Joint Message-Passing Decoding of LDPC Codes and Partial-Response Channels

Brian M. Kurkoski, *Student Member, IEEE*, Paul H. Siegel, *Fellow, IEEE*, and Jack Keil Wolf, *Life Fellow, IEEE*

Invited Paper

***Abstract*—Ideas of message passing are applied to the problem of removing the effects of intersymbol interference (ISI) from partial-response channels. Both bit-based and state-based parallel message-passing algorithms are proposed. For a fixed number of iterations less than the block length, the bit-error rate of the state-based algorithm approaches a nonzero constant as the signal-to-noise ratio (SNR) approaches infinity. This limitation can be removed by using a precoder. It is well known that low-density parity-check (LDPC) codes can be decoded using a message-passing algorithm. Here, a single message-passing detector/decoder matched to the combination of a partial-response channel and an LDPC code is investigated.**

***Index Terms*—Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm, iterative decoding, low-density parity-check (LDPC) code, maximum *a posteriori* (MAP) decoder, message-passing algorithm, partial-response channel, precoding, sum-product algorithm.**

I. Introduction

THE exponential growth in the bit densities of state-of-the-art magnetic recording systems has increased the storage capacity of hard disk drives, which has been a boon for computer users. Capacities have increased to the point that commodity hard drives can now be found in new applications such as consumer video recording devices. However, further increases in bit density are limited in part because shrinking bit size on the surface of the magnetic medium produces degradation in signal-to-noise ratio (SNR) to the point that conventional detectors and error-correcting schemes are overwhelmed.

A new class of error-correcting schemes based upon iterative decoding, typified by low-density parity-check (LDPC) codes [1] and turbo codes [2], can achieve target bit-error rates (BER) at SNRs much lower than conventional error-correcting approaches. As such, the application of iterative decoding to magnetic recording systems holds promise for increasing bit densities, although we are unaware of any mass-produced hard drive yet using such an error-correction system.

Any error-correcting scheme for magnetic recording systems must cope with the intersymbol interference (ISI) present at high bit densities. Partial response is a discrete channel model which approximates the unequalized channel ISI. Conventional Viterbi detectors, which are the *de facto* standard for eliminating ISI in magnetic recording systems, are not applicable to systems using iterative decoders, because they produce hard bit decisions. Iterative decoders are most effective using soft or probabilistic information as inputs. In applications of iterative decoding to magnetic recording, some type of soft-output algorithm (often the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm [3]) replaces the Viterbi algorithm as the partial-response channel detector.

Turbo codes have been shown to have excellent BER performance on the additive white Gaussian noise (AWGN) channel [4], [5]. These gains are preserved when turbo codes are applied to the partial-response channel. A greater performance improvement comes when the channel detector is incorporated within the iterative decoder [6]–[9]. Thus, iterative decoding has been extended beyond the error-correcting code, and the inclusion of the channel detector in an iterative decoder has been called "turbo equalization."

In contrast to turbo codes which have received substantial attention since their 1993 introduction, LDPC codes have led a quiet existence from the 1960s, when they were first proposed by Gallager, until the mid-1990s when interest in them was rekindled. Gallager proposed a simple decoder that used probabilistic decoding on a code defined by a sparse, random parity-check matrix [1]. In 1981, Tanner showed how to construct a bipartite graph, sometimes called a Tanner graph, representing the parity-check matrix of any code, and recognized that Gallager's decoding ideas could be applied to this graph [10]. Additional research in the mid-1990s has refined our notion of what we shall refer to as the Gallager–Tanner (GT) algorithm, and broadened our understanding of how codes can be decoded on a graph.

The GT algorithm is a now classical algorithm for decoding codes defined by an LDPC matrix, and has been shown to have remarkable BER performance [11]. In describing the algorithm, it is convenient to use a bipartite graph to represent the code and the decoding operations; see Fig. 1. On this graph, messages of bit probabilities are alternately passed between two types of nodes, bit nodes (represented by circles) and parity-check nodes (represented by squares). Connections between the two types of nodes are defined by a parity-check matrix. Each node as-

Manuscript received July 10, 2001; revised January 22, 2002. This work was supported by the Center for Magnetic Recording Research, University of California, San Diego. The material in this paper was presented in part at the IEEE Information Theory Workshop, Cairns, Australia, September 2001.

The authors are with the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093-0401 USA (e-mail: kurkoski@ucsd.edu; psiegel@ucsd.edu; jwolf@ucsd.edu).

Communicated by S. Shamai, Guest Editor.
Publisher Item Identifier S 0018-9448(02)04006-3.

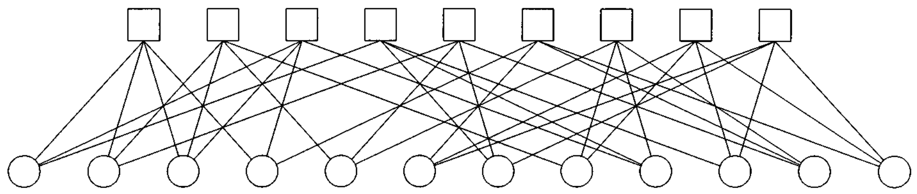

Fig. 1. Classical Tanner graph for a code with three information bits and nine parity bits.

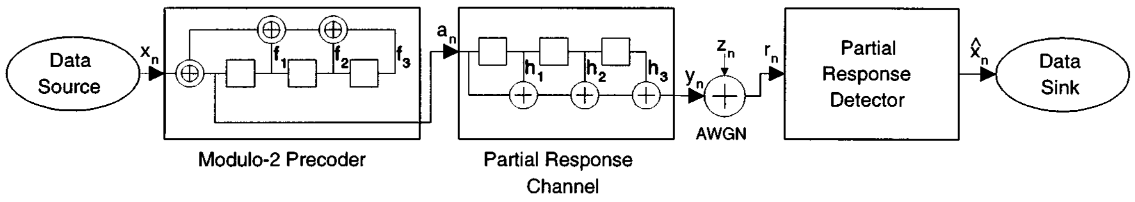

Fig. 2. Block diagram of a partial-response system, possibly precoded, with channel detector.

sumes statistical independence of its inputs and uses elementary rules of probability to calculate outputs, either combining the soft estimates (at the bit nodes) or using the parity-check constraint (at the parity-check nodes). The alternation of messages passing from bit nodes to parity-check nodes and back to bit nodes forms one iteration. When the GT algorithm attempts to decode a codeword, typically several iterations are performed until the decoder converges to a valid codeword. The GT algorithm is a type of message-passing algorithm. See [12] for a more detailed description of the algorithm.

An appealing practical aspect of the GT algorithm is that it consists of many small, independent decoding functions (i.e., the bit nodes and the parity-check nodes). In principle, hardware could implement circuits corresponding to these independent decoders that operate in parallel, potentially leading to a very-high-speed decoder. This aspect is particularly important in magnetic recording applications, where data rate requirements are high, and decoding delay must be low. Kschischang *et al.* proposed a parallel decoder architecture for turbo codes called "concurrent turbo decoding," which likewise could lead to a high-speed implementation [13].

An important theme of this paper is the design of a parallel message-passing detector for partial-response channels, conceived in the spirit of the GT algorithm, which we introduce in Section II. In Section III, we demonstrate that for this proposed detector, there are particular output sequences that cannot be uniquely decoded to a single input sequence. We call such sequences *ambiguous output sequences*, but show also that precoding can eliminate this ambiguity. We show by computer simulation that this algorithm, under appropriate constraints, performs as well as the BCJR algorithm for particular partial-response channels.

Previous proposals for applying LDPC codes to the magnetic recording channel have utilized two separate decoders: the BCJR algorithm for channel detection and the GT algorithm to decode an outer LDPC code. These two separate decoders communicate soft information to each other and operate in accordance with the turbo principle. However, the LDPC code's advantage of low decoding delay is seriously degraded when the serial BCJR algorithm (sliding-window variants, e.g., [14], notwithstanding), is used as the detector.

Accordingly, in Section IV, we investigate a joint detector/decoder for an LDPC-coded partial-response channel that can operate in a completely parallel manner, circumventing the high delay of the BCJR algorithm. We find that BER is lowest when a large number of "turbo" iterations are performed and a small number of channel detector and LDPC code iterations are performed, subject to the constraint that the total number of iterations is fixed. Section V is a concluding summary.

II. Parallel Message-Passing Algorithm for the Detection of Partial-Response Channels

We consider the following partial-response system. A data source generates an independent and identically distributed binary sequence $x(D) = x_0 + x_1 D + x_2 D^2 + \ldots$, $x_i \in \{0, 1\}$, with zeros and ones equally likely. This is passed through a recursive modulo-2 precoder, which has transfer function $1/f(D)$ where

$$f(D) = \sum_{i=0}^{\nu} f_i D^i \bmod 2, \qquad f_i \in \{0, 1\}$$

of maximum degree ν (we also use $\oplus$ to indicate modulo-2 addition). The output of the precoder and the input to the channel is $a(D)$. The high-density magnetic recording medium and readback process can be modeled as a partial-response channel with binary inputs and multilevel outputs. The partial-response channel transfer polynomial is given by $h(D) = \sum_{i=0}^{\nu} h_i D^i$, for a channel of degree ν, h_i real. The partial-response channel output is $y(D) = h(D)a(D)$. It is followed by AWGN $z(D)$ and the received sequence observed by the channel detector is $r(D) = h(D)a(D) + z(D)$. There are N output symbols, so the polynomials $y(D)$ and $r(D)$ have degree $N - 1$. We assume that the partial-response model begins in the zero state at time index 0 and is terminated at the zero state after time index $N - 1$, so the number of information bits is $N - \nu$. An example system with $\nu = 3$ is shown in Fig. 2.

In this paper, we consider both nonprecoded systems (i.e., $f(D) = 1$) and precoded systems (i.e., $f(D) \neq 1$). When there is no precoder present, $x(D) = a(D)$; in such cases, we shall use $x(D)$ to refer to the input to the partial-response channel. For systems that have precoders, the degree of the precoder will

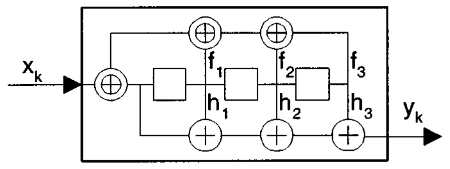

Fig. 3. The precoder and partial-response channel can be viewed as a single state machine.

be less than or equal to the degree of the channel transfer polynomial $h(D)$. For analysis purposes, both the precoder (which represents an actual circuit) and the partial-response channel model (which represents physical effects) can be combined into a single transfer function, see Fig. 3. Although this transfer function is a combination of real and modulo-2 operations, it can be expressed using a state-transition diagram. This state-transition model is used by the partial-response detector and the detector produces estimates for the precoder input $x(D)$ directly.

We describe two types of message-passing algorithms for a partial-response channel. Section II-A describes a detector where the messages passed are bit probabilities and Section II-B describes a detector where the messages passed are vectors of state probabilities.

A. Bit-Based Message Passing

To build a message-passing detector for the partial-response system, we made a graph similar to the bipartite graph of the GT algorithm. But, where the GT algorithm has a parity-check node, we replaced it with a "triangle" function node, which receives the noisy channel sample. A bit node, represented by a circle, is connected to a triangle function node if the partial-response transfer polynomial $h(D)$ indicates a direct dependence between that input and the corresponding channel output. The message passed between nodes is the bit probability $P[x_n]$. Both types of nodes generate bit probability messages using the "sum-product update rule" [15], that is, the output message from a node along edge e is based upon on all inputs to that node except the message from edge e. Let $\boldsymbol{x}_{p-\nu}^{p}$ represent the bits $(x_{p-\nu}, \ldots, x_p)$ and let $\boldsymbol{x}_{p-\nu}^{p\backslash n}$ represent those same bits except bit x_n for $p-\nu \le n \le p$. The received sample at node p is r_p. The message generated by the triangle function node p and passed to bit node n, called R_{pn} is

$$\begin{aligned} R_{pn} &= P[x_n = 1 | r_p] \\ &= \sum_{j_1 \cdots j_\nu,\, i} P\left[x_n = 1,\, \boldsymbol{x}_{p-\nu}^{p\backslash n} = (j_1 \cdots j_\nu),\, y_p = i | r_p\right] \\ &= \sum_{j_1 \cdots j_\nu,\, i} P\left[x_n = 1 | \boldsymbol{x}_{p-\nu}^{p\backslash n},\, y_p\right] \\ &\quad \cdot \frac{P[r_p|y_p] P\left[y_p | \boldsymbol{x}_{p-\nu}^{p\backslash n}\right] P\left[\boldsymbol{x}_{p-\nu}^{p\backslash n}\right]}{P[r_p]} \\ &= \sum_{j_1 \cdots j_\nu,\, i} P\left[x_n = 1 | \boldsymbol{x}_{p-\nu}^{p\backslash n},\, y_p\right] \\ &\quad \cdot \frac{P[r_p|y_p] P\left[y_p | \boldsymbol{x}_{p-\nu}^{p\backslash n}\right] \prod_{k=p-\nu}^{p\backslash n} Q_{kp}}{P[r_p]}. \end{aligned} \tag{1}$$

The sum is performed over $j_1 \cdots j_\nu$ from the input alphabet $\{0, 1\}$ and over i from the output alphabet. The term $P[x_n = 1 | \boldsymbol{x}_{p-\nu}^{p\backslash n}, y_p]$ will be either zero or one; $P[r_p|y_p]$ is computed using knowledge that the channel noise is Gaussian; $P[y_p|\boldsymbol{x}_{p-\nu}^{p\backslash n}]$ will be $1/2$ for all nonzero terms in the summation; $P[\boldsymbol{x}_{p-\nu}^{p\backslash n}]$ are the prior probabilities which are factored into individual probabilities $Q_{kp} = P[x_k]$ because of the independence of the source data. The term Q_{np} is the message generated by bit node n and passed to triangle function node p. This message is given by

$$\begin{aligned} Q_{np} &= \frac{\prod_{k=n}^{(n+\nu)\backslash p} P[x_n = 1 | r_k]}{\prod_{k=n}^{(n+\nu)\backslash p} P[x_n = 1 | r_k] + \prod_{k=n}^{(n+\nu)\backslash p} P[x_n = 0 | r_k]} \\ &= \frac{\prod_{k=n}^{(n+\nu)\backslash p} R_{kn}}{\prod_{k=n}^{(n+\nu)\backslash p} R_{kn} + \prod_{k=n}^{(n+\nu)\backslash p} (1 - R_{kn})}. \end{aligned} \tag{2}$$

In describing a message-passing schedule, as in [15], we assume there is a global clock which synchronizes the generation of new messages. A message-passing schedule on a graph is a predetermined specification of messages that are passed at each clock tick. For the proposed bit-based message-passing algorithm, the following schedule is applied. First, all the triangle function nodes simultaneously generate outputs R_{pn}, according to (1), using prior input from the bit nodes Q_{np}, if available (initialized to probability $1/2$ otherwise). Using that information, bit nodes simultaneously generate messages Q_{np} to send to triangle function nodes, according to (2). This cycle is repeated for a fixed number of iterations, and terminates with the bit nodes generating the final output $P[x_n]$. See Figs. 4 and 5 for bit message-passing graphs corresponding to the dicode ($h(D) = 1 - D$) and EPR4 ($h(D) = 1 + D - D^2 - D^3$) partial-response channels.

The message-passing graph constructed for the dicode channel (Fig. 4) has no cycles, and cycle-free graphs are necessary for exact solutions [16]. When the number of iterations on the dicode graph is equal to N, the number of bit nodes in the graph, the algorithm will produce the same result as the BCJR algorithm. However, in computer simulations, detection of the dicode channel using this graph produced performance similar to the BCJR algorithm for far fewer iterations than N (N is roughly 4500 for magnetic recording systems). In particular, 16 iterations produced results that were indistinguishable from BCJR performance at BER above 10^{-7}, see Fig. 6. An alternative explanation is that because the dicode channel model has two states, the bit message contains enough information to describe the state, and thus is emulating the BCJR algorithm.

However, the message-passing graph for the EPR4 channel (Fig. 5) has many short cycles, and so we cannot expect this algorithm to produce an exact solution. This was verified by com-

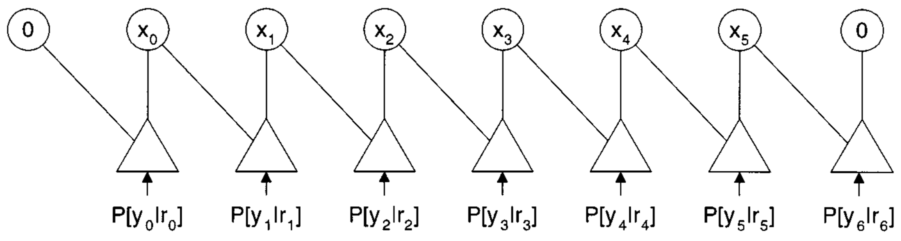

Fig. 4. A bit message-passing diagram for the dicode partial-response channel $(h(D) = 1 - D)$. The message passed is $P[x_n = 1]$.

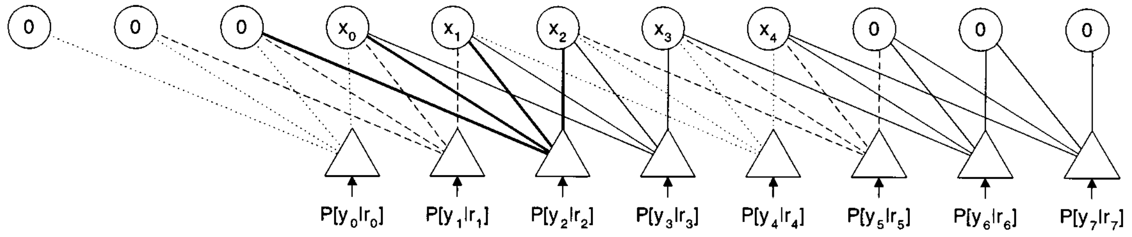

Fig. 5. A bit message-passing diagram for the EPR4 partial-response channel $(h(D) = 1 + D - D^2 - D^3)$. The message passed is $P[x_n = 1]$. Dashed lines added to enhance contrast.

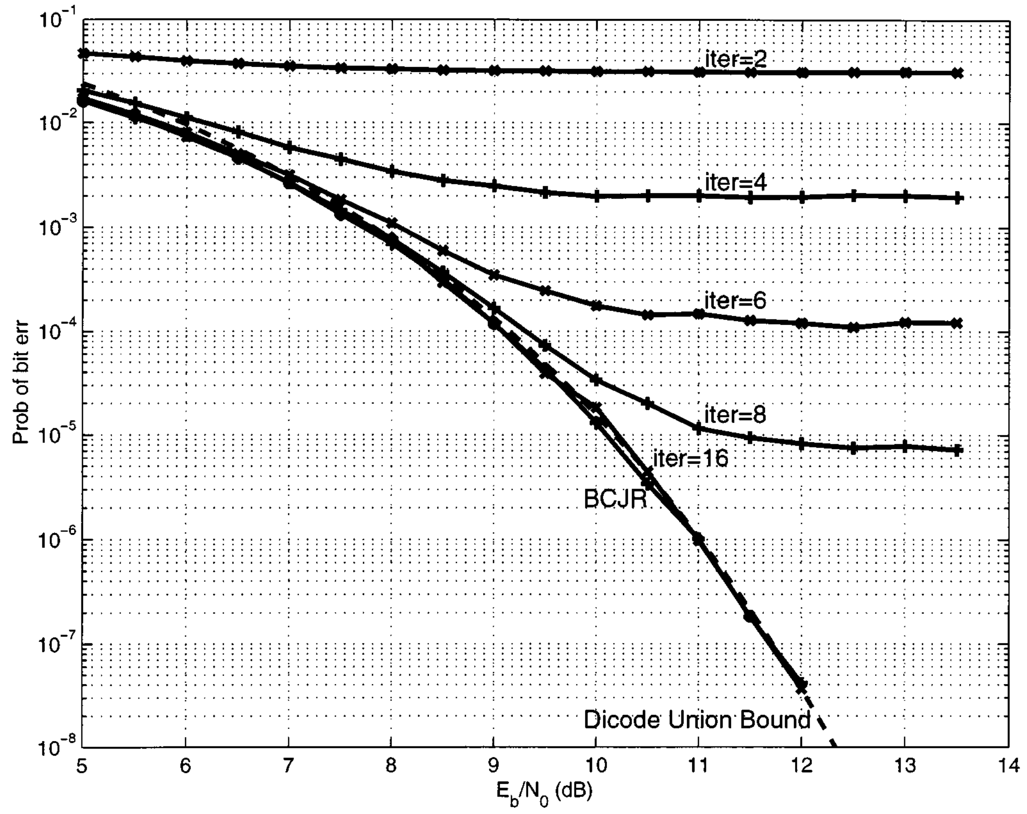

Fig. 6. BER for bit-based message passing on the dicode channel without precoding.

puter simulation, see Fig. 7. For increasing iterations, BER performance improved very slowly, even for a very large number of iterations and very high SNR.

For the BER simulation results presented in this paper, comparisons are made with the appropriate dicode or EPR4 truncated union bound [17].

B. State-Based Message Passing

The key to achieving BCJR-like performance for channels with more than two states is to pass state information, not bit information. In the proposed state-based algorithm, the messages passed are the state probabilities, represented in a vector; this is in contrast to the algorithm of the previous subsection, where

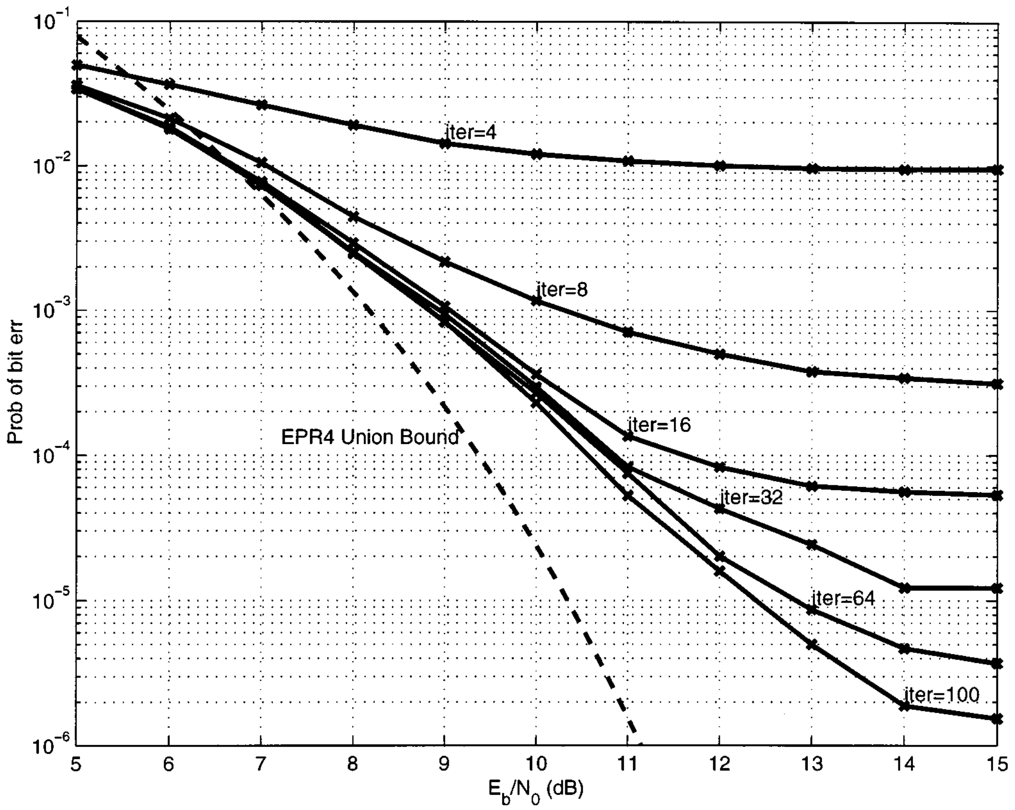

Fig. 7. BER for bit-based message passing on the EPR4 channel without precoding.

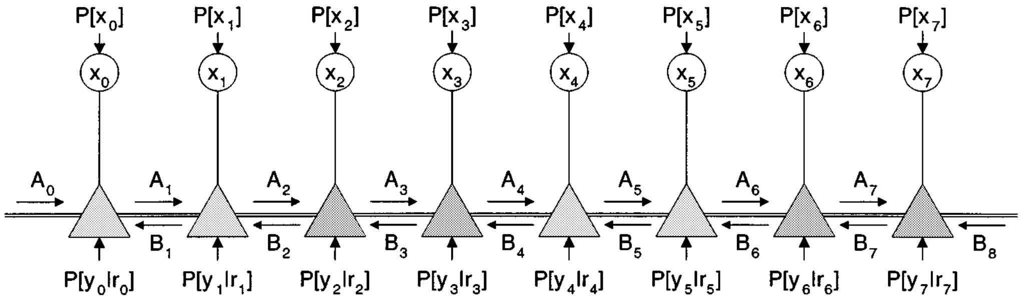
P[x0] P[x1] P[x2] P[x3] P[x4] P[x5] P[x6] P[x7]
x0 x1 x2 x3 x4 x5 x6 x7
A0 A1 A2 A3 A4 A5 A6 A7
B1 B2 B3 B4 B5 B6 B7 B8
P[y0|r0] P[y1|r1] P[y2|r2] P[y3|r3] P[y4|r4] P[y5|r5] P[y6|r6] P[y7|r7]

Fig. 8. A state message-passing diagram for a general partial-response channel. The message passed between the triangle function nodes is the state distribution vector $P[s_n = m]$.

the message is a bit probability which can be represented by a single number. Fig. 8 shows a state message-passing graph for the general partial-response channel. In this graph, the triangle is a state processing node which has four edges on which it sends and receives three types of messages. One type of message is a vector of state probabilities that are communicated with its two neighboring triangle nodes. The other two types of messages are probabilities corresponding to input symbols and output symbols of the channel, $P[x_n]$ and $P[y_n|r_n]$, respectively. In an implementation, the dimension-two vector $P[x_n]$ can be readily represented as a single number, but because the partial-response channel has three or more output symbols, the message $P[y_n|r_n]$ must be represented as a true vector. The message-passing schedule is still parallel. Note that the graph has no loops. This graph is similar to a Wiberg graph for trellises [18], but we have omitted state nodes because they connect two adjacent triangle function nodes and only act as a message relay. A double line is drawn to indicate where state messages are passed between nodes.

The proposed state-based partial-response channel detector algorithm follows. Let s_n be the state of the partial-response channel with $M = 2^\nu$ states, at time n, $n = 0, \ldots, N$. The quantities $A_n(m)$ and $B_n(m)$ are the state messages that are passed right and left, respectively, $m = 0, \ldots, M-1$. These state messages are produced simultaneously by each triangle function node. Sums are performed over edges in the partial-response trellis, as in [19]. For a partial-response trellis, there are $2M$ edges per trellis section, and the trellis is time-invariant. For each trellis edge e, the starting state is $s^S(e)$, the ending state is $s^E(e)$, the associated input label is $x(e)$ and the associated output label is $y(e)$. Fig. 9 shows the correspondence between a node of the message-passing graph and a trellis section.

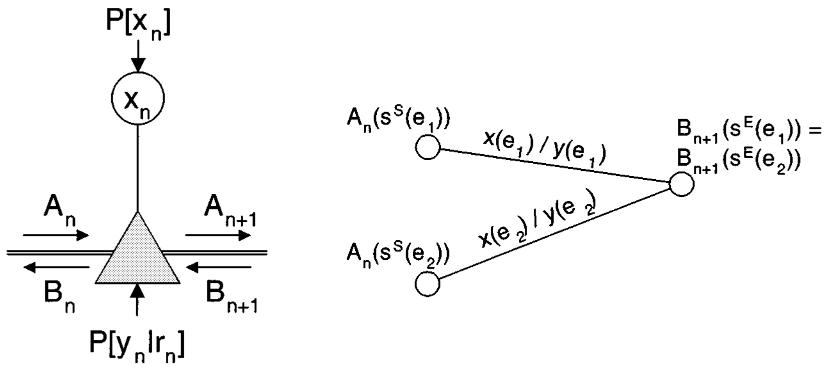

Fig. 9. A single section of the message-passing graph for the partial-response detector, and a corresponding trellis fragment.

Algorithm:

1. Initialize:

$$A_0(m) = B_N(m) = \begin{cases} 1, & m = 0 \\ 0, & m \neq 0 \end{cases}$$

$$A_n(m) = B_n(m) = \frac{1}{M}, \qquad \forall m;\ n \neq 0, N.$$

2. For each node $n = 0, \ldots, N-1$, simultaneously generate the outputs $A_{n+1}(m)$, $B_n(m)$, $m = 0, \ldots, M-1$

$$A_{n+1}(m) = H_{n+1}^A \sum_{e:\, s^E(e)=m} A_n\big(s^S(e)\big) P[x(e)] P[y(e)|r_n]$$

$$B_n(m) = H_n^B \sum_{e:\, s^S(e)=m} B_{n+1}\big(s^E(e)\big) P[x(e)] P[y(e)|r_n].$$

   (Node 0 does not generate B_0 and node $N-1$ does not generate A_N.)

3. Repeat step 2 T times
4. For each node $n = 0, \ldots, N-1$, simultaneously generate $P[x_n]$

$$P[x_n] = H_n \sum_{e:\, x(e)=x_n} A_n\left(s^S(e)\right) P[y(e)|r_n] B_{n+1}\left(s^E(e)\right).$$

In steps 2 and 4, H_{n+1}^A, H_n^B, H_n are proportionality constants chosen so that probabilities sum to one.

For a fixed number of iterations T, the output symbol estimates $P[y_n = y(e)|r_n]$ and the input symbol estimates $P[x_n = x(e)]$ that contribute to the decoding of a given bit are all in a window of length $W = 2T+1$, centered on that bit. Given only the input information in the window, this algorithm produces the *a posteriori* probability (APP) estimate for that bit. We call this the window-APP algorithm.

The algorithm we have described is a parallel schedule applied to a message-passing graph. As has been observed by Wiberg [18, Sec. 3.2], the BCJR algorithm can be represented on a message-passing graph by using a serial schedule. Starting with an initial state distribution for A_0 (usually given as $A_0(0) = 1$, $A_0(m) = 0$, $m \neq 0$), we can compute A_1 at node 0. Once A_1 is available, compute A_2 at node 1; likewise, for $A_3, A_4, \ldots, A_{N-1}$. This is the recursive computation of the state distributions in the forward direction, often called the "alpha" or forward state metrics. Similarly, there are "beta" or backward state metrics. Thus, the two different algorithms can be described by applying appropriate schedules to a single graph. Further, a variety of algorithms can be described by applying different schedules to this graph. In Section III, only the parallel schedule will be used.

III. Window-APP Algorithm Properties

When the number of iterations T for the proposed detector algorithm is greater than or equal to the block length N, the window size for all bits encompasses the entire block, and the algorithm produces the same APP outputs that the BCJR algorithm produces. However, here we investigate the behavior of the window-APP detector for values of T much smaller than the block length.

For partial-response channels that are not precoded, there exist multiple distinct input sequences starting in distinct states that produce identical output sequences within a window of size W. Such output sequences are called *ambiguous output sequences*. For such sequences, even in the absence of noise, this detector cannot distinguish which input sequence generated the observed output sequence and thus the detector has a high probability of failure. The likelihood of such a decoding failure depends upon the probability that initial states and input sequences that lead to ambiguous output sequences will occur. In turn, this probability is a decreasing function of the window size. In the following analysis, we predict the probability of bit error for such a detector, $P_{be}^{\min}(T)$, in the absence of noise. $P_{be}^{\min}(T)$ is the probability that there was a bit error in the center of the window.

Whereas the probability of bit error for conventional detectors approaches zero for increasing SNR, the probability of bit error for this detector will asymptotically approach $P_{be}^{\min}(T)$. Hence, this quantity is the minimum probability of bit error that the detector will achieve.

For the dicode channel there is only one ambiguous output sequence in a window of length W, the all-zero output sequence. The dicode channel has two states, $m \in \{0, 1\}$. For example with $W = 5$, the sequence $\boldsymbol{y}_n^{n+4} = (0, 0, 0, 0, 0)$

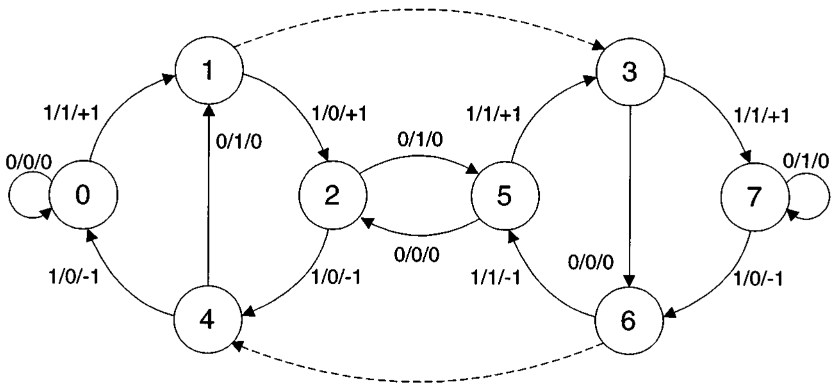

Fig. 10. State diagram for the EPR4 partial-response channel, with labels "precoded input/nonprecoded input/output." Precoder is $f(D) = 1 \oplus D \oplus D^2 \oplus D^3$. Transitions generating outputs -2 and $+2$ are shown with a dashed line, but not labeled.

TABLE I
FOR EPR4, SOME EXAMPLE AMBIGUOUS INPUT SEQUENCES FOR WINDOW SIZE $W = 5$. THE OUTPUT SYMBOLS ARE $y_n \in \{-2, -1, 0, +1, +2\}$

(initial state,input)	state sequence	output
(0,00000)	000000	00000
(2,10101)	252525	00000
(5,01010)	525252	00000
(7,11111)	777777	00000
(0,10000)	012400	+1 +1 −1 −1 0
(5,11010)	537652	+1 +1 −1 −1 0
(4,00010)	400012	−1 0 0 +1 +1
(6,10111)	652637	−1 0 0 +1 +1

could have been generated by either the initial state $m = 0$, and $\boldsymbol{x}_n^{n+4} = (0, 0, 0, 0, 0)$ or by the initial state $m = 1$ and $\boldsymbol{x}_n^{n+4} = (1, 1, 1, 1, 1)$. The detector will arbitrarily choose one of the two possible input sequences and if it chooses the wrong one, all the input bits will be incorrect. This results in a probability of bit error of $1/2$, given that an ambiguous output sequence was received. The probability that W output symbols will all be zero is the probability that $W + 1$ input symbols are identical, that is, $2 \cdot (1/2)^{W+1} = (1/2)^W$. If this sequence is received, the probability of bit error is $1/2$, and so

$$\begin{aligned} P_{be}^{\min}(T)[\text{dicode}] &= P[\text{bit error} \cap AS_{\text{dicode}}] \\ &= P[\text{bit error}|AS_{\text{dicode}}]P[AS_{\text{dicode}}] \\ &= (1/2)^{W+1} = (1/2)^{2T+2} \end{aligned}$$

where AS_{dicode} is the event that the ambiguous output sequence was received.

For the EPR4 channel, there are numerous ambiguous output sequences. The state diagram for EPR4 is given in Fig. 10, and some example ambiguous output sequences are given in Table I.

The EPR4 ambiguous output sequences of length $W > 3$ can be represented in general by

$$\begin{aligned} &[0] + 1 + 1\ [0\ 0] - 1 - 1\ [0\ 0] + 1 + 1\ [0\ 0] \cdots \text{ or} \\ &\qquad + 1\ [0\ 0] - 1 - 1\ [0\ 0] + 1 + 1\ [0\ 0] \cdots \end{aligned} \tag{3}$$

where $[\cdot]$ means that the contents may be omitted, used as is, or repeated an arbitrary number of times. Sequences that differ by only a change of sign throughout are not listed. The symbols $\{-1, 0, +1\}$ must occur as pairs, except at the beginning and end of the sequence, where a single symbol is permitted. A derivation of this list of ambiguous sequences is given in the Appendix.

For EPR4, there are two types of ambiguous output sequences. Let AS_z be the event that the all-zeros output, which is an ambiguous output sequence, was received by the detector. Let AS_{nz} be the event that any other ambiguous output sequence was received. For the EPR4 channel, there are four input sequences that correspond to AS_z: $(0, 0, 0, 0 \cdots)$, $(0, 1, 0, 1, \ldots)$, $(1, 0, 1, 0, \ldots)$, and $(1, 1, 1, 1, \ldots)$. When the event AS_z occurs (i.e., the all-zeros output is received), the detector will randomly choose one of the four possible input sequences, and the probability of bit error will be $1/2$. There are two input sequences for each output sequence corresponding to AS_{nz}. When the event AS_{nz} occurs (i.e., any ambiguous output sequence except the all-zeros output is received), the detector will randomly choose one of the two possible input sequences and the probability of bit error will be $1/4$, as shown in the Appendix.

When the length W is odd, the number of EPR4 ambiguous output sequences described by (3) is $2^{\frac{W+5}{2}} - 3 = 2^{T+3} - 3$. These are generated by $2^{T+4} - 4$ distinct input sequences, four of which generate the all-zeros ambiguous output sequence, as shown in the Appendix. (When the length W is even, the number of ambiguous output sequences is $3 \cdot 2^{\frac{W}{2}+1} - 3$. These are generated by $3 \cdot 2^{\frac{W}{2}+2} - 4$ distinct input sequences.) There are 2^3 possible starting states at the beginning of the window, and 2^{2T+1} possible input bit patterns that could be received in the

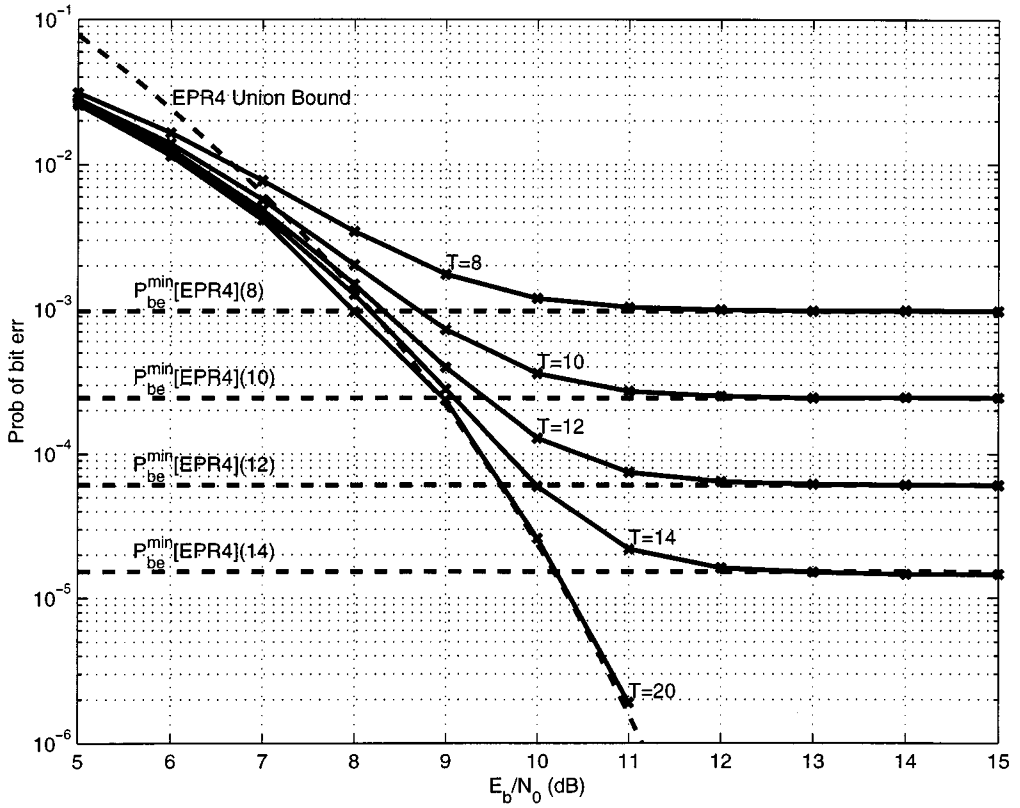

Fig. 11. BER for state-based message passing on the nonprecoded EPR4 partial-response channel, for various iterations.

window. Since each starting state and each input bit pattern is equally likely

$$P[AS_z]=4/(2^3 2^{2T+1}) \text{ and } P[AS_{nz}]=(2^{T+4}-8)/(2^3 2^{2T+1}).$$

Letting E be the event that a bit error occurs in the center of the window, and because all errors are due to ambiguous sequences

$$\begin{aligned} P_{be}^{\min}(T)[\text{EPR4}] &= P[E|AS_z]\,P[AS_z] + P[E|AS_{nz}]\,P[AS_{nz}] \\ &= \frac{1}{2}\cdot\frac{4}{2^3 2^{2T+1}} + \frac{1}{4}\cdot\frac{2^{T+4}-8}{2^3 2^{2T+1}} \\ &= \frac{1}{2^{T+2}}. \end{aligned} \tag{4}$$

This proposed EPR4 detector was simulated on a computer with AWGN noise for various values of T; see Fig. 11. For increasing SNR, the probability of bit error tends toward the predicted value $P_{be}^{\min}(T)$[EPR4]. The proposed algorithm achieves essentially the same BER as the BCJR algorithm, below some SNR threshold which is a function of T. This threshold can be made arbitrarily high as T approaches N, in which case the two algorithms produce identical results.

For example, for $T = 20$ and $E_b/N_0 = 11$ dB, the proposed algorithm achieves a probability of bit error of $2 \cdot 10^{-6}$, essentially indistinguishable from BCJR performance for the computer simulation shown in Fig. 11. The proposed algorithm can produce BCJR-like performance with only $T/N \ll 1$ times the delay of the BCJR algorithm. However, this algorithm has T times the computational complexity of the BCJR algorithm.

For both the dicode and EPR4 channel, when a precoder matched to the partial-response channel is used (i.e., $f(D) = h(D) \bmod 2$), the combined precoder/channel response has the property that any output uniquely identifies the corresponding input in the absence of noise. Although ambiguous output sequences still exist, they are ambiguous only in the starting state, not the input sequence. For example, for the precoded dicode channel $(f(D) = (1 \oplus D))$, the output $\boldsymbol{y}_n^{n+4} = (0, 0, 0, 0, 0)$ can be generated by either the initial state $m = 0$ and input sequence $\boldsymbol{x}_n^{n+4} = (0, 0, 0, 0, 0)$, or by the initial state $m = 1$ and input sequence $\boldsymbol{x}_n^{n+4} = (0, 0, 0, 0, 0)$. Although the two state sequences are distinct, this ambiguity will not cause any bit errors.

For EPR4, the precoder matched to the channel, $f(D) = 1 \oplus D \oplus D^2 \oplus D^3$, as well as the precoder $f(D) = 1 \oplus D^2$ eliminate input–output ambiguity. In the Appendix, we show that these are the only two precoders of degree less than or equal to three that have this property. Computer simulations for EPR4 precoded with $f(D) = 1 \oplus D \oplus D^2 \oplus D^3$ are shown in Fig. 12. In this figure, the truncated union bounds for both the precoded and nonprecoded channel are shown for comparison. Regardless of the number of iterations T, the probability of bit error decreases for increasing SNR, a desirable property, and is in marked contrast to the nonprecoded case.

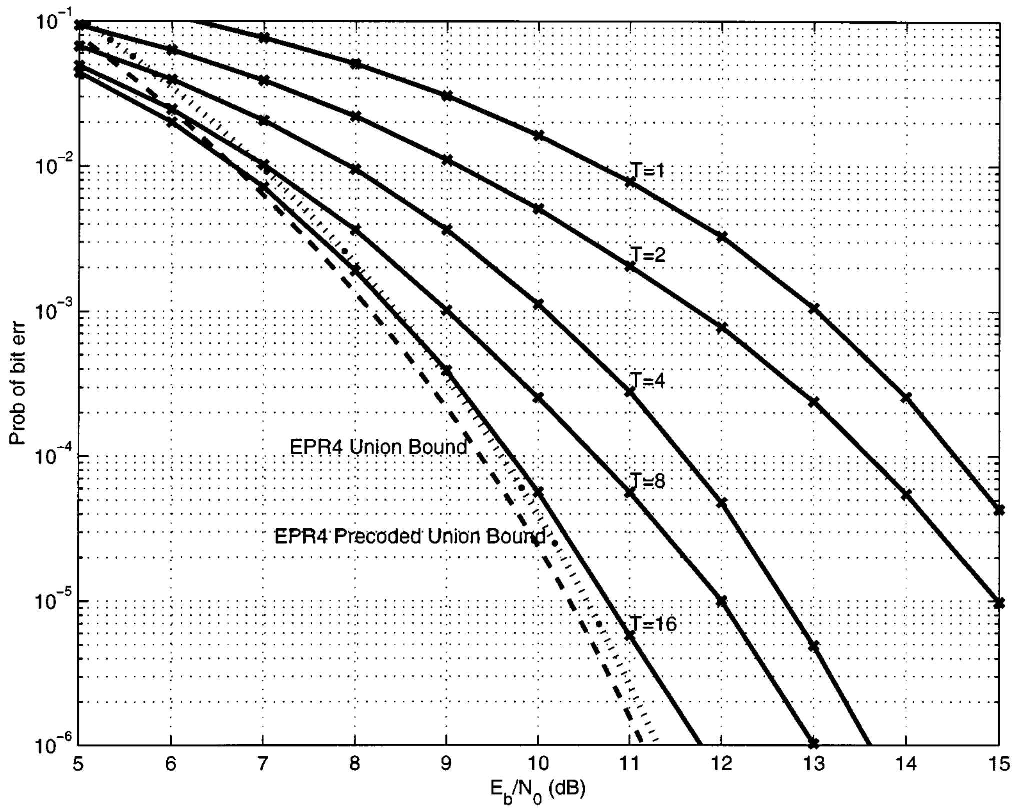

Fig. 12. BER for state-based message passing on the EPR4 partial-response channel precoded with $1 \oplus D \oplus D^2 \oplus D^3$, for various iterations.

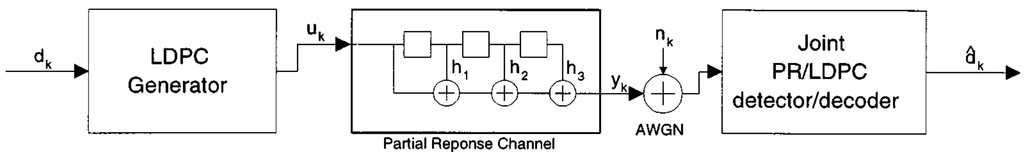

Fig. 13. Block diagram of the proposed LDPC-coded partial-response system.

IV. Joint Decoding of LDPC and Partial-Response Channels

Recently, there have been proposals to use an LDPC code on the partial-response channel [20], [21], with an iterative decoder consisting of a BCJR-type detector for the partial-response channel and a GT-type decoder for the LDPC code. However, the BCJR-type algorithms are slower, and the channel decoder proposed in Section II is well-matched for use with the GT algorithm as both are inherently parallel algorithms. The block diagram for such a partial-response system is shown in Fig. 13.

In [22], Garcia-Frias considers decoding of LDPC codes over finite-state Markov channels and specifically investigates the Gilbert–Elliot channel model. Here, we consider a joint message-passing decoder matched to both an LDPC code and the channel, when the model is a partial-response channel. We assume that the channel is not precoded. A graph for such a decoder is shown in Fig. 14, and we apply the following schedule to that graph. First, the proposed decoder operates for T iterations on the channel samples, then it passes soft estimates of the input symbols to the GT decoder, matched to the LDPC code. Then, the GT decoder operates for S iterations, and the resulting soft estimates of the input symbols are passed back to the channel decoder. This forms one "turbo" iteration, and is repeated U times.

BER performance generally improves with increasing iterations, and we are interested in the relationship between the parameters T, S, and U that would achieve the best performance. To do this, we set $(T+S)U$ equal to a constant (the total number of iterations used in detection and decoding), and found the values of T, S, and U that yielded the lowest BER, via computer simulation. In doing this, we assume that the "costs," in the sense of delay, for the channel detector and GT algorithm are the same, and that one channel iteration can be traded for one decoder iteration, and *vice versa* with no penalty. It should be noted, however, that these costs are dependent on the implementation, and the validity of this assumption will depend upon system parameters such as the rate of the LDPC code and the number of states in the partial-response channel. We assume that no processing begins until all the channel samples have been received and are available to the detector (reference [14] describes a suboptimal version of BCJR that produces final outputs before all the data has been received). Further, we assume that the GT algorithm begins processing only after the channel algorithm is complete (and *vice versa*).

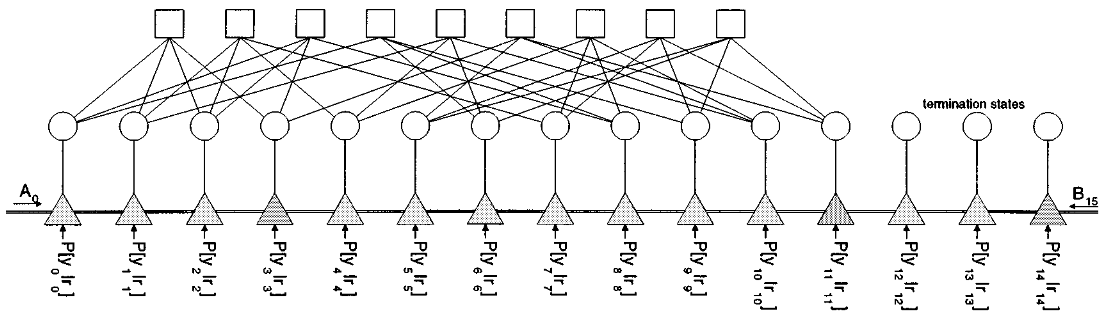

Fig. 14. Joint message-passing detector/decoder for the partial-response channel and LDPC code (termination states shown).

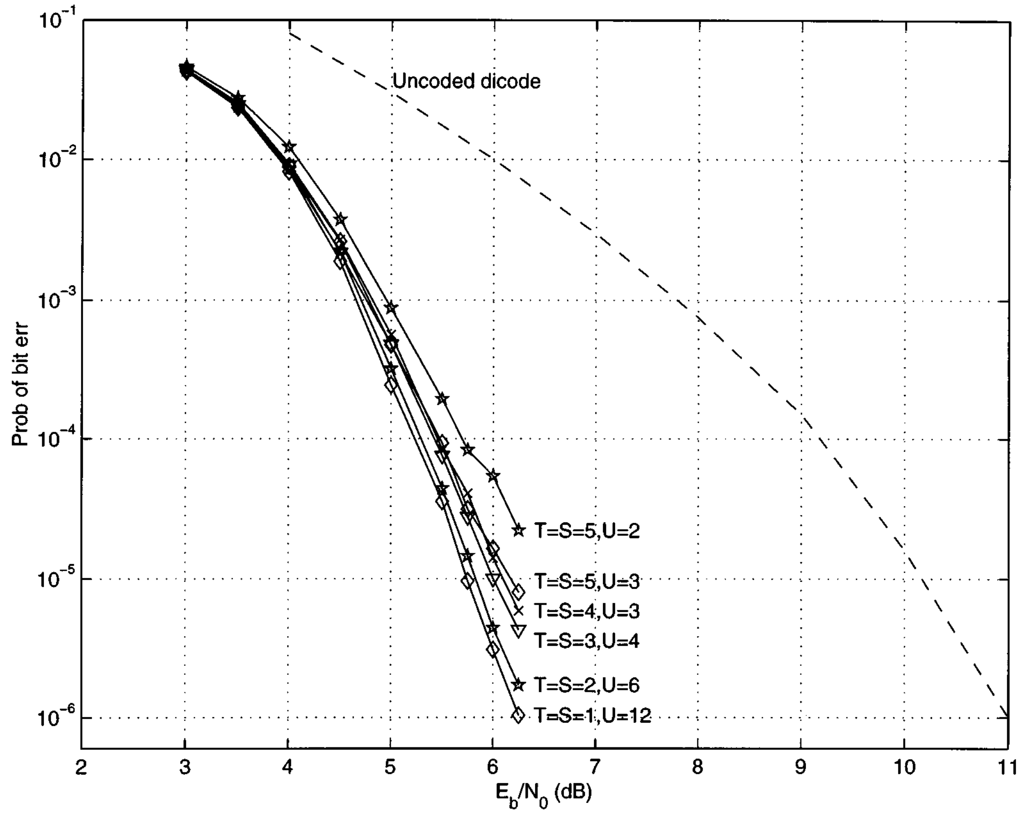

Fig. 15. Simulation results for joint LDPC/dicode message passing decoder (using bit-message passing for the dicode channel).

A joint LDPC/partial-response message-passing system was simulated where the partial-response channel was the dicode model using bit-based message passing, and the LDPC code was a regular (column weight 3), rate $7/8$ LDPC code of block length 495 [23]. Simulation results are shown in Fig. 15. The iteration parameters T, S, U were chosen so that $T = S$ and $(T+S)U = 24$ (except in the case of $T = S = 5$, $U = 2$, in which case $(T+S)U = 20$). Simulation results show that the lowest probability of bit error is obtained when $T = S = 1$ and $U = 12$. In other simulations, we found that by setting $T = 1$ and changing S, or by setting $S = 1$ and changing T, the lowest probability of error was still achieved when $T = S = 1$, subject to a fixed number of iterations.

We also considered how a joint LDPC/partial-response message-passing system would perform when the channel is EPR4. In this case, state-based message passing was used. As with the dicode experiment, we restricted the schedule parameters such that $(T+S)U = 24$. In this case, we found that the optimal schedule, subject to the constraint, was $T = 2$, $S = 1$, $U = 8$; see Fig. 16.

The computational complexity of the BCJR algorithm is equivalent to the proposed algorithm with $T = 1$, and if we use this assumption to fix the total computation of the BCJR-GT detector/decoder system at the same value of $(1+S)U$, then we can compare a BCJR-GT system with the proposed fully parallel system.

The choice between the proposed algorithm and a BCJR-GT decoded system represents a tradeoff between decoding delay and performance. On one hand, the proposed message-passing detector/decoder has low latency compared to a conventional BCJR-GT decoder. In principle, the delay of the parallel message-passing detector algorithm is T, and that of the GT algorithm is S, so the overall decoding delay of the proposed system is $(T+S)U$, which is 24 in the case of the system simulated

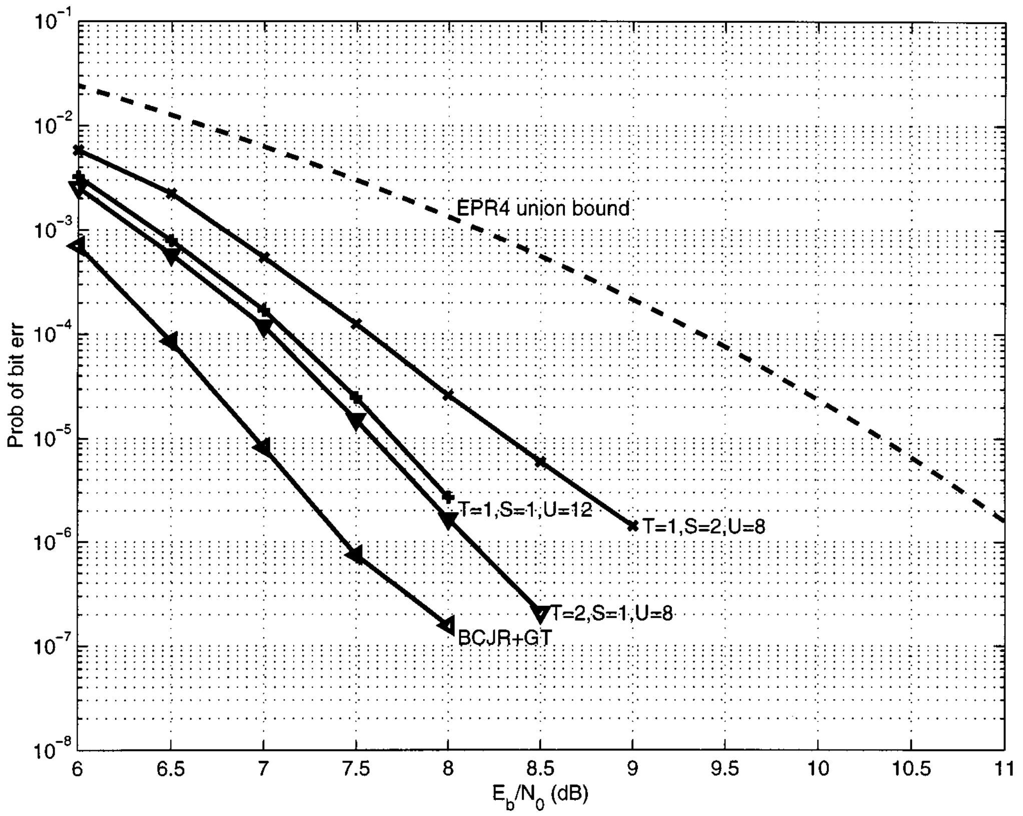

Fig. 16. Simulation results for joint LDPC/EPR4 message passing decoder, passing state information.

in Fig. 16. In contrast, the BCJR algorithm has a delay of N, so the overall decoding delay for a BCJR-GT system is $(N+1)U$, which is 5952 for the system that we simulated. These numbers are very rough estimates, and we use them only to emphasize the much lower decoding delay of the proposed system. On the other hand, the simulation results show that the BCJR-GT EPR4 system has $\approx$ 0.6 dB better performance at 10^{-6} probability of bit error.

V. Conclusion

We have presented two types of detectors for the partial-response channel that are parallel, message-passing algorithms. One is a bit-based message-passing algorithm that was constructed in the spirit of the GT algorithm. It is effective at detecting the dicode channel, but when applied to the EPR4 channel, the observed BER performance was poor.

The other detector we presented was a state-based, parallel message-passing algorithm. This state-based algorithm is a window-APP algorithm for general partial-response channels, in contrast to the BCJR algorithm, which is serial in nature. However, both algorithms can be described on a single message-passing graph, differentiated only by the schedule that is applied to that graph. The parallel algorithm will produce results different from the BCJR algorithm when $T < N$. We showed by analysis, and confirmed by simulation, that the BER for the state-based message-passing decoder can never be lower than a fixed constant, and we calculated this constant for the dicode and EPR4 channels. Fortunately, this lower bound can be eliminated using the proper precoder.

For decoding LDPC codes over partial-response channels, there is a joint decoder which combines the GT algorithm with the proposed channel detector, both parallel algorithms. As such, this algorithm has low decoding delay, as compared to a conventional BCJR-GT decoder. For the dicode partial-response channel with bit-based message passing, the parameters $T = 1$, $S = 1$, $U = 12$ achieve the lowest probability of bit error when the joint decoder is used and the total number of iterations is fixed at 24. When the same restrictions are applied to the EPR4 partial-response channel using state-based message passing, we find that the parameters $T = 2$, $S = 1$, $U = 8$ achieve the lowest probability of bit error. The trend is that if the total number of iterations is constrained to be a fixed number, then the lowest BER is achieved when the constituent decoder iterations $(T,\, S)$ are small, and the number of turbo iterations (U) is large.

Directions for future work include the application of density evolution techniques to this joint decoder to determine performance thresholds and to analyze iteration schedules, as well as the investigation of the effect of precoded channels in the context of an LDPC code.

Appendix

In this appendix, we consider the system of Fig. 2, i.e., a partial-response system without an outer LDPC code. We generalize the channel slightly and assume that the input and output are bi-infinite, that is,

$$x(D) = \sum_{i=-\infty}^{\infty} x_i D^i, \qquad y(D) = \sum_{i=-\infty}^{\infty} y_i D^i.$$

Let $\boldsymbol{y}_1^W$ be an ambiguous output sequence of length W. We consider two distinct input sequences $\boldsymbol{x}_{-2}^W$ and $\boldsymbol{x}'^W_{-2}$ of length $W+3$ (for the EPR4 channel) that generate the same output sequence $\boldsymbol{y}_1^W$. That is, if $y(D)=h(D)x(D)$ and $y'(D)=h(D)x'(D)$, then $\boldsymbol{y}_1^W=\boldsymbol{y}'^W_1$, an ambiguous output sequence. We assume W is odd.

A. Ambiguous Output Sequences for EPR4

In this subsection, we list the ambiguous sequences for the nonprecoded EPR4 partial-response channel. A null error sequence $\varepsilon_x(D)=x(D)-x'(D)$ is a bi-infinite sequence with the property that $\|h(D)\varepsilon_x(D)\|^2=0$. It was shown in [24] for a broad class of channels, including the EPR4 channel, that if the input error sequence $\varepsilon_x(D)$ is one of the following null error sequences: $(0)^\infty$, $(+)^\infty$, $(-)^\infty$, $(+-)^\infty$, $(-+)^\infty$, $(+0)^\infty$, $(0+)^\infty$, $(-0)^\infty$, or $(0-)^\infty$, then the distance between the corresponding outputs $y(D)=h(D)x(D)$ and $y'(D)=h(D)x'(D)$ will be zero. Here, the notation $(\cdot)^\infty$ represents an infinite periodic sequence with a pattern given by the contents of the brackets. The symbols "$+$" and "$-$" represent the input errors $+1$ and -1, respectively.

If the difference between two distinct sequences $\boldsymbol{x}_{-2}^W$ and $\boldsymbol{x}'^W_{-2}$ is a truncated version of one of the listed null error sequences, then the distance between $\boldsymbol{y}_1^W$ and $\boldsymbol{y}'^W_1$ is zero. In other words, the two output sequences will be identical, and, therefore, $\boldsymbol{y}_1^W$ will be an ambiguous output sequence. Using the list of possible null error sequences, we generate a list of possible ambiguous output sequences for EPR4.

Case A: Consider

$$\boldsymbol{x}_{-2}^W=(1,0,1,0,1,0,\ldots)$$

and

$$\boldsymbol{x}'^W_{-2}=(0,1,0,1,0,1,\ldots)$$

which correspond to a truncated version of the null error sequence $\varepsilon_x=(+-)^\infty$. Then, $\boldsymbol{y}_1^W=\boldsymbol{y}'^W_1=(0,0,0,\ldots)$, the all-zeros sequence. Input sequences $\boldsymbol{x}_{-2}^W$ and $\boldsymbol{x}'^W_{-2}$ that correspond to truncated null error sequences of $\varepsilon_x=(+)^\infty$, $\varepsilon_x=(-)^\infty$, and $\varepsilon_x=(-+)^\infty$ also produce the all-zeros sequence. In all, there are four input sequences that generate the all-zeros ambiguous output sequence.

Case B: Consider:

$$\boldsymbol{x}_{-2}^W=(1,x_{-1},1,x_1,1,x_3,1,\ldots) \tag{5}$$

and

$$\boldsymbol{x}'^W_{-2}=(0,x_{-1},0,x_1,0,x_3,0,\ldots) \tag{6}$$

which correspond to a truncated version of the null error sequence $\varepsilon_x=(+0)^\infty$. Both input sequences will generate the output

$$\boldsymbol{y}_1^W=((x_1-x_{-1}),(x_1-x_{-1}),(x_3-x_1),(x_3-x_1),\ldots).$$

If $x_{-1}=0$, then the possible output sequences are described by

$$\boldsymbol{y}_1^W=[0\;0]+1+1\;[0\;0]-1-1\cdots \tag{7}$$

where $[\cdot]$ means that the contents may be omitted, used as is, or repeated an arbitrary number of times. If instead $x_{-1}=1$, then the possible output sequences are described by (7) with a sign change throughout.

There are $2^{\frac{W+3}{2}}$ distinct input sequences described by (5). However, because both

$$\boldsymbol{x}_{-2}^W=(1,0,1,0,\ldots)\quad\text{and}\quad\boldsymbol{x}_{-2}^W=(1,1,1,1,\ldots)$$

generate the all-zeros sequence, the number of distinct ambiguous output sequences generated by (5) is $2^{\frac{W+3}{2}}-1$. There are $2^{\frac{W+3}{2}}$ distinct input sequences described by (6), so the number of distinct input sequences for Case B is $2^{\frac{W+5}{2}}$.

Case C: Consider

$$\boldsymbol{x}_{-2}^W=(x_{-2},1,x_0,1,x_2,1,x_4,\ldots) \tag{8}$$

and

$$\boldsymbol{x}'^W_{-2}=(x_{-2},0,x_0,0,x_2,0,x_4,\ldots) \tag{9}$$

which correspond to a truncated version of the null error sequence $\varepsilon_x=(0+)^\infty$. Both input sequences will generate the output

$$\boldsymbol{y}_1^W=((x_0-x_{-2}),(x_2-x_0),(x_2-x_0),\ldots).$$

If $x_{-2}=0$ and $x_0=0$, then the possible output sequences are described by

$$\boldsymbol{y}_1^W=0\;[0\;0]+1+1\;[0\;0]-1-1\cdots. \tag{10}$$

If $x_{-2}=0$ and $x_0=1$, then the possible output sequences are described by

$$\boldsymbol{y}_1^W=+1\;[0\;0]-1-1\;[0\;0]+1+1\cdots. \tag{11}$$

If $x_{-2}=1$, then the possible output sequences are described by (10) or (11) with a sign change throughout. The sequences

$$\boldsymbol{y}_1^W=[0]+1+1\;[0\;0]-1-1\cdots \tag{12}$$

and their counterparts with a sign change throughout, are a convenient way to represent (7) and (10). Together, (11) and (12) are a convenient way to list the possible ambiguous output sequence of the EPR4 channel model, as in (3).

There are $2^{\frac{W+3}{2}}$ distinct input sequences described by (8). However, because both

$$\boldsymbol{x}_{-2}^W=(0,1,0,1,\ldots)\quad\text{and}\quad\boldsymbol{x}_{-2}^W=(1,1,1,1,\ldots)$$

generate the all-zeros sequence, the number of distinct ambiguous output sequences generated by (8) is $2^{\frac{W+3}{2}}-1$. There are $2^{\frac{W+3}{2}}$ distinct input sequences described by (9), so the number of distinct input sequences for Case C is $2^{\frac{W+5}{2}}$.

Case D: The input sequences that correspond to truncated versions of the null error sequences $\varepsilon_x=(-0)^\infty$ and $\varepsilon_x=(0-)^\infty$ will generate the same output sequences as in Cases B and C. The null error sequence $\varepsilon_x=(0)^\infty$ is a trivial case.

The distinct ambiguous output sequences are characterized in Cases B and C. (Case A described only the all-zeros output sequence and Case D described no new ambiguous output sequences.) The number of ambiguous output sequences described in Cases B and C is $2^{\frac{W+3}{2}}-1$ each, but since the all-zeros sequence is described in both cases, the number of ambiguous output sequences is one less than the sum, namely, $2^{\frac{W+5}{2}}-3$.

The distinct input sequences that generate the ambiguous output sequences are also completely described in Cases B and C. However, these two cases both include the four sequences $\boldsymbol{x}_{-2}^{W} = (0, 0, 0, 0, \ldots)$, $\boldsymbol{x}_{-2}^{W} = (1, 1, 1, 1, \ldots)$, $\boldsymbol{x}_{-2}^{W} = (1, 0, 1, 0, \ldots)$ and $\boldsymbol{x}_{-2}^{W} = (0, 1, 0, 1, \ldots)$, so the number of distinct input sequences that generate ambiguous output sequences is $2^{\frac{W+7}{2}} - 4$. The number of input sequences that generate the all-zeros output sequence is 4, and the remaining $2^{\frac{W+7}{2}} - 8 = 2^{T+4} - 8$ input sequences generate nonzero ambiguous output sequences, as applied in (4).

When a nonzero ambiguous sequence is received, the probably of bit error is $1/4$. To see this, assume that x is the input that generates a nonzero ambiguous output sequence, and x' is a distinct input sequence that generates the same nonzero ambiguous output sequence. Then, the corresponding truncated null error sequence must be one of

$$\varepsilon_x = \{(+0)^\infty, (0+)^\infty, (-0)^\infty, (0-)^\infty\}.$$

A zero in the truncated null error sequence corresponds to agreement between the corresponding bits in x and x', and $\{+, -\}$ corresponds to disagreement. Thus, half the bits in the input sequences agree. The probability that the detector will choose the incorrect input sequence is $1/2$, so the net probability of bit error is $1/4$.

B. Precoder Existence

In this section, we show that the only two precoders of degree less than or equal to three that map input sequences to output sequences in a one-to-one fashion for the EPR4 channel are $f(D) = 1 \oplus D^2$ and $f(D) = 1 \oplus D \oplus D^2 \oplus D^3$.

Let $f(D) = f_0 \oplus f_1 D \oplus f_2 D^2 \oplus f_3 D^3$, and assume that $f_0 = 1$. As before, $x(D)$ is the input to a precoder $1/f(D)$, and the output of the precoder is $a(D) = \frac{x(D)}{f(D)}$. Let $a(D)$ and $a'(D)$ be two distinct channel input sequences which produce the same output sequence. We want to choose an $f(D)$ such that $a(D)f(D) = a'(D)f(D)$ for the terms that are inside the window $1, \ldots, W$. The D^k term of this equality is

$$\begin{aligned} f_0 a_k \oplus f_1 a_{k-1} \oplus f_2 a_{k-2} \oplus f_3 a_{k-3} \\ = f_0 a'_k \oplus f_1 a'_{k-1} \oplus f_2 a'_{k-2} \oplus f_3 a'_{k-3}. \end{aligned} \quad (13)$$

Consider the case when the truncated null error sequence is $\varepsilon_a = (+0)^\infty$, with

$$\boldsymbol{a}_{-2}^{W} = (1, a_{-1}, 1, a_1, 1, a_3, 1, \ldots)$$

and

$$\boldsymbol{a}_{-2}^{W} = (0, a_{-1}, 0, a_1, 0, a_3, 0, \ldots).$$

These two input sequences will generate the same output sequence. Substituting these two sequences into (13), we get

$$\begin{aligned} f_0 \oplus f_1 a_{k-1} \oplus f_2 \oplus f_3 a_{k-3} &= f_1 a_{k-1} \oplus f_3 a_{k-3} \\ f_0 \oplus f_2 &= 0. \end{aligned}$$

That is, $f_0 = f_2 = 1$. If instead we have the case $\varepsilon_a = (0+)^\infty$, then the D^k term becomes

$$\begin{aligned} f_0 a_k \oplus f_1 \oplus f_2 a_{k-2} \oplus f_3 &= f_0 a_k \oplus f_2 a_{k-2} \\ f_1 \oplus f_3 &= 0. \end{aligned}$$

That is, $f_1 = f_3 = 1$ or $f_1 = f_3 = 0$. Applying the other possible truncated null error sequences does not result in any additional restrictions on the precoder. Therefore, the two precoders which map inputs to outputs in a one-to-one fashion are $f(D) = 1 \oplus D^2$ and $f(D) = 1 \oplus D \oplus D^2 \oplus D^3$. We note that these are also the only two precoders that are divisible by $1 \oplus D^2$.

References

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

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Communications*, vol. 2. Geneva, Switzerland, May 1993, pp. 1064–1070.

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

[4] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.

[5] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 909–926, May 1998.

[6] W. E. Ryan, L. L. McPheters, and S. W. McLaughlin, "Combined turbo coding and turbo equalization for PR4-equalized Lorentzian channels," in *Proc. Conf. Information Sciences and Systems*, Mar. 1998.

[7] W. E. Ryan, "Performance of high rate turbo codes on a PR4-equalized magnetic recording channel," in *Proc. IEEE Int. Conf. Communications*. Atlanta, GA, June 1998, pp. 947–951.

[8] T. Souvignier, M. Öberg, P. H. Siegel, R. E. Swanson, and J. K. Wolf, "Turbo decoding for partial response channels," *IEEE Trans. Commun.*, vol. 48, pp. 1297–1308, Aug. 2000.

[9] C. Heegard, "Turbo coding for magnetic recording," in *Proc. IEEE Information Theory Workshop*, San Diego, CA, Feb. 1998, pp. 18–19.

[10] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.

[11] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, pp. 1645–1646, Aug. 1996.

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

[13] F. R. Kschischang, B. J. Frey, and P. G. Gulak, "Concurrent turbo-decoding," in *Proc. IEEE Int. Symp. Information Theory*, Ulm, Germany, June/July 1997.

[14] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.

[15] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[16] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.

[17] A. D. Weathers, S. A. Altekar, and J. K. Wolf, "Distance spectra for PRML channels," *IEEE Trans. Magn.*, vol. 33, pp. 2809–2811, Sept. 1997.

[18] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping Univ., S-581 83 Linköping, Sweden, 1996.

[19] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A soft-input soft-output APP module for iterative decoding of concatenated codes," *IEEE Commun. Lett.*, vol. 1, pp. 22–24, Jan. 1997.

[20] J. L. Fan, A. Friedmann, E. Kurtas, and S. W. McLaughlin, "Low density parity check codes for magnetic recording," in *Proc. 37th Annual Allerton Conf. Communication, Control, and Computing*, 1999, pp. 1314–1323.

[21] H. Song, R. M. Todd, and J. R. Cruz, "Low density parity check codes for magnetic recording channels," *IEEE Trans. Magn.*, vol. 36, pp. 2183–2186, Sept. 2000.

[22] J. Garcia-Frias, "Decoding of low-density parity check codes over finite-state binary Markov channels," in *Proc. IEEE Int. Symp. Information Theory*, Washington, DC, June 2001, p. 72.

[23] D. MacKay. Encyclopedia of Sparse Graph Codes. [Online]. Available: http://wol.ra.phy.cam.ac.uk/mackay/codes/data.html.

[24] S. A. Altekar, M. Berggren, B. E. Moision, P. H. Siegel, and J. K. Wolf, "Error-event characterization on partial-response channels," *IEEE Trans. Inform. Theory*, vol. 45, pp. 241–247, Jan. 1999.