in [8]. However, the updating rules for check nodes differ. In [8], the updating of pdfs for check nodes is based on a single pdf, which is the average of all pdfs of the bit nodes. In our approach, the pdf of a check node is updated from all G different pdfs of bit nodes from G different groups, based on a combinatorial analysis with a consideration of the degree of the check node and all possible combinations of the pdfs of the bit nodes.

2) Density Evolution of Replica Shuffled BP: It is straightforward to extend these updating rules of pdfs for shuffled BP to replica shuffled BP. For instance, in nonsynchronous replica shuffled BP with two subdecoders, the updating rule of the pdfs of the outgoing belief messages from bit nodes is the same as that in plain shuffled BP, while the pdfs of incoming belief messages to bit nodes are modified as

$$f_{V_{G+1-g}}^{(i)} \leftarrow f_{V_g}^{(i)} \tag{13}$$

for $G/2 < g \leq G$. If an unnatural updating ordering is employed, the indices $G+1-g$ and g in (13) are replaced with $\overrightarrow{O}_{G+1-g}$ and $\overrightarrow{O}_g$, respectively.

Density evolution of synchronous replica shuffled BP operates in the same way while updating pdfs of incoming belief messages to bit nodes synchronously, i.e.,

$$f_{V_{G+1-g}}^{(i-1)} \leftarrow f_{V_g}^{(i)} \tag{14}$$

for $1 \leq g \leq G/2$, and

$$f_{V_{G+1-g}}^{(i)} \leftarrow f_{V_g}^{(i)} \tag{15}$$

for $G/2 < g \leq G$. 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 nodes in the gth group from a check node with degree l is

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

Since the pdfs of the outgoing messages of check nodes with different degree are distinct, the expectation of these pdfs is the overall pdf of the messages incoming to bit nodes int the gth group

$$f_{U_g}^{(i)} = \sum_{l=1}^{dc} \rho_l \sum_{j=0}^{l-1} \frac{\binom{g-1}{j} \cdot \binom{G-g}{l-1-j}}{\binom{G-1}{l-1}} \cdot \Psi\left(\Psi^j f_{\overline{V}_{g'<g}}^{(i)}, \Psi^{l-1-j} f_{\overline{V}_{g'>g}}^{(i-1)}\right).$$

Similarly, the pdf of outgoing messages from bit nodes in the gth group at iteration i becomes

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

3) Simulation Results: Fig. 3 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 rate-$1/2$ $(3,6)$ regular LDPC codes with $E_b/N_o = 1.111$ dB. In the simulation, we assume the bits in an LDPC codeword are divided into $G = 1000$ groups. 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.

Fig. 4 depicts the BER versus the number of iterations predicted by density evolution with replica shuffled BP decoder of two subdecoders using nonsynchronous 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 nonsynchronous exchanging schedule. It is also worth mentioning that the synchronous scheme requires less memory than the nonsynchronous scheme, but more frequent memory access.

Fig. 5 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

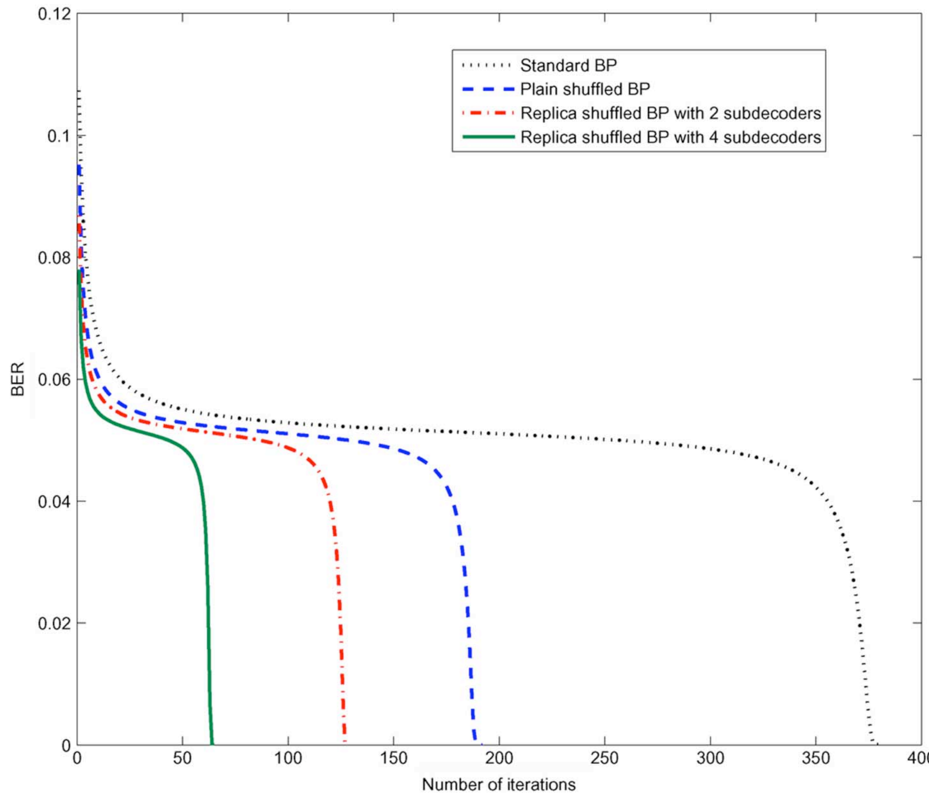

Fig. 3. BER versus number of iterations predicted by density evolution with the standard BP, plain shuffled BP, replica shuffled BP with two and four subdecoders (synchronous scheme), for decoding a $(3, 6)$ regular LDPC code at the SNR of 1.111 dB.

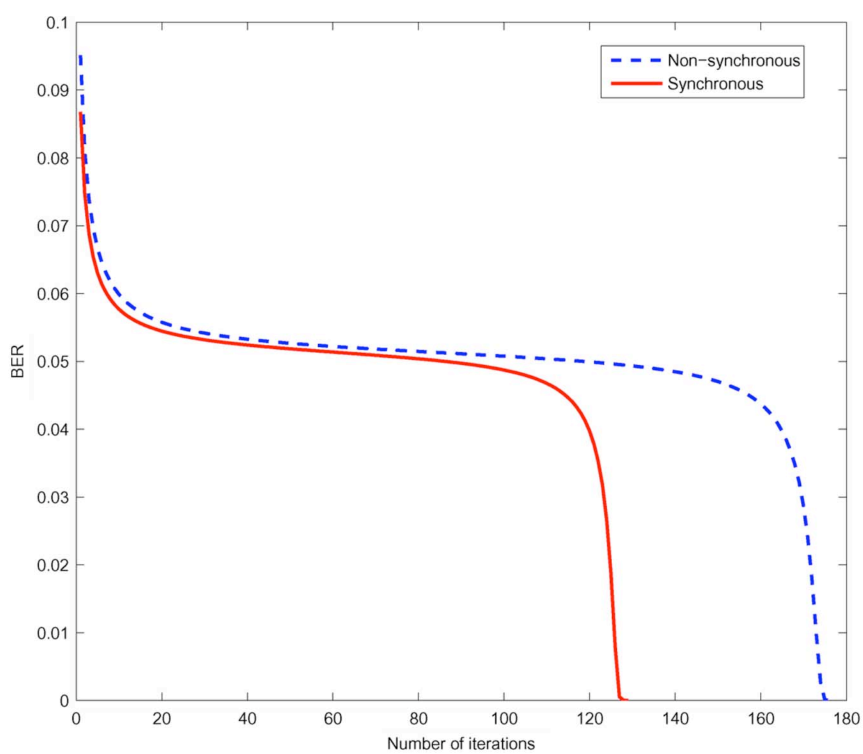

Fig. 4. BER versus number of iterations predicted by density evolution with replica shuffled BP with two subdecoders under nonsynchronous and synchronous updating schemes, for decoding a $(3, 6)$ regular LDPC code.

[19]. We observe a similar behavior as in the case of regular LDPC codes.

Fig. 6 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 of 0.409 dB. 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,

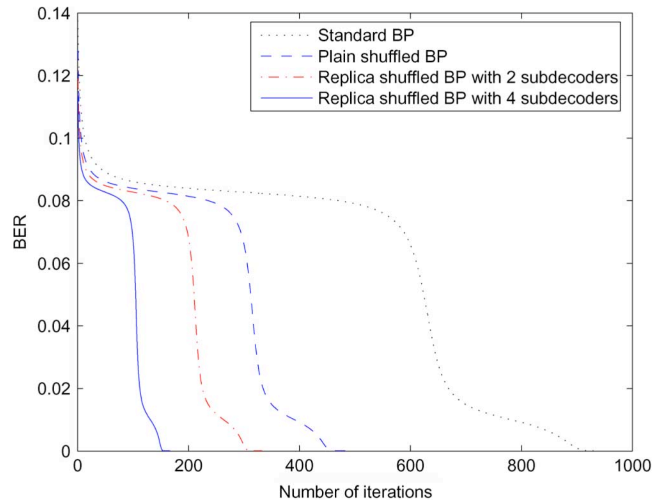

Fig. 5. BER versus number of iterations predicted by density evolution with the standard BP, plain shuffled BP, replica shuffled BP with two and four subdecoders (synchronous scheme), for decoding an irregular LDPC code at the SNR of 0.409 dB.

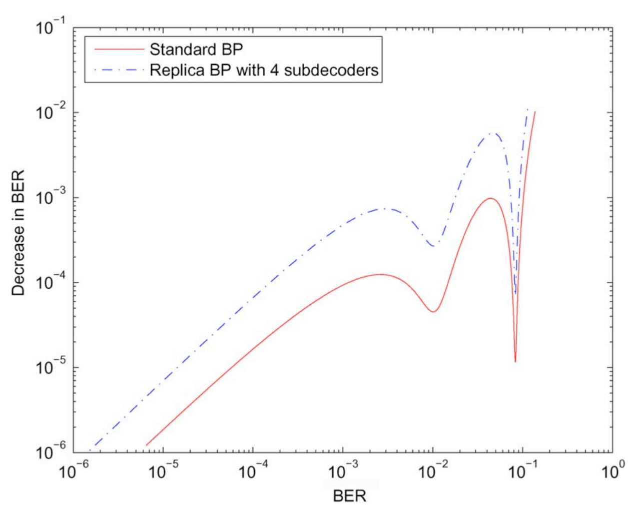

Fig. 6. BER versus decrease in BER predicted by density evolution with the standard BP and replica shuffled BP with four subdecoders and synchronous updating, for decoding an irregular LDPC code at the SNR of 0.409 dB.

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.

C. Analysis by EXIT Charts

EXIT charts [11]–[13] are another effective technique to study the convergence behavior of iterative decoding. They are easy to visualize and to program and are 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 the average mutual information 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

$$\begin{array}{ccc} \text{Set 1} & \text{Set 2} & \text{Set 3} \end{array}$$
$$\left[\begin{array}{c|c|c} 1\,0\,0 & 0\,0\,1 & 0\,1\,0 \\ 0\,1\,0 & 1\,0\,0 & 0\,0\,1 \\ 0\,0\,1 & 0\,1\,0 & 1\,0\,0 \\ \hline 0\,1\,0 & 1\,0\,0 & 0\,0\,1 \\ 0\,0\,1 & 0\,1\,0 & 1\,0\,0 \\ 1\,0\,0 & 0\,0\,1 & 0\,1\,0 \end{array}\right]$$

Fig. 7. An example for illustrating the ideal parity-check matrix of a $(2, 3)$ regular LDPC code with length 9.

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

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

where $\sigma_{\text{ch}}^2 = 8R\cdot\frac{E_b}{N_0}$ and $J(\cdot)$ is defined as

$$J(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{\mathrm{e}^{-(\xi-\sigma^2/2)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}}\cdot\log_2[1+\mathrm{e}^{-\xi}]d\xi \tag{20}$$

$J^{-1}(\cdot)$ is the inverse function of $J(\cdot)$. The approximation functions of $J(\cdot)$ and $J^{-1}(\cdot)$ are given in [20, the Appendix].

1) EXIT Charts 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, quasi-cyclic regular LDPC codes have this feature. This ideal model is also suitable for codes constructed using the progressive edge-growth (PEG) method [21]. The parity-check matrix corresponding to this ideal model is referred to as the "ideal" parity-check matrix. Fig. 7 illustrates an example of the ideal parity-check matrix of a $(2, 3)$ regular LDPC code with length 9.

Based on the above ideal model, since all the edges of the variable nodes in the same set connect to different check nodes, they cannot 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 by I_{U_i} 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

$$I_{U,SHF}(I_V, d_c) = \frac{1}{d_c}\sum_{i=1}^{d_c} I_{U_i}. \tag{21}$$

It is worth stating that for standard BP, I_{U_i}'s are the same for all edges of a check node since all of them are processed simultaneously. However, that is not the case for plain shuffled BP. In

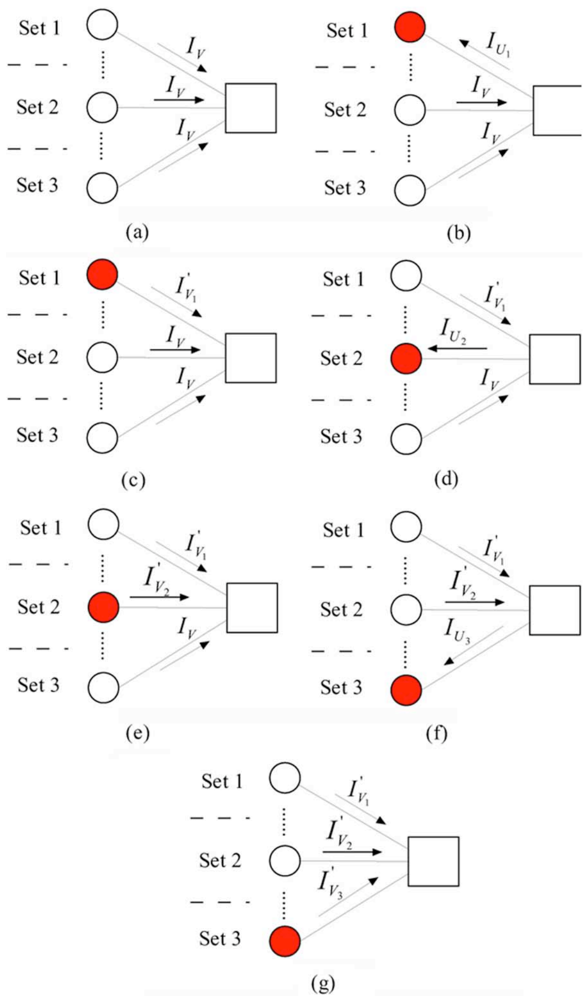

Fig. 8. The mutual information updating process for the LDPC code with the ideal parity-check matrix in Fig. 7.

plain shuffled BP, variable nodes are processed in a fully serial manner and in our ideal model described before this means the edges of a check node are processed serially, so I_{U_i} is improved as the increase of i. For example, consider the ideal parity check matrix in Fig. 7. Fig. 8 illustrates its updating process using plain shuffled BP. Since the processing of each check node is identical, Fig. 8 depicts only one check node. The dark dots in Fig. 8 represent the variable nodes that are being processed. Based on the ideal model assumption, we know that the ith edge of all the check nodes only connects to the variable nodes in the ith set. Supposing variable nodes are processed from Set 1 to Set 3, then all the ith edges of the check nodes are processed before any jth edge of any check node for $j > i$. When the variable nodes in Set 1 are processed, they take the output extrinsic information I_{U_1} from the first edges of the check nodes as their input *a priori* information as shown in Fig. 8(b). Since the *a priori* information of a check node is I_V initially, following (19), we have $I_{U_1} \approx I_{U,STD}(I_V, d_c)$. Based on (18), the output extrinsic information from the variable nodes in Set 1 $I'_{V_1} = I_{V,STD}(I_{U_1}, d_v, \frac{E_b}{N_0}, R)$ as shown in Fig. 8(c). Then the updating of Set 1 is completed. Next we process the variable nodes in Set 2 as shown in Fig. 8(d) and (e). The variable nodes in Set 2 take the output extrinsic information I_{U_2} from the second edges of the check nodes as their input *a priori* information. To calculate I_{U_2}, we follow (19) and take the average of I'_{V_1} and I_V as the input *a priori* information, i.e.,

$$I_{U_2} \approx I_{U,STD}\left(\frac{I'_{V_1} + I_V}{2}, d_c\right).$$

Then based on (18), the output extrinsic information I'_{V_2} of the variable nodes in Set 2 equals $I_{V,STD}(I_{U_2}, d_v, \frac{E_b}{N_0}, R)$. Finally, we process the variable nodes in Set 3. They take the output extrinsic information I_{U_3} from the third edges of the check nodes as their input *a priori* information. Similarly, I_{U_3} is obtained from (19) with the average of I'_{V_1} and I'_{V_2} as the input *a priori* information, i.e.,

$$I_{U_3} \approx I_{U,STD}\left(\frac{I'_{V_1} + I'_{V_2}}{2}, d_c\right).$$

Then the variable nodes in the third set output extrinsic information I'_{V_3}, which equals $I_{V,STD}(I_{U_3}, d_v, \frac{E_b}{N_0}, R)$ as shown in Fig. 8(g), and one iteration is completed.

The above updating process can be generalized to any (d_v, d_c) regular LDPC code with the ideal parity-check matrix, i.e.,

$$I_{U_i} = I_{U,STD}\left(\frac{(d_c - i)I_V + \sum_{k=1}^{i-1} I'_{V_k}}{d_c - 1}, d_c\right) \tag{22}$$

$$I'_{V_i} = I_{V,STD}\left(I_{U_i}, d_v, \frac{E_b}{N_0}, R\right) \tag{23}$$

for $i = 1, 2, \ldots, d_c$.

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

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

The EXIT function for a variable node in the shuffled BP decoding is given by

$$I_{V,SHF}\left(I_{U_{av}}, d_v, \frac{E_b}{N_0}, R\right) = I_{V_{av}}. \tag{24}$$

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

$$I_{U_{av}} = \sum_{i=1}^{d_c} I_{U_i}/d_c = \sum_{i=1}^{d_c} J_1(\sigma_i^2)/d_c \approx J_1\left(\frac{1}{d_c}\sum_{i=1}^{d_c}\sigma_i^2\right).$$

Therefore, it follows

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

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

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) EXIT Charts of Replica Shuffled BP: It is straightforward to extend this method to replica shuffled BP. Using a similar approach, we can show 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 nonsynchronous 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 nonsynchronous updating can be written as

$$I_{U,REP_2,NS}(I_V,d_c)=\frac{1}{d_c}\sum_{i=d_c/2}^{d_c} 2I_{U_i}\quad(\text{even } d_c) \tag{25}$$
$$I_{U,REP_2,NS}(I_V,d_c)=\frac{1}{d_c}\left(\sum_{i=\lceil d_c/2\rceil+1}^{d_c} 2I_{U_i}+I_{U_{\lceil d_c/2\rceil}}\right)\quad(\text{odd } d_c). \tag{26}$$

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. Thus, 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

$$I_{U,REP_D,S}(I_V,d_c)=I_{U,SHF}\left(I_{V_D},d_c\right) \tag{27}$$
$$I_{V_i}=I_{V,SHF}\left(I_{U,SHF}(I_{V_{i-1}},d_c),d_v,\frac{E_b}{N_0},R\right),\ i=2,3,\dots,D \tag{28}$$

with $I_{V_1}=I_V$.

While these derivations allow us to model the convergence of each method, it is well known that the threshold derived on a tree cannot be changed by modifying the scheduling of the algorithm only. So the threshold value remains the same for all methods.

Theorem 1: Based on EXIT chart analysis, the threshold of a code decoded by plain shuffled BP or replica shuffled BP is the same as 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 $I_E=I_{V,STD}(I_A,d_v,\frac{E_b}{N_0},R)$, then $I_A=I_{U,STD}(I_E,d_c)$. In plain shuffled BP decoding, if we use I_E as the input *a priori* information to check nodes, then the extrinsic information of the first edge in a check node is I_A because $I_A=I_{U,STD}(I_E,d_c)$. Variable nodes take I_A as input and send back I_E to check nodes because $I_E=I_{V,STD}(I_A,d_v,\frac{E_b}{N_0},R)$. From this we can see the input mutual information to check nodes is not improved during the process of updating variable nodes serially, i.e., $I_{U_i}\equiv I_A$ and $I'_{V_i}\equiv I_E$. So $I_{U,SHF}(I_E,d_c)=I_A$ and $I_E=I_{V,SHF}(I_A,d_v,\frac{E_b}{N_0},R)$, which means the EXIT curves of variable and check nodes in plain shuffled BP also cross each other at the point A. The same result can be proved for replica shuffled BP. □

In general, the actual Tanner graph does not satisfy all the constraints of our ideal model, but the convergence behavior of the corresponding code can still be well approximated by the ideal model as shown next. Fig. 9 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 charts that the nonsynchronous scheduling converges slower than the synchronous one, as shown in Fig. 4. Fig. 10 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. Fig. 11 depicts EXIT curves superimposed to constant-BER curves [28, Ch. 9]. For the same BER, the iteration number of standard BP is twice that of shuffled BP and eight times that of replica shuffled BP with four subdecoders and synchronous updating.

Fig. 12 depicts the EXIT curves of different decoding methods at the SNR of 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.

3) EXIT Charts 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.

It is very easy to observe this result using the ideal parity-check matrix because the variable nodes in each set do not

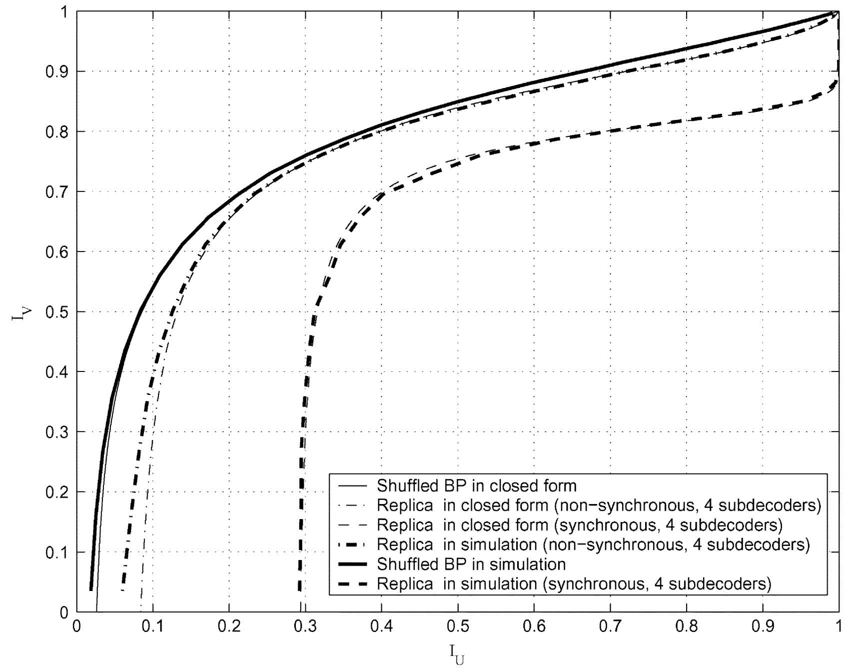

Fig. 9. Comparison between the EXIT curves obtained from the simulation method of [13] and the proposed closed forms for a $(3,6)$ regular LDPC code at the SNR of 1.5 dB.

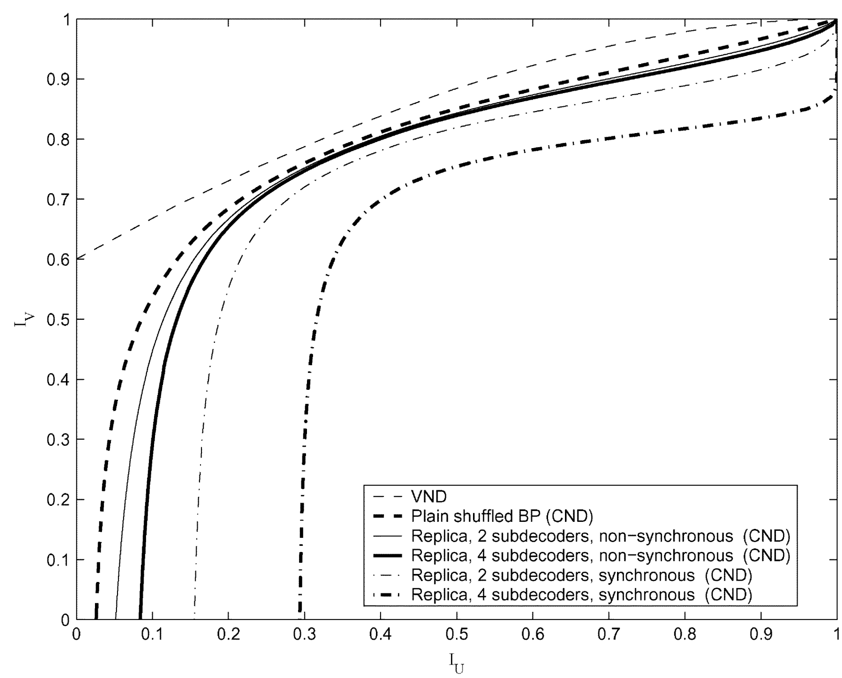

Fig. 10. EXIT curves (in closed form) for shuffled BP and four types of replica shuffled BP decodings at the SNR of 1.5 dB (variable nodes (VND) and check nodes (CND)).

benefit from each other and they can be processed in parallel without changing the performance. 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 speedup 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

$$I_{U,SHF,GR_G}(I_V,d_c) = \frac{1}{d_c}\sum_{i=1}^{d_c} I_{U_i}. \tag{29}$$

If $i \bmod S_G = 1$, then

$$I_{U_i} = I_{U,STD}\left(\frac{(d_c-i)I_V + \sum_{k=1}^{i-1} I'_{V_k}}{d_c-1}, d_c\right) \tag{30}$$

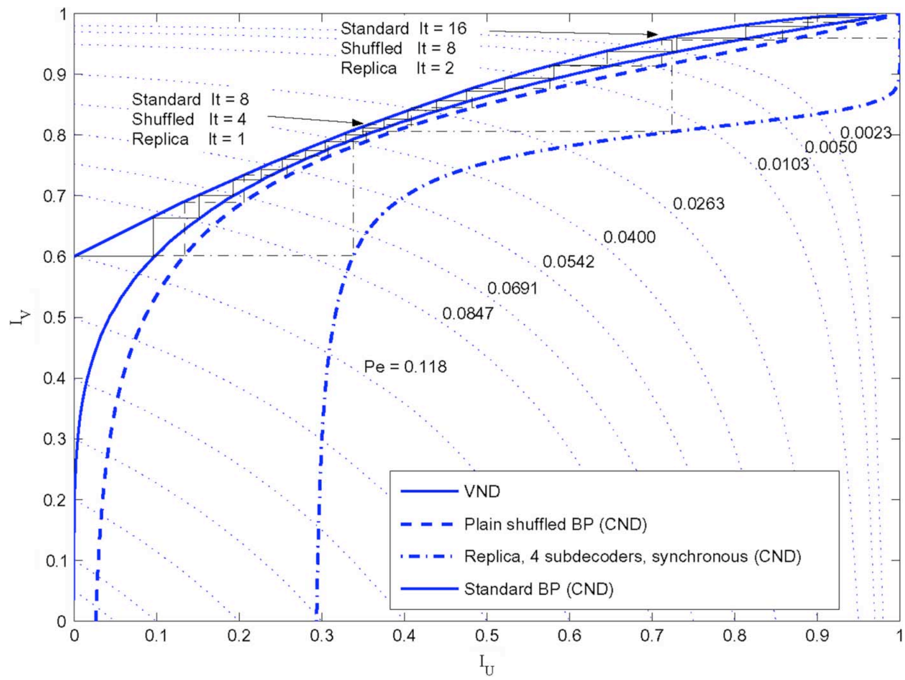

Fig. 11. EXIT curves (in closed form) for standard BP, shuffled BP, and replica shuffled BP with four subdecoders with synchronous updating at the SNR of 1.5 dB, superimposed to constant-BER curves.

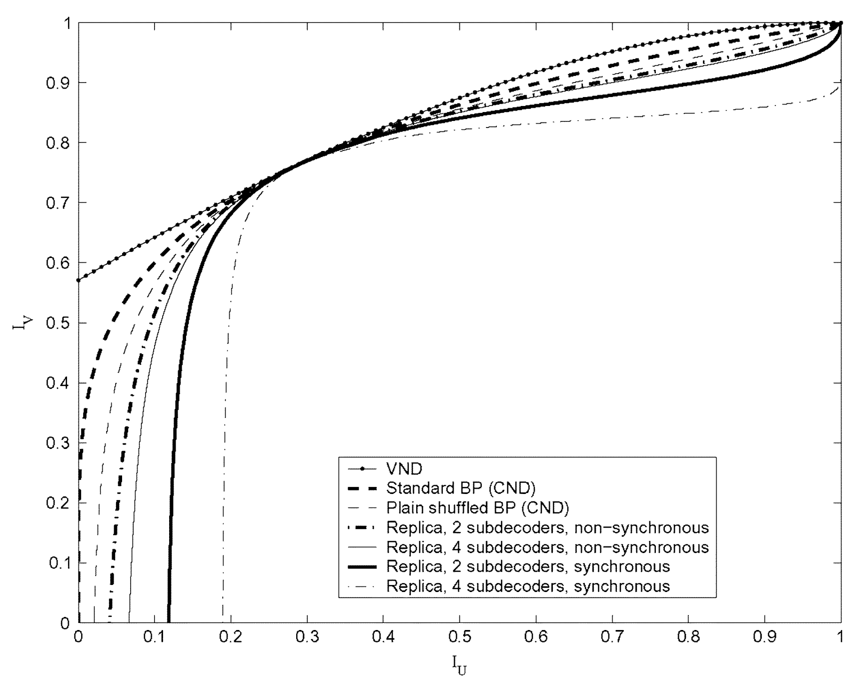

Fig. 12. EXIT curves (in closed form) for standard BP, shuffled BP, and four types of replica shuffled BP at the SNR of 1.11 dB.

$$I'_{V_i} = I_{V,STD}\left(I_{U_i}, d_v, \frac{E_b}{N_0}, R\right). \tag{31}$$

Otherwise

$$I_{U_i} = I_{U_m} \tag{32}$$
$$I'_{V_i} = I'_{V_m} \tag{33}$$

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

The preceding analysis is for vertical shuffled BP, similar results can be obtained for horizontal shuffled BP in [15], [16].

Theorem 3: When decoding a regular LDPC code, horizontal group plain shuffled BP should have at least d_v groups in order to have at any given iteration the same performance as plain shuffled BP based on the ideal model.

In horizontal shuffled BP, instead of dividing variable nodes into d_c sets, we divide check nodes into d_v sets. The check nodes

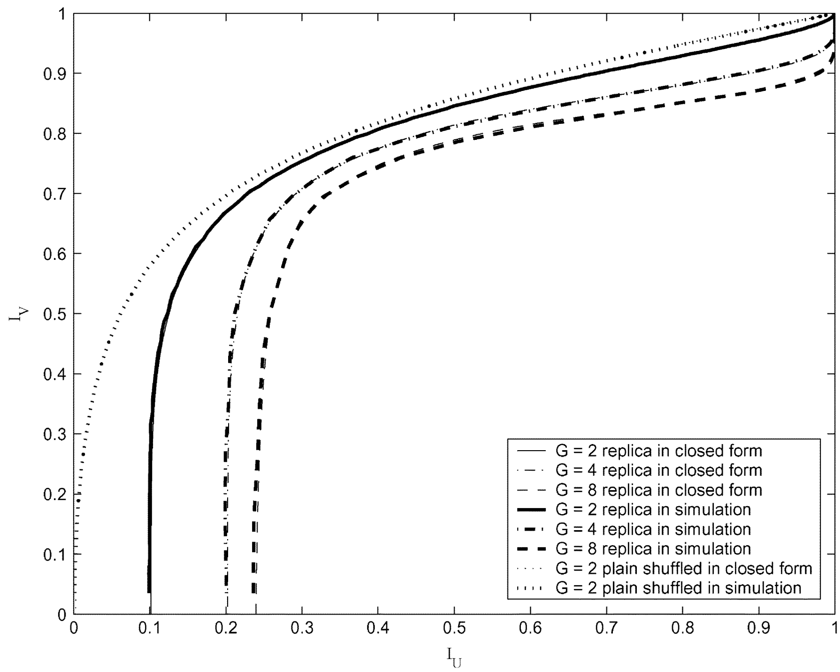

Fig. 13. Comparison between the EXIT curves obtained from the simulation method of [13] and the proposed closed forms for group shuffled BP and group replica shuffled BP with four subdecoders and synchronous updating, for decoding a $(3,6)$ regular LDPC code at the SNR of 1.5 dB.

in each set do not benefit from each other and they can be processed in parallel without changing the performance.

4) EXIT Chart of Group Replica Shuffled BP: The EXIT function of group replica shuffled BP with nonsynchronous updating is almost the same as that of replica shuffled BP (i.e., $G = N$) except that I_{U_i}'s in (25) and (26) are obtained from (30) and (32).

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

$$I_{U,REP_D,S,GR_G}(I_V,d_c) = I_{U,STD}\left(I_{V_G},d_c\right) \tag{34}$$

$$I_{V_i} = I_{V,STD}\left(I_{U,STD}\left(I_{V_{i-1}},d_c\right),d_v,\frac{E_b}{N_0},R\right),\ i=2,3,\ldots,G \tag{35}$$

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

$$I_{U,REP_D,S,GR_G}\left(I_V,d_c\right) = I_{U,SHF,GR_T}\left(I_{V_D},d_c\right) \tag{36}$$

$$I_{V_i} = I_{V,SHF,GR_T}\left(I_{U,SHF,GR_T}\left(I_{V_{i-1}},d_c\right),d_v,\frac{E_b}{N_0},R\right),\quad i=2,3,\ldots,D \tag{37}$$

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 4: 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.

Fig. 13 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.

Fig. 14 depicts the error performance of shuffled BP, group shuffled BP with six 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 Theorems 2 and 4 have almost the same performance as their corresponding non-group counterparts.

D. Simulation Results

Fig. 15 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 $I_{\mathrm{Max}} = 10$, and a group number larger or equal to four, are approximately the same as that of standard BP with $I_{\mathrm{Max}} = 60$.

Fig. 16 depicts the WER of standard and replica shuffled BP decoding of a $(16200,7200)$ irregular LDPC code which was

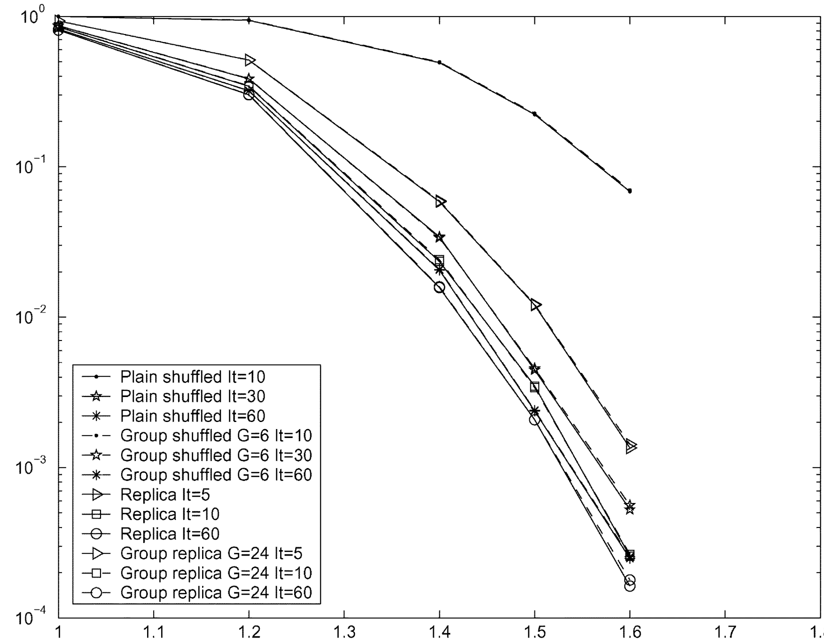

Fig. 14. Word error rate (WER) of shuffled BP, group shuffled BP with six groups, replica shuffled BP with four subdecoders, and synchronous updating and its group version with 24 groups, for decoding a $(8000, 4000)$ $(3, 6)$ regular LDPC code.

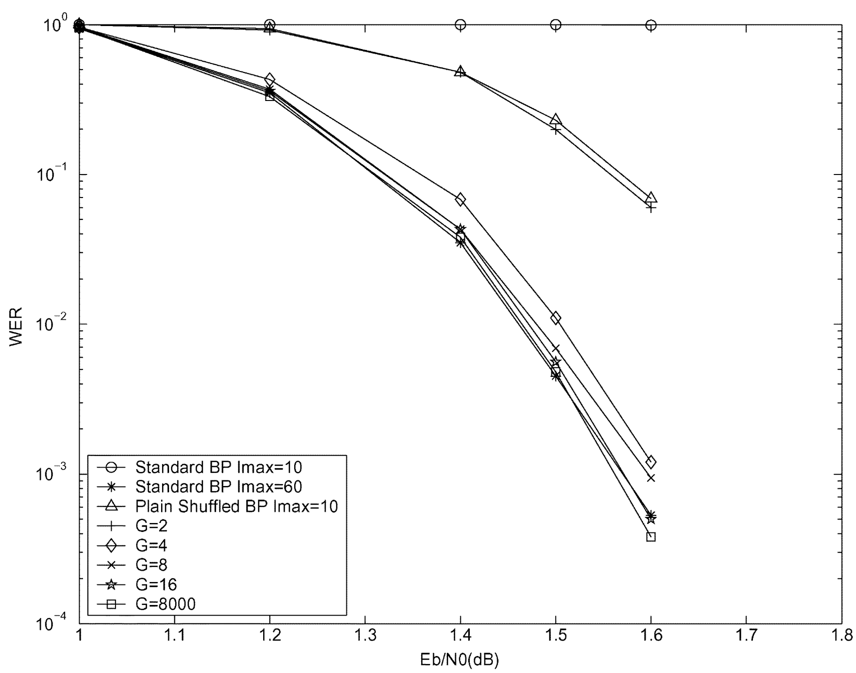

Fig. 15. WER of a $(8000, 4000)(3, 6)$ LDPC code with group shuffled BP algorithm, for $G = 2, 4, 8, 16, 8000$ and at most 10 iterations.

constructed in a semirandom 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 $I_{\text{Max}} = 10$ and $G = 32$ provides a similar performance as that of standard BP with $I_{\text{Max}} = 70$.

For most Gallager type LDPC codes, synchronous replica shuffled BP offers similar ultimate performance as nonsynchronous replica shuffled BP. However, for some LDPC codes with relatively higher density parity-check matrices (such as LDPC codes based on Euclidean and projective geometry (PG)), nonsynchronous replica shuffled BP may provide a better ultimate performance than the synchronous one. In Table II, synchronous and nonsynchronous replica shuffled BP with two subdecoders and 200 iterations for both decoding algorithms are compared (the large iteration number was chosen to ensure convergence in both cases). For this code, the nonsynchronous schedule provides a better performance.

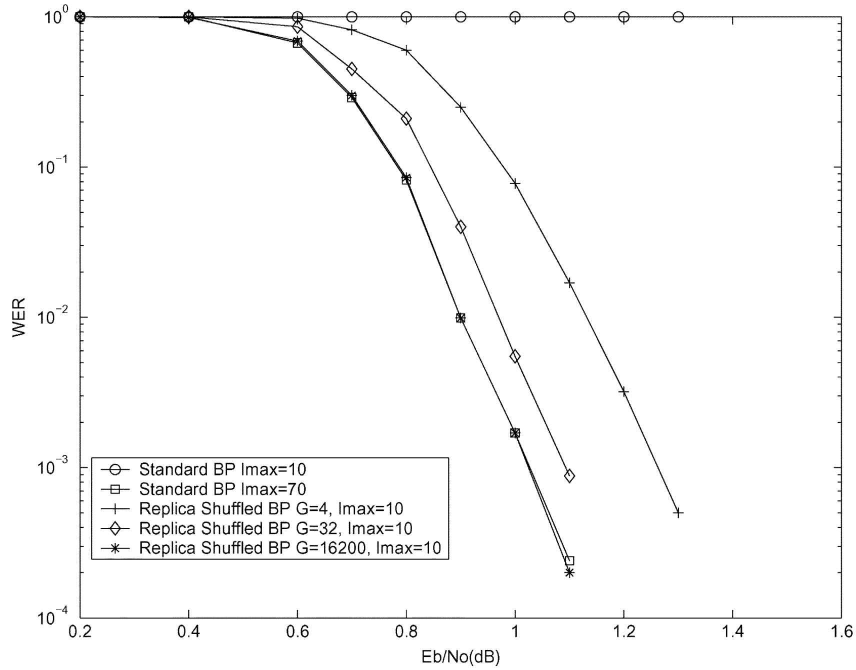

Fig. 16. Error performance for iterative decoding of a $(16200, 7200)$ irregular LDPC code.

TABLE II
PERFORMANCE COMPARISON OF NONSYNCHRONOUS AND SYNCHRONOUS REPLICA SHUFFLED BP DECODING FOR THE $(273, 191)$ PG-LDPC CODE

SNR (dB)	Non-synchronous	Synchronous
2.0	1.5e-2	3.0e-2
2.5	3.0e-3	5.0e-3
3.0	3.5e-4	6.3e-4
3.5	2.0e-5	5.7e-5

III. 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. At each iteration, the decoding of each component decoder is based on not only the received channel values, but also the extrinsic messages delivered by other component decoders. For simplicity, we consider a turbo code that consists of two rate-$1/n$ systematic convolutional codes with encoders in feedback form. Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_K)$ be an information block of length K and $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K)$ be the corresponding coded sequence, where $\boldsymbol{c}_k = (c_{k,1}, c_{k,2}, \ldots, c_{k,n})$, for $k = 1, 2, \ldots, K$, is the output code block at time k. Assume binary phase-shift keying (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 $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K)$ be the received sequence, where $\boldsymbol{y}_k = (y_{k,1}, y_{k,2}, \ldots, y_{k,n})$ is the received block at time k. Let $\hat{\boldsymbol{u}} = \{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_K\}$ denote the estimate of $\boldsymbol{u}$. Let s_k denote the encoder state at time k. Following [4], define $\alpha_k(s) = p(s_k = s, \boldsymbol{y}_1^k)$, $\gamma_k(s', s) = p(s_k = s, y_k | s_{k-1} = s')$, $\beta_k(s) = p(\boldsymbol{y}_{k+1}^K \,|\, s_k = s)$, where $\boldsymbol{y}_a^b = (\boldsymbol{y}_a, \boldsymbol{y}_{a+1}, \cdots, \boldsymbol{y}_b)$, 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].

A. Algorithms

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.

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 $\tilde{\boldsymbol{u}} = (\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_K)$ be the sequence permuted by the interleaver corresponding to the original information sequence $\boldsymbol{u} = (u_1, u_2, \ldots, u_K)$, 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

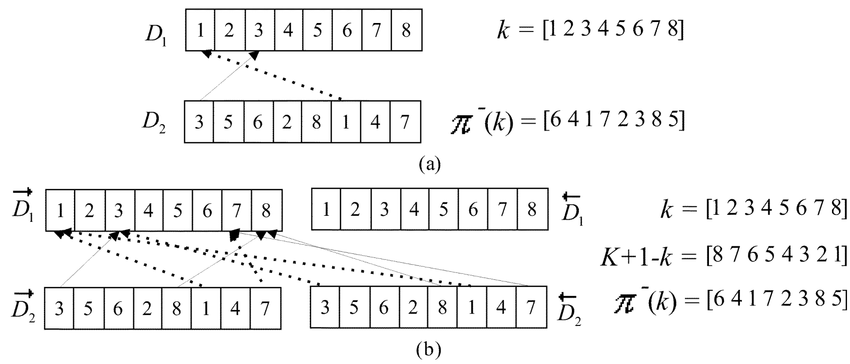

Fig. 17. Examples for illustrating the processing of plain and replica shuffled turbo decodings. (a) Example of plain shuffled turbo decoding with $k = 8$. (b) Example of replica shuffled turbo decoding with $k = 8$.

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) Replica Shuffled Turbo Decoding: In the plain shuffled turbo decoding summarized in Section III-A2, 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 $\overrightarrow{L}^{(i)}_{e1}(\hat{u}_k)$, the larger k is, the more reliable this message is. Similarly, for the extrinsic message $\overleftarrow{L}^{(i)}_{e1}(\hat{u}_k)$ 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., $\overrightarrow{T}^1_k = \overrightarrow{T}^2_k = \overleftarrow{T}^1_k = \overleftarrow{T}^2_k$, where the $\overrightarrow{T}^1_k$ ($\overleftarrow{T}^1_k$) and $\overrightarrow{T}^2_k$ ($\overleftarrow{T}^2_k$) 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 $\boldsymbol{u}$ and of the interleaved sequence $\tilde{\boldsymbol{u}}$, 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 $\overrightarrow{T}^1_{k-1}$, the values of $\overrightarrow{\alpha}^{(1)}_k(s)$ should be updated and the values of $\overrightarrow{\gamma}^{(1)}_k(s)$ are needed. There are two possible cases. The first case is $k > \pi^{-}(k)$, which means the extrinsic value $\overrightarrow{L}^{(i)}_{e2}(\hat{u}_k)$ 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 $\overrightarrow{L}^{(i)}_{e2}(\hat{u}_k)$ is used to compute the values $\overrightarrow{\gamma}^{(1)}_k(s)$, $\overrightarrow{\alpha}^{(1)}_k(s)$, and $\overrightarrow{L}^{(i)}_{e1}(\hat{u}_k)$. The second case is $k < \pi^{-}(k)$, which implies the extrinsic value $\overrightarrow{L}^{(i)}_{e2}(\hat{u}_k)$ of the information bit $\hat{u}_k$ has not been delivered yet by $\overrightarrow{D}_2$. Then in plain shuffled turbo decoding, the values $\alpha^{(1)}_k(s)$ and $L^{(i)}_{e1}(\hat{u}_k)$ 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 $\overleftarrow{L}^{(i)}_{e2}(\hat{u}_k)$ of the information bit $\hat{u}_k$ has already been delivered by decoder $\overleftarrow{D}_2$. Then this newly available $\overleftarrow{L}^{(i)}_{e2}(\hat{u}_k)$, instead of $\overrightarrow{L}^{(i-1)}_{e2}(\hat{u}_k)$ is used to compute the values $\overrightarrow{\gamma}^{(1)}_k(s)$, $\overrightarrow{\alpha}^{(1)}_k(s)$, and $\overrightarrow{L}^{(i)}_{e1}(\hat{u}_k)$. The second subcase is $K+1-k < \pi^{-}(k)$, which implies both extrinsic messages of the information bit $\hat{u}_k$, i.e., $\overleftarrow{L}^{(i)}_{e2}(\hat{u}_k)$ and $\overrightarrow{L}^{(i)}_{e2}(\hat{u}_k)$, are not available yet. In this subcase, the values of $\overrightarrow{\alpha}^{(1)}_k(s)$ and $\overrightarrow{L}^{(i)}_{e1}(\hat{u}_k)$ 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 $\hat{\boldsymbol{u}} = (\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_K)$, where

$$\hat{u}_k = \mathrm{sgn}\Big[(\overrightarrow{L}^{(i)}_{e1}(\hat{u}_k) + \overleftarrow{L}^{(i)}_{e1}(\hat{u}_k))/2 + (\overrightarrow{L}^{(i)}_{e2}(\hat{u}_k) + \overleftarrow{L}^{(i)}_{e2}(\hat{u}_k))/2 + \frac{4}{N_0} y_{k,1}\Big]$$

which is different from the estimate in standard turbo decoding [3] and plain shuffled turbo decoding.

Fig. 17(a) and (b) illustrate the decoding processes of plain and replica shuffled turbo decoding, respectively, with $K = 8$. In Fig. 17(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

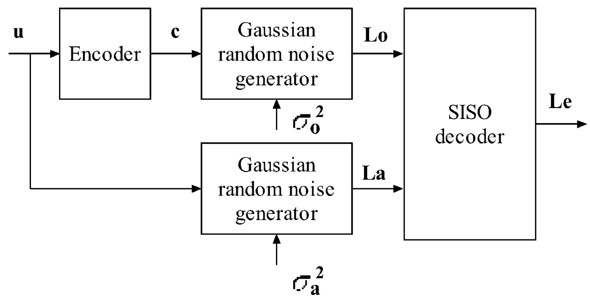

Fig. 18. Monte Carlo model for computing the transfer function of a given turbo code with conventional turbo decoding.

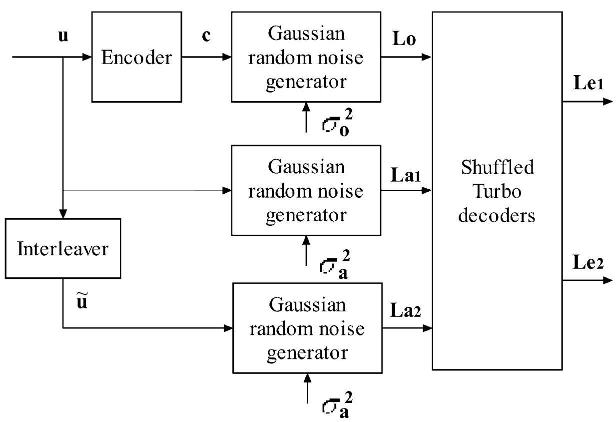

Fig. 19. Monte Carlo model for computing the transfer function of plain shuffled turbo decoding.

is already available. In Fig. 17(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.

B. Analysis by EXIT Charts

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 by 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

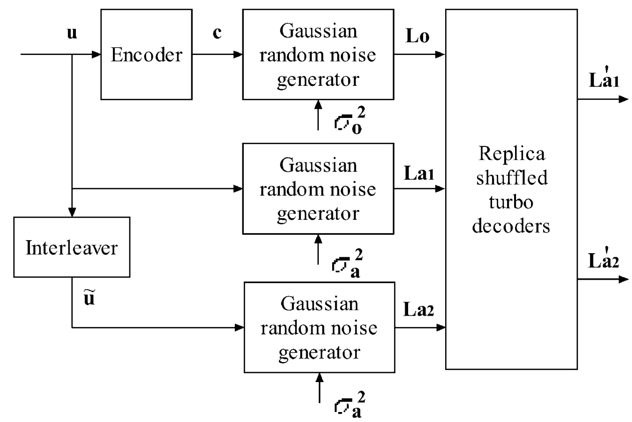

Fig. 20. Monte Carlo model for computing the transfer function of replica shuffled turbo decoding.

$$L_o|c \sim \mathcal{N}\left(\mu_o, \sigma_o^2\right) \tag{38}$$

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

$$L_a|u \sim \mathcal{N}\left(u\sigma_a^2/2, \sigma_a^2\right) \tag{39}$$

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 $J(\sigma_a)$ and $J(\cdot)$ has been defined in (20).

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]

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

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.

1) Analysis of Plain Shuffled Turbo Decoding: In [28, Ch. 9], a Monte Carlo model is used to derive the EXIT chart for a given turbo code. Its structure is shown in Fig. 18, with two Gaussian random noise generator outputs L_o and L_a whose distributions satisfy (38) and (39), respectively. Then L_o and L_a are sent to the single-input single-output (SISO) decoder, which outputs L_e. Based on (20) 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

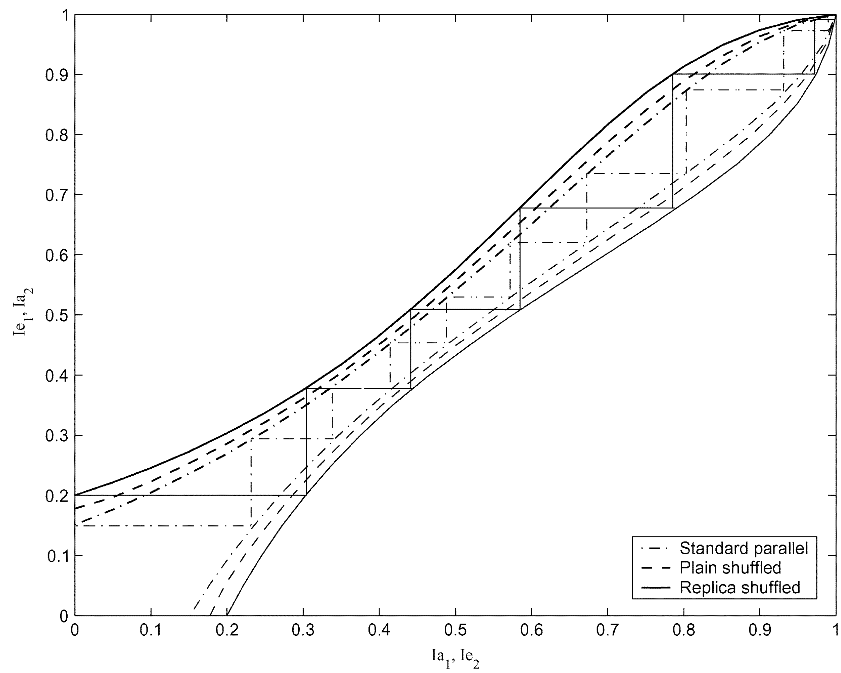

Fig. 21. EXIT charts of a two-component turbo code with interleaver size 16384, for standard parallel, plain shuffled, and replica shuffled turbo decoding, $E_b/N_0 = 0.15$ dB.

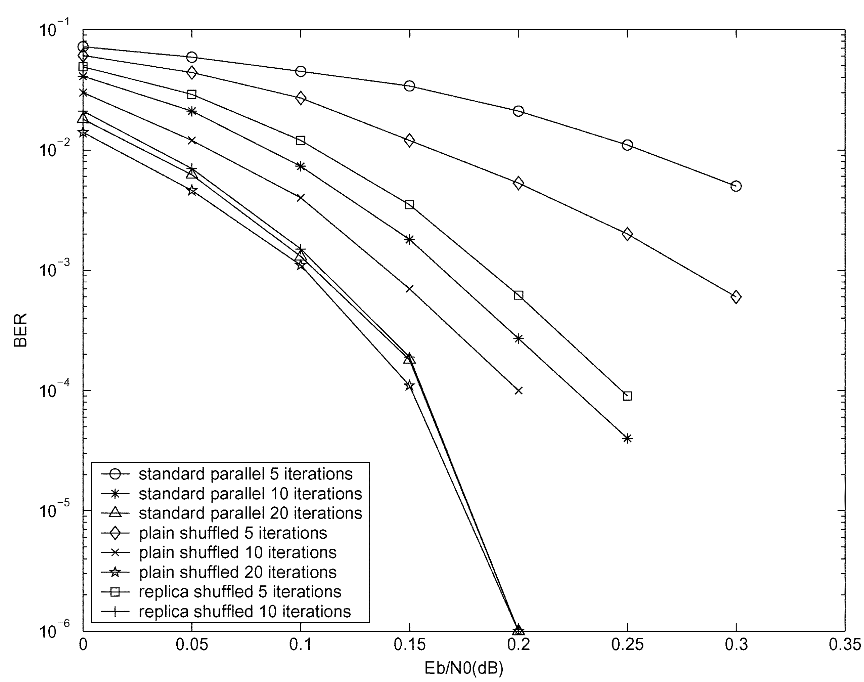

Fig. 22. Bit error performance of a two-component turbo code with interleaver size 16384, for standard parallel, plain shuffled and replica shuffled decodings.

noise generators in the model to compute the transfer function, as shown in Fig. 19. The first two generators are identical to those in Fig. 18, while the third one takes the interleaved sequence $\tilde{\boldsymbol{u}}$ 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).

2) Analysis of Replica Shuffled Turbo Decoding: For replica shuffled turbo decoding, the model to compute the transfer function is depicted in Fig. 20. 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 Fig. 20, 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.

C. Simulation Results

Fig. 21 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 of 0.15 dB. We observe that the replica shuffled turbo decoding converges faster than both the parallel and plain shuffled turbo decoding.

Fig. 22 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 decibel. Furthermore, at the SNR value of 0.15 dB, 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 Fig. 21.

IV. 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.

Based on EXIT charts analysis, we derived an estimate for the least number of groups needed for group plain and replica shuffled BP to achieve the same performance as plain and replica shuffled BP, respectively. This result is useful because the fully serial updating of plain and replica shuffled BP is often not attractive in practice, and group shuffled BP is needed. This result indicates that we can achieve the same performance as the fully serial version with only a few carefully chosen groups.

Since group plain shuffled BP can be viewed as a special case of synchronous group replica shuffled BP, group replica shuffled BP decoding provides more flexibility than group plain shuffled BP decoding and we can find schedulings for which group replica shuffled BP decoder has better performance than group plain shuffled BP using the same decoding time and the same hardware resources. However, we observed that in general, the scheduling of group plain shuffled BP is very good.

For most Gallager type LDPC codes, both synchronous and nonsynchronous replica shuffled BP achieve similar error performance after convergence. In that case, the synchronous scheduling requires less iterations than the nonsynchronous one. However, for some LDPC codes with relatively high density parity-check matrices (such as LDPC codes based on Euclidean and projective geometry), nonsynchronous replica shuffled BP may provide a better performance than the synchronous one.

The replica approach is particularly useful for turbo codes since their decoding is serial. EXIT charts have been used to estimate the convergence of replica shuffled, plain shuffled, and parallel turbo decoding. From both EXIT chart analysis and simulation results, it is observed that replica shuffled turbo decoding can save about half iterations compared with parallel turbo decoding, which is a significant improvement.

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: MIT Press, 1963.

[2] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, 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, no. 10, 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. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.

[5] J. Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, 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. P. C. Fossorier, "Shuffled belief propagation decoding," in *Proc. 36th Annu. Asilomar Conf. Signals, Systems and Computers*, Pacific Grove, CA, Nov. 2002, pp. 8–15.

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

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

[10] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[11] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

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

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

[14] F. Guilloud, "Generic Architecture for LDPC Codes Decoding," Ph.D. dissertation, ENST, Paris, France, 2004.

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

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

[17] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and more," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, 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 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, 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. Inf. Theory*, vol. 47, no. 2, 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, no. 4, pp. 670–678, Apr. 2004.

[21] X. Hu, E. Eleftheriou, and D. Arnold, “Progressive edge-growth Tanner graphs,” in *Proc. 2001 IEEE Global Telecommun. Conf.*, San Antonio, TX, Nov. 2001, pp. 995–1001.

[22] S. Tong and X. Wang, “Convergence analysis of Gallager codes under different message-passing schedules,” *IEEE Commun. Lett.*, vol. 9, no. 3, pp. 249–251, Mar. 2005.

[23] J. Hagenauer, E. Offer, and L. Papke, “Iterative decoding of block and convolutional codes,” *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

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

[25] *Draft DVB-S2 Standard*, [Online]. Available: http://www.dvb.org

[26] A. Viterbi, “An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes,” *IEEE J. Sel. Areas Commun.*, vol. 12, no. 2, pp. 260–264, Feb. 1998.

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

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