Joint Message-Passing Symbol-Decoding of LDPC Coded Signals over Partial-Response Channels

Rathnakumar Radhakrishnan and Bane Vasić
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ-85721
Email: {rathna, vasic}@ece.arizona.edu

***Abstract*— We consider the problem of joint detection and decoding of low-density parity-check (LDPC) coded signals over partial response (PR) channels. A method to graphically represent the constraints imposed by the channel and the code on the channel output sequence is introduced. This enables the design of a detector and decoder that estimates a posteriori probabilities of noiseless channel output symbols rather than binary channel inputs. By running the sum-product algorithm (SPA) on this graph, a joint decoder is obtained that is shown to perform significantly better than the turbo-equalizer.**

I. Introduction

Low-density parity-check (LDPC) codes, invented by Gallager [1] and rediscovered by Mackay and Neal [2] have been shown to be capacity achieving on memoryless channels. It has also been shown to achieve excellent error rate performance over channels with memory, such as magnetic storage [3] and optical communication channels [4]. For reasons of bandwidth efficiency, these inter-symbol interference (ISI) channels are equalized to a partial-response (PR) target with relatively small memory compared to the unequalized channel. Any LDPC decoder must cope with the controlled amount of ISI introduced by the PR. For an uncoded system, the channel input sequence is optimally detected in the presence of ISI by the Viterbi algorithm. For an LDPC coded system, the optimal maximum a posteriori (MAP) decoding is impractical, but good error rate performance can be achieved by using the *turbo* principle [5]. In this technique, information is iteratively passed back and forth between a soft-input soft-output (SISO) detector and a SISO LDPC decoder. The ISI in the channel output is eliminated by a detector using the channel information unknown to the decoder, and subsequently, the output is decoded by a LDPC decoder using the code structure information unknown to the detector. Since, Viterbi algorithm produces only hard decisions, algorithms like soft-output Viterbi algorithm (SOVA) [6] and Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [7] are used for SISO detection and the well-known sum-product algorithm (SPA) [8] is used for SISO decoding. This iterative algorithm is known as *turbo-equalizer*. Though sub-optimal, it is the best and most widely used decoder known today.

The performance can be improved beyond what is achieved by turbo-equalizer, if both channel and the code information are simultaneously used to make decisions on the channel inputs. This is referred to as joint detection and decoding or simply as joint decoding, and is the focus of this paper. It has also been the focus of some recent works in [9], [10], [11] and [12]. A practical and popular approach in this direction has been to design message-passing (MP) decoding algorithms that operate on a graph, which represents the constraints imposed by both the channel and the LDPC code.

In turbo-equalizer, the channel constraints are represented by the channel trellis and the code constraints are represented by the bipartite graph of the code, known as the Tanner graph. Detectors like SOVA and BCJR operate serially on the trellis, while the LDPC decoder operates parallely on the Tanner graph. While aiming to conceive a joint MP decoding algorithm operating on a graph, it is logical to first consider the problem of representing the channel constraints as a graph and then to design a parallel MP detection algorithm that operates on this graph. This problem has been addressed by Kurkoski et. al. in [9], who introduced parallel bit-based and state-based MP algorithms for channel detection. The bit-based MP algorithm is useful only in channels with unit memory length, but has been modified for use in channels with longer memory by Colavolpe et. al. [10]. Even though in [9], the state-based MP detector was combined with the LDPC MP decoder to obtain a joint MP decoder, it achieved at best the same performance as the turbo-equalizer. This is due to the fact that the joint MP decoder is simply a parallel schedule of the turbo-equalizer.

The challenge in jointly using both the channel and code information arises from the fact that the channel imposes constraints on the channel *output* sequences, whereas the code imposes constraints on the channel *input* sequences. The idea motivating our approach is the observation that by imposing constraints on the channel input sequences, the code also imposes certain constraints on the noiseless channel output sequences. In this paper, we modify the LDPC MP decoder to produce information on the noiseless channel output symbols rather than on channel inputs. This enables us to combine it with a modified version of the state-based MP detector to design a joint decoder, that significantly surpasses the performance of the turbo-equalizer. The joint decoder estimates a posteriori probabilities (APPs) of channel output symbols, from which APPs of channel inputs are derived. Also, as it will be shown, this algorithm can be used irrespective of the channel memory length, although as described in Section III, it may perform relatively better for channels with small memory.

The rest of the paper is organized as follows. We describe a graphical model used to represent the channel and describe an optimal MP detection algorithm operating on this graph in Section II. This model is extended to include code constraints and a joint MP decoding algorithm that operates on this combined graph is described in Section III. Bit error simulated performance for an LDPC code is shown in Section IV and finally, the paper is concluded in Section V.

II. Message-Passing Detection Algorithm

A. System model

Fig. 1 shows the system model considered, where the channel response is represented by a polynomial $h(D)$ or the corresponding coefficient vector $\mathbf{h}$. A sequence $\mathbf{u}$ of k binary bits is encoded by an LDPC code into a codeword $\mathbf{x}$ of n binary bits. The codeword is transmitted through the PR channel, whose non-binary output is corrupted by additive white Gaussian noise (AWGN). The noiseless channel output, $\mathbf{y} = \mathbf{x} * \mathbf{h}$, is of length $n + m$, where $*$ denotes the convolution operator and m denotes the channel memory length. The noisy channel output is given by $\mathbf{r} = \mathbf{y} + \mathbf{z}$. We first consider an uncoded system, where the optimal detector is the one that estimates MAP probability of the bits $p(x_i|\mathbf{r})$, $\forall\, i = 0, 1, \ldots, k-1$, where x_i is the i^{th} element of the vector $\mathbf{x}$ and $x_i \in \{0, 1\}$. These quantities are efficiently determined by operating the BCJR algorithm on the channel trellis.

B. Channel graph

The trellis of a channel represents the constraints imposed on the range of noiseless channel output sequences. A channel trellis can be given as a factor graph [8] shown in Fig. 2, where $q_0, q_1, \ldots, q_{n+1}$ represents state (hidden) variables, $x_0, x_1, \ldots, x_n$ represents channel inputs and $y_0, y_1, \ldots, y_n$ represents noiseless channel outputs. The factor graph can be divided into n sections, where the i^{th} section denoted by T_i is defined by all valid triples $\{q_i, y_i, q_{i+1}\}$. Therefore, each section acts as a *local* constraint of the channel. Consequently, a sequence of state and channel output variables $\{q_0, q_1, \ldots, q_{n+1}, y_0, y_1, \ldots, y_n\}$ is valid if and only if it satisfies all local constraints $T_0, T_1, \ldots, T_n$.

When a *global* channel constraint is factored into local channel constraints, numerous scheduling schemes for implementing the BCJR algorithm are possible. Typically, one instance of a BCJR algorithm is operated on one section of the trellis at any time instant and is progressively moved to other sections. This scheduling is known as fully-serial [8]. On the other extreme, n instances of the BCJR algorithm can operate on each section of the trellis simultaneously, exchanging information through the state variables during every iteration. This scheduling is known as fully-parallel, and is referred to as the parallel state-based MP algorithm in [9]. Naturally, intermediate scheduling schemes are possible. For example, the factor graph shown in Fig. 2 can be divided into p sections ($p < n$), and during every iteration, p instances of the BCJR algorithm can operate on each of these sections, exchanging information with the adjacent ones, but within each section, the BCJR can operate serially.

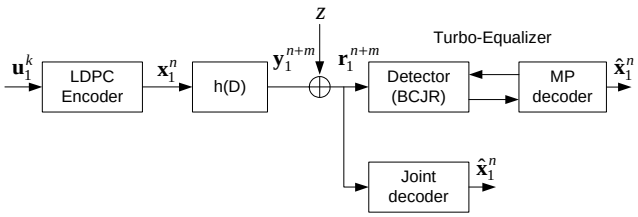

Fig. 1. Block diagram of a PR system. Decoder is either turbo-equalizer (upper branch) or joint decoder (lower branch) and noise is modeled as additive white Gaussian.

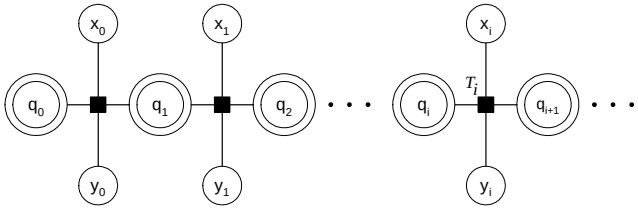

Fig. 2. Generalized factor graph representation of a PR channel trellis.

In this work, we consider the fully-parallel MP scheduling. This necessitates the need for state variables to store and relay information between adjacent sections after every iteration. We modify the factor graph representing the channel constraints to remove the need for state variables and correspondingly alter the detection algorithm. The graph is now simply represented as a bipartite graph G shown in Fig. 3, where circles correspond to the noiseless channel symbols y_i, and squares correspond to the local channel constraints. These sets of nodes are referred to as symbol nodes and channel nodes respectively. Every channel node represents two sections of the trellis. For example, a channel node s_i acts as a local constraint and represents all valid 5-tuples $\{q_i, y_i, q_{i+1}, y_{i+1}, q_{i+1}\}$. Therefore, unlike the factor graph in Fig. 3, every section of the trellis is represented by two channel nodes in this graph. As will be described in the next section, information pertaining to state transition probabilities is exchanged between symbol nodes through the channel nodes. Like the factor graph of the trellis, graph G is a generic cycle-free representation of a PR channel, irrespective of its memory length. However, the size of the set represented by the channel nodes increases exponentially with increase in memory length. If memory length is m, the size of this set is $2^{(m+2)}$.

C. Message-passing symbol detector

Now, we describe a MP detection algorithm that operates on graph G and produces APPs of output symbols, from which APPs of channel inputs are derived. If x_k and y_k are the channel input and noiseless output at time k, then

$$y_k = f(x_k, x_{k-1}, \ldots, x_{k-m}), \tag{1}$$

where the function $f()$ is determined by the channel response $h(D)$. Let $A = \{a_0, a_1, \ldots, a_{2^m-1}\}$ be the set of possible

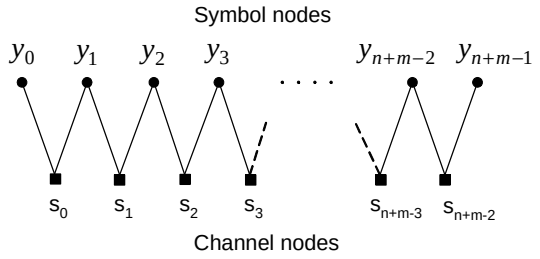

Fig. 3. A graph that represents constraints imposed by the channel on the noiseless channel output sequences.

noiseless channel output symbols when the current channel input $x_k = 0$, and let $B = \{b_0, b_1, \ldots, b_{2^m-1}\}$ be the set of possible noiseless channel output symbols when the channel input $x_k = 1$. Then, APPs of the channel input x_k are given as,

$$p\left(x_k = 0|\mathbf{r}\right) = \sum_{i=0}^{2^m-1} p(y_i = a_i|\mathbf{r}). \tag{2}$$

Though elements of set A and B may not be unique, we emphasize that they correspond to a unique state transition in the corresponding channel trellis. Since the MP detection algorithm is not operated explicitly on the trellis, we simply refer the quantities $p(y_i)$ as symbol probabilities rather than state transition probabilities.

Let φ_i denote the set of symbol pairs represented by channel node s_i of the graph shown in Fig. 3. Therefore, $(y_i, y_{i+1}) \in \varphi_i$, $\forall\ i$. The constraints imposed on the output sequences by the channel can be viewed as a *code*, where each channel node s_i corresponds to a code of length 2 and the set φ_i corresponds to its set of *codewords*. With the knowledge of φ_i for all channel nodes, the output sequence can thus be *decoded* without the use of state variables. Since the underlying channel graph G is a tree, APPs can be estimated optimally using the SPA. Now, the MP detection algorithm is given as follows.

MP Detection Algorithm:

1) Initialization: Since, channel inputs are i.i.d, all state transitions are initially equally likely. Therefore, $p(y_i) = 1/2^{m+1}$, $\forall\ i = 0, 1, \ldots, n+m-1$.
2) Message from symbol nodes to channel nodes during the t^{th} iteration:
$$\begin{aligned} M^{(t)}_{y_k \to \mathrm{s}_{k-1}} &= M^{(t-1)}_{\mathrm{s}_k \to y_k} \\ M^{(t)}_{y_k \to \mathrm{s}_k} &= M^{(t-1)}_{\mathrm{s}_{k-1} \to y_k},\ \forall\ k = 1, \ldots, n+m-2 \end{aligned} \tag{3}$$
where, $M^{(t)}_{y_k \to s_{k-1}}$ denotes the message sent from symbol node y_k to channel node s_{k-1} during the t^{th} iteration. Other terms are defined similarly.
3) Message from channel nodes to symbol nodes during the t^{th} iteration:
$$\begin{aligned} M^{(t)}_{\mathrm{s}_k \to y_k} &= \frac{p(y_k|r_k, r_{k+1}, \varphi_k)}{p(r_k|y_k)p(y_k)},\ \forall\ y_k \\ M^{(t)}_{\mathrm{s}_k \to y_{k+1}} &= \frac{p(y_{k+1}|r_k, r_{k+1}, \varphi_k)}{p(r_{k+1}|y_{k+1})p(y_{k+1})},\ \forall\ y_{k+1}. \end{aligned} \tag{4}$$
Messages received from the symbol nodes during the current iteration serve as the *a priori* symbol probabilities for the channel node operation.
4) APPs of channel output symbols: After repeating the above steps for a fixed number of iterations, APPs of channel output symbols are calculated as,
$$M^{(t)}_{y_k} = M^{(t)}_{\mathrm{s}_k \to y_k} \cdot M^{(t)}_{\mathrm{s}_{k-1} \to y_k} \cdot p(r_k|y_k) \cdot p(y_k) \tag{5}$$
where, $p(y_k)$ is the initial *a priori* probability.
5) APPs of channel input bits: The algorithm halts after calculating channel input APPs using Eqn. 2.

Remark: $p(y_k|r_k, r_{k+1}, \varphi_k)$ and $p(y_{k+1}|r_k, r_{k+1}, \varphi_k)$ in Eqn. 4 can be computed in a straightforward way, since $|\varphi_k| \leq 2^{m+2}$, and m is small. In the context of a trellis, this is same as operating BCJR only on the two sections of the trellis represented by the channel node s_k.

When the number of iterations equal the length of the sequence, the APPs obtained using the above algorithm are same as that obtained from the BCJR algorithm. Usually, only a small number of iterations are required to obtain performance close to optimal. However, as observed in [9], all schedulings of the BCJR algorithm, except the fully-serial scheduling, result in an error floor if enough iterations are not run. Like other MP detection algorithms proposed in the literature, this algorithm is more complex than BCJR, primarily due to the parallel scheduling of the algorithm, but it can potentially reduce latency time and is suitable for high-speed applications. Its most important advantage is that it can be combined with a LDPC MP decoder to obtain a joint MP decoder.

III. Joint Message-Passing Symbol-Decoding Algorithm

In this section, we extend the graphical model described earlier to include constraints imposed by the parity checks of the LDPC code. The tripartite graph shown in Fig. 4 is obtained by including the parity check nodes to the channel graph of Fig. 3. Connections between the parity check nodes and the symbol nodes are defined in the same way as the connections between the parity check nodes and the variable nodes. Using this combined graph, a joint decoding algorithm is developed that estimates the symbol APPs using both the channel and the code information simultaneously. The joint decoding algorithm is outlined below.

Joint MP Decoding Algorithm:

1) Initialization: Symbol *a priori* probabilities $p(y_i) = 1/2^{m+1}$.
2) Message from symbol nodes to channel nodes during the t^{th} iteration: For every k, compute,
$$\begin{aligned} M^{(t)}_{y_k \to \mathrm{s}_{k-1}} &= M^{(t-1)}_{\mathrm{s}_k \to y_k} \cdot \prod_{j|h_{jk}=1} M^{(t-1)}_{\mathrm{c}_j \to y_k} \\ M^{(t)}_{y_k \to \mathrm{s}_k} &= M^{(t-1)}_{\mathrm{s}_{k-1} \to y_k} \cdot \prod_{j|h_{jk}=1} M^{(t-1)}_{\mathrm{c}_j \to y_k}. \end{aligned} \tag{6}$$
3) Message from symbol nodes to check nodes during the t^{th} iteration: If $H = \{h_{ij}\}$ is the parity check matrix

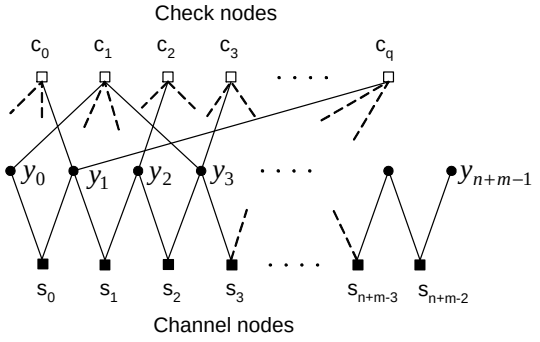

Fig. 4. A graph that represents constraints imposed by the channel and the parity checks of the LDPC code on the noiseless channel output sequences. Parity checks imposes certain constraints on the channel output sequences by imposing constraints on the channel input sequences.

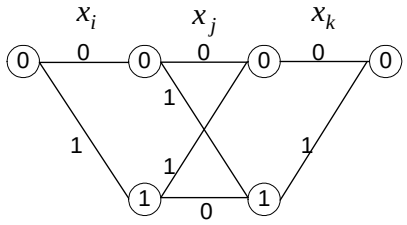

Fig. 5. Trellis diagram of a single parity check code of length 3. The states indicate the parity of the incoming sequences.

of the LDPC code, then for every i and k such that $h_{ik} = 1$, compute,

$$M^{(t)}_{y_k \to c_i} = M^{(t-1)}_{s_k \to y_k} \cdot M^{(t-1)}_{s_{k-1} \to y_k} \cdot \prod_{j|h_{jk}=1, j \neq i} M^{(t-1)}_{c_j \to y_k}. \tag{7}$$

4) Message from channel nodes to symbol nodes during the t^{th} iteration: This is same as Eqn. 4.
5) Message from check nodes to symbol nodes during the t^{th} iteration: For every i and k such that $h_{ik} = 1$, compute,

$$M^{(t)}_{c_i \to y_k} = \frac{p(y_k|\mathbf{r}_i, C_i)}{p(r_k|y_k)p(y_k)}, \ \forall \ y_k \tag{8}$$

where, C_i denotes the event that the check c_i is satisfied and $\mathbf{r}_i$ denotes the set of received samples at the locations of the variable nodes connected to check c_i.
6) APPs of channel symbols: For every k, compute,

$$M^{(t)}_{y_k} = M^{(t)}_{s_k \to y_k} \cdot M^{(t)}_{s_{k-1} \to y_k} \cdot \prod_{j|h_{jk}=1} M^{(t)}_{c_j \to y_k} \cdot p(r_k|y_k) \cdot p(y_k) \tag{9}$$

where, $p(y_k)$ is the initial *a priori* probability.
7) APPs of channel input bits: After every iteration, the channel input APPs are calculated using Eqn. 2. All the above steps are repeated until either the decoded sequence satisfies all channel and code constraints or a preset maximum number of iterations is reached.

Remark: Observe that the check node operation of the LDPC MP decoder is modified to provide symbol information for use in joint decoding. In the traditional decoder, the check node operation is efficiently computed by the *tanh* function, whereas the new check node operation is more complex. We describe next an efficient method to compute this new operation.

For ease of exposition, consider a degree 3 check node $c_l = x_i \oplus x_j \oplus x_k$. The check node c_l implies that the three variable nodes form a single parity check code, denoted by $\mathbb{C}_X$. This is compactly represented by the code trellis shown in Fig. 5. All paths beginning and ending at state 0 correspond to codewords of the code $\mathbb{C}_X$. Input bit APPs conditioned on the event C_l can be determined by operating the BCJR algorithm on this trellis, which simply turns out be the *tanh* check node operation.

The check node c_l is connected to symbol nodes y_i, y_j and y_k in the combined graph. By imposing constraints on the variable nodes x_i, x_j and x_k, the check c_l also imposes certain constraints on the corresponding output symbols. In other words, the check node c_l implies that the three symbol nodes also form a code, which we denote by $\mathbb{C}_Y$. In order to obtain symbol APPs, a trellis for the code $\mathbb{C}_Y$, referred to as the expanded code trellis, is constructed by *expanding* the edges of the code trellis shown in Fig. 5. The expanded code trellis is shown in Fig. 6. An edge representing a state transition in code trellis is now replaced by 2^m edges representing all possible noiseless channel outputs generated during the corresponding state transition. For example, if $x_i = 0$ is transmitted, the corresponding noiseless channel output $y_i \in A$. Further, the edge labels are elements of set A or B depending on whether the corresponding edge label in code trellis is 0 or 1. If the three symbol nodes are *independent*, i.e. they have at least m other symbol nodes between them, then all paths beginning and ending at state 0 of the expanded code trellis correspond to codewords of the code $\mathbb{C}_Y$. Therefore, the symbol APPs of y_i, y_j and y_k conditioned on the event C_l is determined by operating the BCJR algorithm on the expanded code trellis.

If the symbol nodes connected to a check node are not independent, the above method can still be used for an approximate calculation of Eqn. 8. In principle, the joint symbol decoder can be applied for channels with any memory length, although the number of check nodes that violates the independence property will be lower for channels with small memory. The performance of the joint decoder would be severely affected if the LDPC code contains many pairs of consecutively occurring variable nodes connected to a check, as such codes would result in many four-cycles in the combined graph.

In practice, the check node degree is much higher than 3, but the computation of Eqn. 8 using the expanded trellis is still simple, since the expanded code trellis will always have only two states as long as the check node represents a single parity check code.

IV. Simulation Results

We illustrate the performance of the joint symbol-decoding algorithm by simulating an LDPC coded PR system, where

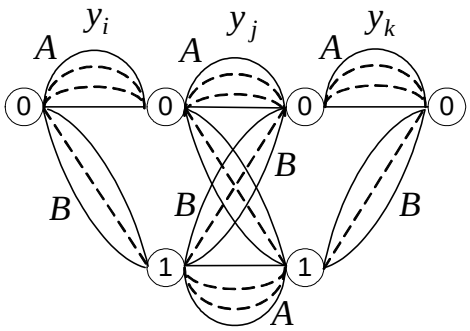

Fig. 6. Expanded code trellis of Fig. 5. The state transition edge labels are either elements of set A or B.

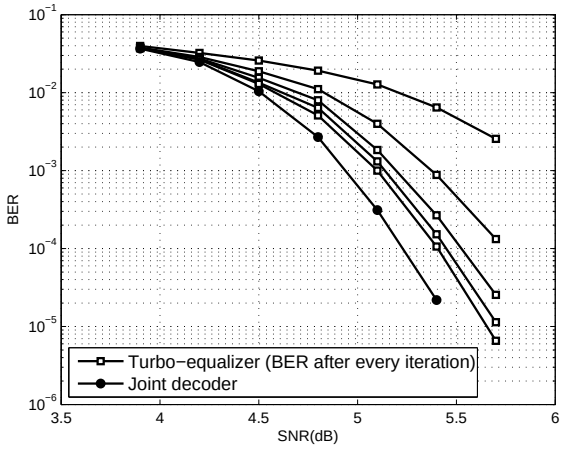

Fig. 7. Bit error rate comparison of (1908,212) random LDPC code on a PR4 channel when decoded using turbo-equalizer and the joint decoder.

the channel is given by the impulse response $[1, 0, -1]$ (PR4), and the LDPC code is of length 1908 and rate 0.89 [13]. The channel output sequences are decoded by using both turbo-equalizer and the joint symbol-decoder. When turbo-equalizer is used, the number of global iterations is restricted to 5 and the number of internal LDPC decoder iterations is restricted to 3. These settings give the best bit error rate (BER) performance. Increasing the number of global iterations beyond 5 does not improve the performance significantly. When the joint symbol-decoder is used, the number of iterations is restricted to 60. The performance comparison is shown in Fig. 7. At a signal-to-noise ratio (SNR) of 5.4 dB the BER obtained by the joint decoder is almost an order of magnitude better than the turbo-equalizer. Also, the figure suggests that the gain increases with increasing SNR. Among the 212 parity checks of this code, 100 parity checks contain at least one pair of variable nodes (or symbol nodes) that are not independent. However, the joint decoder was applied to the code without any modification, implying that the computation of Eqn. 8 was approximate. In spite of this, the decoder was able to achieve significant gain over the turbo-equalizer.

V. Conclusion

The problem of joint detection and decoding of LDPC coded signals over partial response channels is considered. In order to jointly use both the channel and code information, the LDPC decoder is modified to produce information on channel output symbols rather than on channel inputs. This is combined with a message-passing detector to develop a joint decoder that estimates channel input APPs by first estimating channel output symbol APPs. The performance of this decoder is shown to significantly outperform that of the turbo-equalizer for a random LDPC code of rate 0.89.

VI. Acknowledgement

This work was supported by Seagate Technology and the NSF under grant ECCS-0725405.

References

[1] R. G. Gallager, "Low-density parity-checks codes," *IEEE Trans. Inform. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[2] D. J. C. Mackay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electronics Lett.*, vol. 32, no. 18, pp. 1645–1646, 1996.

[3] 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. on Communication, Control and Computing*, 1999, pp. 1314–1323.

[4] B. Vasić, B. Djordjevic, and R. Kostuk, "Low-density parity check codes and iterative decoding for long haul optical communication systems," *J. Lightwave Technol.*, vol. 21, no. 2, pp. 438–446, Feb. 2003.

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

[6] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Global Telecomm. Conf. (GLOBECOM'89)*, vol. 3, Dallas, Texas, Nov. 1989, pp. 1680–1686.

[7] 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. 20, no. 2, pp. 284–287, Mar. 1974.

[8] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[9] B. M. Kurkoski, P. H. Siegel, and J. K. Wolf, "Joint message-passing decoding of LDPC codes and partial-response channels," *IEEE Trans. Commun.*, vol. 48, no. 6, pp. 1410–1422, June 2002.

[10] G. Colavolpe and G. Germi, "On the application of factor graphs and the sum product algorithm to ISI channels," *IEEE Trans. Commun.*, vol. 53, no. 5, pp. 818–825, May 2005.

[11] P. Pakzad and V. Anantharam, "Kikuchi approximation method for joint decoding of LDPC codes and partial-response channels," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1149–1153, July 2006.

[12] G. Colavolpe, "On LDPC codes over channels with memory," *IEEE Trans. Wireless Commun.*, vol. 5, no. 7, pp. 1757–1765, July 2006.

[13] D. Mackay. Encyclopedia of sparse graph codes. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data