Decoding Low-Density Parity Check Codes with Normalized APP-Based Algorithm

Jinghu Chen and Marc P.C. Fossorier[1]
Department of Electrical Engineering, University of Hawaii,
Honolulu, HI 96822, USA
e-mail:jinghu,marc@spectra.eng.hawaii.edu

Abstract- In this paper, we propose a normalized *a posteriori* probability (APP) based algorithm for the decoding of low-density parity check (LDPC) codes. The normalized APP-based algorithm utilizes normalization to improve the accuracy of the soft values delivered by the simplified APP-based algorithm from one iteration to another during the iterative decoding, and can achieve very good tradeoff between decoding complexity and performance.

I. Introduction

Low-density parity check (LDPC) codes [1], first proposed by Gallager in the 1960's, and later rediscovered by MacKay and Neal [2, 3], can achieve near Shannon limit error performance [4, 5] and represent a very promising prospect for error control coding.

One popular way to describe a LDPC code is by means of a bipartite graph, with one subset of nodes being all the bits, and the other subset of nodes being all the checks [6]. Each bit node n, $n = 1, 2, \ldots, N$, has $t(n)$ nodes connected to it, which are all check nodes; we denote the set of these $t(n)$ check nodes as $\mathcal{M}(n)$. Similarly each check node m, $m = 1, 2, \ldots, M$, has $tr(m)$ bit nodes connected to it; this set is denoted as $\mathcal{N}(m)$. This graph representation can facilitate the analysis and design of LDPC codes, as well as their decoding.

Some majority logic decodable codes can be regarded as a special family of regular LDPC codes. Among many such codes are one-step majority logic decodable (OSMLD) codes, including difference-set cyclic (DSC) codes [7]. These kinds of codes are referred to as geometric LDPC codes since they can be constructed systematically based on finite geometries [8, 9]. They are competitive candidates for high rate LDPC codes of medium length, as it has been shown that the performances of these codes are better than those of regular LDPC codes with the same block length N and dimension K. Another advantage of these codes is that they are cyclic or quasi-cyclic codes. As a result, the encoding complexity can be greatly reduced compared to that of conventional LDPC codes, since cyclic code encoders can be easily implemented with register circuits. However, these codes have check sums with comparatively larger weights than those of conventional Gallager codes, and have more checks on each bit. Consequently, the degrees of both the bit nodes and check nodes in the bipartite graph are larger than that of conventional Gallager codes, resulting in a greater decoding complexity.

[1]This research was supported by NSF under Grants CCR-97-32959 and CCR-00-98029, and by the Hawaii Center for Advanced Communications (HCAC).

LDPC codes can be decoded with a soft decision iterative decoding algorithm called belief propagation (BP) algorithm [12], or sum-product algorithm. The BP algorithm is a kind of message passing algorithm working on a bipartite graph. Each node in the bipartite graph can be regarded as a message processor, which accepts messages from its neighbor nodes, processes them and sends updated messages back to its neighbors. We can simplify the processing in either bit nodes or check nodes, or both, and obtain different simplified versions of the BP algorithm. In [10, 11], the uniformly most powerful (UMP) BP-based algorithm is introduced which actually simplifies the processing of BP in check nodes. The simplification causes degradation in performance, especially for LDPC codes with check sums of large weights, such as geometric LDPC codes. In [15], the authors suggest a way to improve the UMP BP-based algorithm by normalization. The normalized BP-based algorithm can nearly achieve the same error performance as BP decoding with reduced complexity. Also presented in [10] is the UMP *a posteriori* probability (APP) based algorithm which simplifies the processing in both bit and check nodes. Unfortunately, the UMP APP-based algorithm can not achieve very good performance for LDPC codes in general. In this paper, we propose to improve the error performance of the UMP APP-based algorithm by normalization. The proposed normalized UMP APP-based algorithm is effective for both LDPC codes with check sums of small weights such as Gallager codes, and those with check sums of large weights such as geometric codes. It can further reduce the decoding complexity of the UMP BP-based algorithm of [15] by a factor proportional to $t(n)$.

II. The BP Algorithm and Its Simplifications

In the following, we assume BPSK modulation which maps a codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_N)$, with $c_n = 0$ or 1, into a transmitted sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, according to $x_n = 2c_n - 1$, for $n = 1, 2, \ldots, N$. Then $\boldsymbol{x}$ is transmitted over a channel corrupted by additive white Gaussian noise (AWGN) with zero mean and power spectral density $\frac{N_0}{2}$ W/Hz. The received value corresponding to x_n after the demodulator is $y_n = x_n + \nu_n$, where ν_n is a random variable with zero mean and variance $\frac{N_0}{2}$.

At each iteration of BP decoding, each check node receives messages from all bit nodes connected to it, and after processing, it sends messages back to these bit nodes. Then a similar procedure is applied to each bit node. We assume all the messages passing between bit and check nodes are in the form of

0-7803-7206-9/01/$17.00 © 2001 IEEE

log-likelihood ratios (LLR's). Moreover, we define the following notations associated with a given iteration:

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

A *BP Algorithm*

The BP algorithm can be described as follows.

Initialization: For each n and each $m_i \in \mathcal{M}(n)$, set $z_{m_i n} = F_n$.

Iterative processing: For each iteration, the processing includes three consecutive steps.

1. Processing in check nodes: (See Fig. 1 (a))
   For each check node m and each bit node n_i connected to node m, $n_i \in \mathcal{N}(m)$, update L_{mn_i} by

$$T_{mn_i} = \prod_{n' \in \mathcal{N}(m)\backslash n_i} \frac{1-\exp(z_{mn'})}{1+\exp(z_{mn'})} ; \quad (1)$$

$$L_{mn_i} = \ln \frac{1-T_{mn_i}}{1+T_{mn_i}} . \quad (2)$$

2. Processing in bit nodes: (See Fig. 1 (b))
   For each bit node n and each check node m_i connected to node n, $m_i \in \mathcal{M}(n)$, update $z_{m_i n}$ by

$$z_{m_i n} = F_n + \sum_{m' \in \mathcal{M}(n)\backslash m_i} L_{m'n} . \quad (3)$$

   Also, for each n, update z_n for hard decision by

$$z_n = F_n + \sum_{m \in \mathcal{M}(n)} L_{mn} . \quad (4)$$

3. Hard decision and stopping criterion test:
   Create the hard decision vector $\hat{\boldsymbol{x}} = [\hat{x}_n]$ such that $\hat{x}_n = 1$ if $z_n > 0$, and $\hat{x}_n = 0$ if $z_n < 0$. If $\hat{\boldsymbol{x}}$ is a codeword, it is considered as a valid decoded word and the decoding process ends; if the number of iterations exceeds some maximum number and $\hat{\boldsymbol{x}}$ is not a valid codeword, a failure is declared and the decoding process ends; otherwise the decoding repeats from Step 1.

B *BP-Based Algorithm*

The processing in the check nodes for BP algorithm can be simplified. First, for $n_i \in \mathcal{N}(m)$, define σ_{mn_i} as the hard decision according to z_{mn_i}, i.e., $\sigma_{mn_i} = 1$ if $z_{mn_i} > 0$; $\sigma_{mn_i} = 0$, otherwise. Also, define σ_m as the modulo-2 sum of the hard decisions of all the bits of the check sum m, according to $\{z_{mn_i}\}$, i.e., $\sigma_m = \sum_{n_i \in \mathcal{N}(m)} \sigma_{mn_i} \bmod 2$. Then $\sigma_m \oplus \sigma_{mn_i}$ represents the modulo-2 sum of the hard decisions of all the bits in $\mathcal{N}(m)\backslash n_i$. Note that the sign of T_{mn_i} in (1) is $(-1)^{\sigma_m \oplus \sigma_{mn_i}}$. Therefore,

$$\prod_{n' \in \mathcal{N}(m)\backslash n_i} \frac{1-\exp(z_{mn'})}{1+\exp(z_{mn'})} \approx (-1)^{\sigma_m \oplus \sigma_{mn_i}} \frac{\exp(\min_{n' \in \mathcal{N}(m)\backslash n_i} |z_{mn'}|) - 1}{\exp(\min_{n' \in \mathcal{N}(m)\backslash n_i} |z_{mn'}|) + 1} . \quad (5)$$

Based on (5), (2) can be simplified into

$$L_{mn_i} \approx (-1)^{\overline{\sigma_m \oplus \sigma_{mn_i}}} \min_{n' \in \mathcal{N}(m)\backslash n_i} |z_{mn'}| . \quad (6)$$

Replacing (2) with (6) for the processing in check nodes, we obtain the BP-based algorithm of [10, 11], which performs additions only. The BP-based algorithm also reduces the storage requirement, since for each check node m, only the smallest two amplitudes of all $|z_{mn_i}|$ need to be stored for representation of all L_{mn_i}. In addition, the positive factor $\frac{4}{N_0}$ in F_n can be omitted and no *a priori* information about the AWGN channel is required.

C *Iterative APP Algorithm*

The iterative APP algorithm can be regarded as a simplification of the BP algorithm, and the simplification is done for the processing in bit nodes. In the BP algorithm, the processing in each bit node n includes two calculations: (i) for each check node m_i connected to it, following (3), the *a priori* message F_n is added to the incoming messages from all check nodes connected to it except for that from check node m_i, such that $z_{m_i n}$, the outgoing message to check node m_i, is somewhat uncorrelated with the incoming message $L_{m_i n}$ from check node m_i; (ii) following (4), the *a priori* message F_n is added to all the messages from check nodes $\mathcal{M}(n)$ to obtain the *a posteriori* information z_n used to make hard decision for bit n.

In the iterative APP algorithm, we only do calculation (ii) to compute z_n's. Then the z_n's are not only used for hard decision, but also substituted for all $z_{m_i n}$ ($m_i \in \mathcal{M}(n)$), as shown in Fig. 1 (c). Indeed, with this simplification, correlation is introduced between the bidirectional messages along each edge. On the other hand, the computational complexity and storage size of the iterative APP algorithm can be greatly reduced since we only need to compute and store one value for each bit node. Note that this decoding can be viewed as an iterative implementation of [14].

D *Iterative APP-Based Algorithm*

Since the processing in check nodes for iterative APP algorithm is the same as that for BP algorithm, it can also be simplified with (6) to avoid exponential and logarithmic operations and to reduce storage requirement. This results in the UMP iterative APP-based algorithm.

III. Normalized Iterative APP-Based Algorithm

For conventional LDPC codes, the iterative APP algorithm has much worse performance than that of the BP algorithm. For example, it has been shown in [10] that for (504, 252) and (1008, 504) Gallager codes with $tr = 6$ and $t = 3$, the performance with the iterative APP algorithm is more than 1.0 dB worse than that of the BP algorithm. The reason lies in the fact that the degree of each bit node, which is equal to t, is very small, and hence for every bit node, the correlation between the outgoing and incoming messages along each of the t edges is very significant. However, for geometric LDPC codes, because the degree of each node is usually very large, the fore-mentioned correlation is somewhat reduced, and it doesn't have predominant effect on the error performance. As a result, the iterative APP algorithm is not a bad choice for geometric LDPC codes since it can provide good tradeoff between decoding complexity and performance. It has been shown in [8, 13] that the gap between the error performances of BP decoding and iterative APP decoding is only about 0.2 dB for the (273, 191) DSC code.

Furthermore, the iterative APP algorithm can tremendously reduce the storage requirement and computational complexity of BP decoding. For example, for the (1057, 813) DSC code with $tr = t = 33$, if BP decoding is adopted, then 1057×33 units are needed to store $\{z_{mn}\}$ and the same amount of units are needed to store $\{L_{mn}\}$. However, with iterative APP algorithm, we only need 1057 and 1057×33 units to store $\{z_n\}$ and $\{L_{mn}\}$, respectively. Nevertheless the total storage is only reduced by nearly one half. The decoding is also faster since the processing in bit nodes is greatly simplified.

As mentioned in Section II-D, the iterative APP algorithm can be further simplified to iterative APP-based algorithm if the processing in check nodes is approximated with (6). Consequently, for the fore-mentioned code, 1057 units to represent $\{z_n\}$ and $\{L_{mn}\}$ are now required, corresponding to a reduction of $tr = t$. Unfortunately, while for conventional LDPC codes, the iterative APP-based algorithm brings no degradation to the already bad performance of iterative APP decoding, for geometric LDPC codes, the performance of the iterative APP-based decoding is much worse than that of iterative APP decoding. Consequently, the approximation and resulting complexity reduction is paid at the expense of a severe performance degradation. Fortunately, this degradation can be reclaimed with the same normalization approach as that of [15], as considered below.

For a given pair of m and n_i, denote L_1 and L_2 as the value L_{mn_i} computed by (2) and (6), respectively. It can be shown that the two following facts hold [15]:

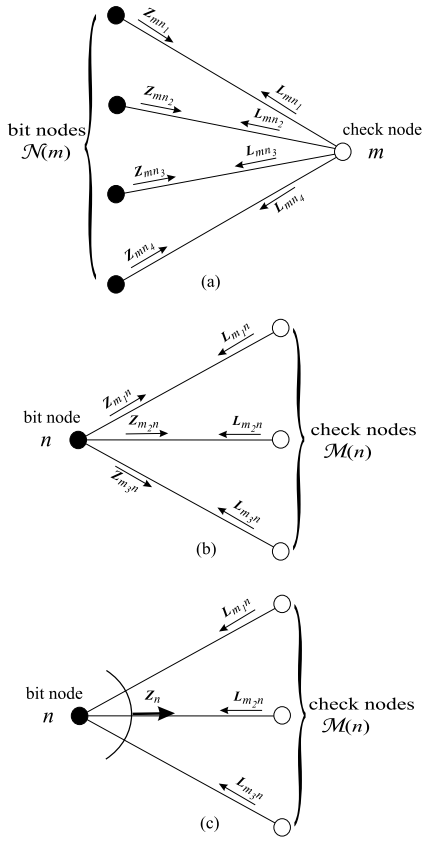

Fig. 1: (a) Processing in check nodes; (b) Processing in bit nodes with BP algorithm;(c) Processing in bit nodes with APP algorithm.

1. $\text{sgn}(L_1) = \text{sgn}(L_2)$;
2. $|L_2| > |L_1|$.

We have shown in [15] that with these two facts, the BP-based algorithm can be improved by normalization, i.e. by dividing L_2 by a factor α greater than 1 to get a much better approximation of L_1. The theoretical computation of α has been derived in [15] and is reviewed below. Since the iterative APP-based algorithm has the same processing in check nodes as BP-based algorithm, this suggests that the iterative APP-based algorithm can be improved with the same normalization approach. We refer to this algorithm as normalized iterative APP-based algorithm.

We propose to determine the scaling factors for the normalized iterative APP-based algorithm by the same method as proposed in [15], i.e., by forcing the mean of the normalized magnitude $|L_2|/\alpha$ to equal the mean of the magnitude $|L_1|$, or

$$\alpha = \frac{\text{E}(|L_2|)}{\text{E}(|L_1|)} \quad . \tag{7}$$

The scaling factors depend on the signal-to-noise ratio (SNR). For the first iteration, they can either be determined based on Monte Carlo simulations, or obtained theoretically with the formulas derived in [15]. Also, for a given LDPC code, we can

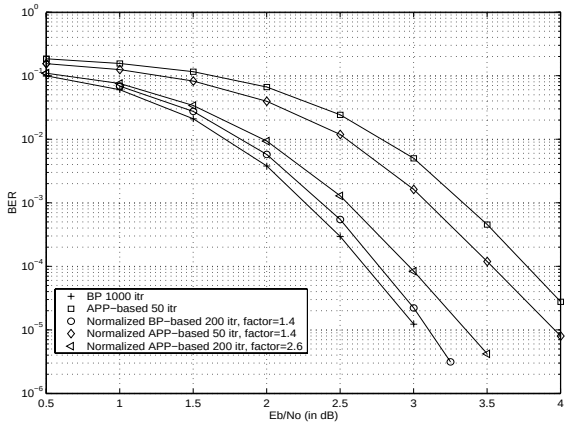

Fig. 2: Bit error performance for iterative decoding of the (504, 252) LDPC code with BP, APP-based, normalized BP-based and normalized APP-based algorithms.

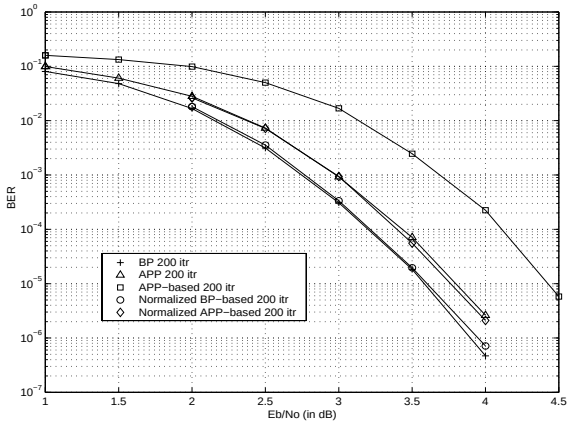

Fig. 3: Bit error performance for iterative decoding of the (273, 191) DSC code with BP, APP, APP-based, normalized BP-based and normalized APP-based algorithms.

choose one scaling factor for all iterations and all SNR values, so that the normalized iterative APP-based algorithm becomes independent of the SNR value.

In iterative APP-based decoding, correlation values are introduced by: (i) the overestimation of the values L based on (6) in the processing at the check nodes; (ii) the approximation of BP by APP in the processing at the bit nodes. This suggests that even larger scaling values than for BP-based decoding should be chosen for the APP-based algorithm, especially when the number of check sums is small.

Normalization is not a new concept. For example, in [16], the authors propose to clip and scale numbers in the VLSI design of a decoder for the (73, 45) DSC code and use the same simplification of (3) by (4) discussed in Section II-C. However, the scaling is mainly intended to represent numbers efficiently in hardware implementation and the entire *a posteriori* values rather than only the extrinsic information values are scaled. In [17, 18], the authors show that the performance of SOVA decoding can be improved by normalization, since the soft values delivered by the SOVA algorithm are usually overestimated, compared to those of Max-Log-MAP algorithm. Normalization values have also been proposed in [19, 20] to approach the error performance of Max-Log-MAP decoding with a suboptimum version of the Max-Log-MAP algorithm applied to iterative decoding of block product codes. However, for the Max-Log-MAP algorithm, the two facts discussed above and essential to the effectiveness of our proposed scaling method no longer hold [15].

IV. Simulation Results

In the following, the maximum number of iterations has been chosen as the smallest value for which near convergence performance of the algorithm considered is achieved. Fig. 2 shows the bit error performance of different algorithms for the regular (504, 252) LDPC code with $tr = 6$ and $t = 3$. If we only consider the overestimation of the values L based on (6), then the scaling factor $\alpha = 1.4$ is chosen for both normalized BP-based and normalized APP-based algorithms. However, with this factor, the performance of the normalized iterative APP-based algorithm is still much worse than that of BP and normalized BP-based algorithms, since due to the small number of check sums $t = 3$, the correlation introduced by approximating BP with APP is very serious. By increasing this factor to $\alpha = 2.6$, the normalized iterative APP-based algorithm can achieve its best performance which is only about 0.3 dB worse than that of the BP algorithm.

Fig. 3 and 4 depict the bit error performance of the (273, 191) and (1057, 813) DSC codes, respectively. Similar curves were obtained for block error performances. For the normalized iterative APP-based algorithm, we choose the scaling factors $\alpha = 2.0$ for the former code and $\alpha = 4.0$ for the latter one, as we computed for the normalized BP-based algorithm based on the analysis of [15]. Fig. 3 shows that for the (273, 191) DSC code, the performance of the normalized iterative APP-based algorithm is almost the same as that of the iterative APP algorithm. It is about 0.6 dB better than that of the iterative APP-based algorithm, and only about 0.2 dB worse than that of the BP algorithm. No noticeable improvements are observed by changing the scaling factor for this code. Fig. 4 shows that the performance of the normalized iterative APP-based algorithm is more than 1.0 dB better than that of the iterative APP-based algorithm, and only slightly worse than that of the BP algorithm. Surprisingly, the performance of the normalized iterative APP-based algorithm is even about 0.15 dB better than that of the iterative APP algorithm.

We also simulated the (4161, 3431) DSC code for which $tr = t = 65$. In this case, we compute the scaling factor $\alpha = 8.0$ based on [15]. The performance of the normalized iterative APP-based is only about 0.1 dB worse than that of the BP algorithm as shown in Fig. 5.

V. Conclusion

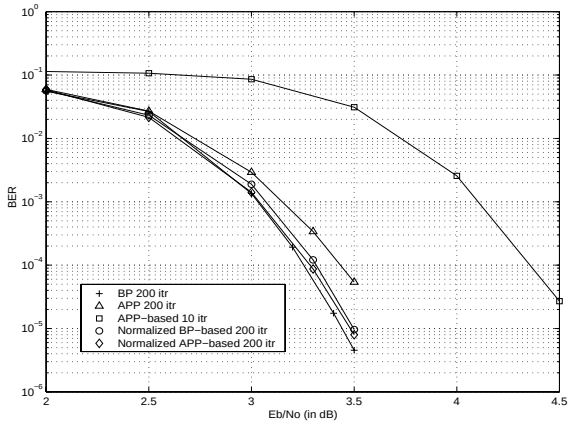

Fig. 4: Bit error performance for iterative decoding of the (1057, 813) DSC code with BP, APP, APP-based, normalized BP-based and normalized APP-based algorithms.

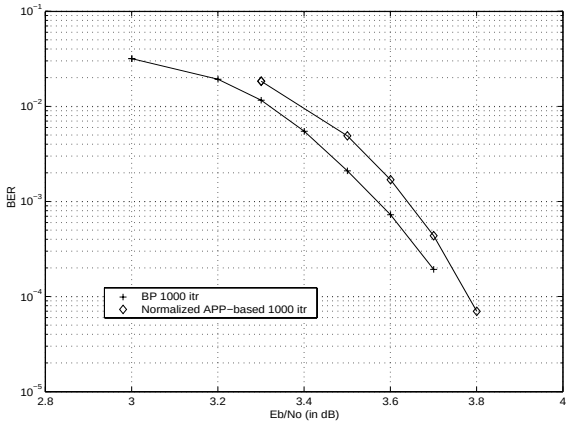

Fig. 5: Bit error performance for iterative decoding of the (4161, 3431) DSC code with BP and normalized APP-based algorithms.

In this paper, we have summarized different simplifications of the BP algorithm for decoding LDPC codes. We proposed a normalized iterative APP-based algorithm to improve the performance of the iterative APP-based algorithm. The simulation results shows that the normalized iterative APP-based algorithm can achieve good performance for both conventional LDPC codes with check sums of small weights and geometric LDPC codes with check sums of large weights. However, while for the latter the scaling factors can be derived based on [15], larger scaling factors than those derived in [15] are needed to achieve the best error performance for the former codes.

From these results, we conclude that for LDPC codes with check sums of large weights, the proposed normalized iterative APP-based decoding performs nearly as well as BP decoding. For LDPC codes with check sums of small weights, the approach of [15] is needed to achieve nearly the same error performance as BP decoding, while the proposed approach provides good trade-off between error performance and decoding complexity.

ACKNOWLEDGMENTS

The authors wish to thank R.M. Tanner for interesting discussions and providing [16].

REFERENCES

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

[2] D.J.C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399-431, Mar. 1999.

[3] D.J.C. MacKay and R.M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electronics Letters* 32(18), pp. 1645-1646, Aug. 1996.

[4] T. Richardson and R. Urbanke, "The Capacity of Low-Density Parity Check Codes under Message-Passing Decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599-618, Feb. 2001.

[5] T. Richardson, A. Shokrollahi and R. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity Check Codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619-637, Feb. 2001.

[6] R. M. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533-547, Sept. 1981.

[7] E. J. Weldon, "Difference-Set Cyclic Codes," *Bell Syst. Tech. J.*, vol. 45, pp. 1045-1055, Sept. 1996.

[8] R. Lucas, M. Fossorier, Y. Kou and S. Lin, "Iterative Decoding of One-Step Majority Logic Decodable Codes Based on Belief Propagation," *IEEE Trans. Comm.*, vol. 48, pp. 931-937, June 2000.

[9] Y. Kou, S. Lin and M. Fossorier, "Low Density Parity Check Codes Based on Finite Geometries: A Rediscovery and More," *IEEE Trans. Inform. Theory*, to appear.

[10] M. Fossorier, M. Mihaljević and H. Imai, "Reduced Complexity Iterative Decoding of Low Density Parity Check Codes Based on Belief Propagation," *IEEE Trans. Commun.*, vol. 47, pp. 673-680, May 1999.

[11] A. J. Felstrom and K. S. Zigangirov, "Time-Varying Periodic Convolutional Codes with Low-Density Parity-Check Matrix," *IEEE Trans. Inform. Theory*, vol. 45, pp. 2181-2191, Sept. 1999.

[12] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.

[13] R. Lucas, M. Bossert, and M. Breitbach, "On Iterative Soft-Decision Decoding of Linear Binary Block Codes And Product Codes", *IEEE J. Select. Areas Commun.*, vol. 16, pp.276-296, Feb. 1998.

[14] J. L. Massey, *Threshold Decoding*. Cambridge, MA: M.I.T. Press, 1963.

[15] J. Chen and M. Fossorier, "Near Optimum Universal Belief Propagation Based Decoding of LDPC Codes," *IEEE Trans. Commun.*, to appear.

[16] K. Karplus and H. Krit, "A Semi-Systolic Decoder for the PDSC-73 Error-Correcting Code," *Discrete Applied Mathematics 33*, North-Holland, pp. 109-128, 1991.

[17] L. Papke and P. Robertson, "Improved Decoding with the SOVA in a Parallel Concatenated (Turbo-code) Scheme," *Proc. ICC96*, pp. 102-106, June 1996.

[18] L. Lin and R. S. Cheng, "Improvements in SOVA-Based Decoding for Turbo Codes," *Proc. ICC97*, pp. 1473-1478, June 1997.

[19] R. M. Pyndiah, "Near Optimum Decoding of Product Codes: Block Turbo Codes," *IEEE Trans. Commun.*, vol. 46, pp.1003-1010, Aug. 1998.

[20] P. A. Martin and D. P. Taylor, "On Multilevel Codes and Iterative Multistage Decoding," *IEEE Trans. Commun.*, to appear.