

# Transactions Letters

## Kikuchi Approximation Method for Joint Decoding of LDPC Codes and Partial-Response Channels

Payam Pakzad and Venkat Anantharam

**Abstract**—In this letter, we apply the Kikuchi approximation method to the problem of joint decoding of a low-density parity-check code and a partial-response channel. The Kikuchi method is, in general, more powerful than the conventional loopy belief propagation (BP) algorithm, and can produce better approximations to an underlying inference problem. We will first review the Kikuchi approximation method and the generalized BP algorithm, which is an iterative message-passing algorithm based on this method. We will then report simulation results which show that the Kikuchi method outperforms the best conventional iterative method.

### I. INTRODUCTION

ITERATIVE decoding techniques, such as the decoding algorithms for turbo codes (see [2]) and low-density parity-check (LDPC) codes (see [3] and [5]), have enjoyed much attention in the recent years due to their apparent ability to efficiently produce good approximations to the optimal maximum *a posteriori* (MAP) estimates. In applications such as magnetic storage, where a target bit-error rate (BER) needs to be achieved, these codes can operate at much lower signal-to-noise ratios (SNRs) than any other practical error-correcting approach. Operation at lower SNRs, in turn, translates to achieving higher bit densities on the same storage device. At high densities, any efficient decoding technique must properly model and address the issue of intersymbol interference (ISI). The ISI in magnetic recording channels are conventionally described using partial-response (PR) models, in which the channel is modeled as a finite impulse response (FIR) filter. “Turbo equalization” is an iterative technique in which information is passed back and forth between soft decoders for the ISI channel, and a preceding error-correcting code. As we will see later in Section III, this corresponds to an implementation of the well-known belief propagation (BP) algorithm of [9] on a graph with cycles.

Paper approved by W. E. Ryan, the Editor for Modulation, Coding, and Equalization of the IEEE Communications Society. Manuscript received June 21, 2004; revised June 6, 2005 and December 19, 2005. This work was supported in part by ONR/MURI under Grant N00014-1-0637, in part by the National Science Foundation under Grant SBR-9873086, in part by the Defense Advanced Research Projects Agency under Grant F30602-00-2-0538, in part by the California Micro Program, in part by Texas Instruments, in part by Marvell Technologies, and in part by STMicroElectronics.

P. Pakzad was with the School of Computer and Communication Engineering, Ecole Polytechnique Federale de Lausanne, Lausanne 1015, Switzerland. He is now with Digital Fountain, Inc., Fremont, CA 94538 USA (e-mail: payam@digitalfountain.com).

V. Anantharam is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: ananth@eecs.berkeley.edu).

Digital Object Identifier 10.1109/TCOMM.2006.877944

In this letter, we propose using a more powerful iterative algorithm, known as the generalized BP (GBP) algorithm based on the Kikuchi approximation method. We will briefly introduce the Kikuchi approximation method and the GBP algorithm in Section II. Readers are referred to [8] and [12] for more details. We will then apply this method to a particular turbo equalization problem, which was addressed using conventional methods in [4]. We will show that the Kikuchi-based GBP algorithm outperforms the best iterative method based on the BP algorithm.

### II. KIKUCHI APPROXIMATION METHOD

We will now give a short description of the Kikuchi approximation method for calculating the desired marginals of a product distribution. Let  $x := (x_0, \dots, x_{N-1})$ , where for each  $i \in [N] := \{0, \dots, N-1\}$ ,  $x_i$  is a variable taking value in  $[q_i] := \{0, \dots, q_i - 1\}$ , with  $q_i \geq 2$ .

Let  $R$  be a collection of subsets of  $[N]$ ; we call each  $r \in R$  a *region*. We assume that each variable index  $i \in [N]$  appears in at least one region  $r \in R$ .

Associated with each region  $r \in R$  is a nonnegative *kernel function*,  $\alpha_r(\mathbf{x}_r)$ , depending only on the variables that appear in  $r$ . Then the corresponding *R-decomposable (Boltzmann) product distribution* is defined as

$$B(\mathbf{x}) := \frac{1}{Z} \prod_{r \in R} \alpha_r(\mathbf{x}_r). \quad (1)$$

Here  $Z$  is the normalizing constant, and is called the *partition function*. For a subset  $s \subset [N]$ , we denote by  $B_s(\mathbf{x}_s) := \sum_{\mathbf{x}_{[N] \setminus s}} B(\mathbf{x})$  the *s-marginal* of  $B(\mathbf{x})$ . We are interested in finding one or more of the  $B_r(\mathbf{x}_r)$ 's for  $r \in R$ , and/or the partition function  $Z$ ; for example, as is the case in later parts of this letter,  $B(\mathbf{x})$  can be the joint *a posteriori* probability density on the variables, which factorizes as a product of conditional probabilities and parity-check terms.

The key idea behind the Kikuchi approximation method is to convert the above marginalization problem into a constrained optimization problem, and then approximate the objective function and the constraint set to obtain an approximation to the desired marginals. To that end, let  $b(\mathbf{x})$  denote a probability distribution on  $\mathbf{x}$ , and denote its marginals by  $b_r(\mathbf{x}_r)$  for each  $r \subseteq [N]$ . We define the *variational free energy* for the problem as

$$F(b(\mathbf{x})) := U(b(\mathbf{x})) - H(b(\mathbf{x})) \quad (2)$$

where  $U := \sum_{\mathbf{x}} b(\mathbf{x}) \prod_{r \in R} \alpha_r(\mathbf{x}_r)$  is the *average energy* and  $H := -\sum_{\mathbf{x}} b(\mathbf{x}) \log(b(\mathbf{x}))$  is the *entropy* of the system.

It can be shown that  $F(b)$  is uniquely minimized when  $b(\mathbf{x})$  equals the Boltzmann distribution  $B(\mathbf{x})$  of (1), and we have

$$F_0 := \min_{b(\mathbf{x})} F(b(\mathbf{x})) = F(B(\mathbf{x})) = -\log(Z). \quad (3)$$

The Kikuchi approximation method proposes to solve a related constrained minimization problem of the following form (see [8] for details):

$$\{B_r(\mathbf{x}_r)\} \simeq \{b_r^*(\mathbf{x}_r)\} := \arg \min_{\{b_r(\mathbf{x}_r)\} \in \Delta_R^K} F_R^K(\{b_r(\mathbf{x}_r)\}). \quad (4)$$

Here  $\Delta_R^K$  is a set of local constraints to enforce consistency between the  $b_r$ 's, and is defined as

$$\Delta_R^K := \left\{ \{b_r(\mathbf{x}_r), r \in R\} : \begin{aligned} &\forall t, u \in R \\ &\text{s.t. } t \subset u, \sum_{\mathbf{x}_{u \setminus t}} b_u(\mathbf{x}_u) = b_t(\mathbf{x}_t) \text{ and} \\ &\forall u \in R, \sum_{\mathbf{x}_u} b_u(\mathbf{x}_u) = 1 \end{aligned} \right\}. \quad (5)$$

Also,  $F_R^K(\{b_r\})$ , known as the Kikuchi free energy, is an approximation to the variational free energy (2), and is defined as

$$F_R^K(\{b_r(\mathbf{x}_r)\}) := \sum_{r \in R} \sum_{\mathbf{x}_r} -b_r(\mathbf{x}_r) \log(\alpha_r(\mathbf{x}_r)) + \sum_{r \in R} \sum_{\mathbf{x}_r} c_r b_r(\mathbf{x}_r) \log(b_r(\mathbf{x}_r)) \quad (6)$$

where  $c_r$ 's are constants known as the *overcounting factors*, and are uniquely defined given the collection  $R$  of regions. More precisely, we set  $c_r = 1$  for the maximal regions of  $R$ , i.e., the regions that are not contained in any other region. For any other region  $s \in R$ , we define  $c_s$  recursively as

$$c_s := 1 - \sum_{r \in R, s \subset r} c_r. \quad (7)$$

In [8], we discuss in detail why this choice of overcounting factors is reasonable; in short, as is evident from the definition of  $c_s$  above, these factors balance the approximation of (6), in the sense that, considering the multiplicities, the terms containing a subset  $\mathbf{x}_s$  of variables appear exactly once in the second sum in (6), which is the approximation to the entropy function  $H(b)$ .

We will refer to the constrained minimization problem of (4) as the Kikuchi approximation problem, where it is understood that the desired marginals  $\{B_r(\mathbf{x}_r)\}$  are approximated by the minimizers  $\{b_r^*(\mathbf{x}_r)\}$ .

As discussed in [11], the standard BP is an algorithm that tries to solve a simple class of Kikuchi approximation problems known as the Bethe case. When using a more suitable choice of the collection  $R$  of the regions, the *Kikuchi beliefs*  $\{b_r^*(\mathbf{x}_r)\}$  of (4) can better approximate the true marginals  $\{B_r(\mathbf{x}_r)\}$  of the

product distribution than the beliefs obtained from the conventional BP algorithm; see, e.g., [8] for discussion and examples.

### A. Graphical Representations of the Kikuchi Problem

The standard technique to solve a constrained optimization problem such as that of (4) is to form the Lagrangian, where for each constraint of (5), a multiplier will be defined. Viewing these multipliers (or a function of them) as “messages” in an iterative message-passing algorithm, one can construct algorithms whose fixed points coincide with the solutions of the Kikuchi constrained optimization; see, e.g., [8] for details. We will describe one such algorithm called GBP in Section II below.

As with the BP algorithm, graphical models can be used to represent a Kikuchi problem and serve as the basis for an iterative message-passing algorithm to solve that problem. We therefore define a graphical representation of a Kikuchi problem as a graph, whose vertices correspond to the regions  $r \in R$ , and whose edges correspond to the consistency constraints that define  $\Delta_R^K$ . Viewing  $R$  as a *partially ordered set* (poset)<sup>1</sup> with respect to set inclusion, it is easy to see that the standard *Hasse diagram*<sup>2</sup> of the poset is one such graph. As we will see in the next section, we then associate with each edge of the graphical representation, a message  $m_{rs}(x_s)$ . The belief function  $b_r(\mathbf{x}_r)$  at a node  $r$  is defined in terms of the messages in a local neighborhood of the corresponding node, and at each iteration of the algorithm, a message  $m_{rs}$  is updated in a way that the beliefs  $b_r(\mathbf{x}_r)$  and  $b_s(\mathbf{x}_s)$  become consistent, i.e.,  $\sum_{\mathbf{x}_{r \setminus s}} b_r(\mathbf{x}_r) = b_s(\mathbf{x}_s)$ .

We restrict our attentions to the graphical representations whose edges are a subset of the edges of the Hasse diagram. Clearly, such representations are not unique, since redundant constraints (and the corresponding edges) may be freely added or removed. It is, therefore, advantageous to find the *minimal* such graphical representation, by removing a maximal set of redundant edges; algorithms on such minimal graphs have the fewest messages and are, hence, the least complex per each iteration. As discussed in full detail in [8], although these minimal graphical representations are not unique in the graph-theoretic sense, all such realizations are equivalent in representing the same Kikuchi free energy approximation, and their corresponding algorithms have the same fixed points.

### B. GBP Algorithm

GBP is a message-passing algorithm that tries to solve the Kikuchi constrained minimization problem (4). As suggested by the name (due to [11]), it is a generalization of the standard BP, and given a good choice of the collection of Kikuchi regions, it is expected to approximate the marginals better than BP.

Let  $G$  be a graphical representation of the Kikuchi problem (4). We associate with each (region) vertex  $r$  of  $G$  a belief function  $b_r(\mathbf{x}_r)$ . We also associate with each (directed) edge ( $p \rightarrow$

<sup>1</sup>A poset is a set, like  $R$  here, together with a relation, e.g., set-inclusion  $\subseteq$ , such that the relation is transitive, reflexive, and antisymmetric. See, e.g., [10] for more details.

<sup>2</sup>Hasse diagram of a poset is a directed graph, with nodes that correspond to the set elements, and where an edge ( $r \rightarrow s$ ) exists if and only if (iff)  $r$  is the *cover* of  $s$ , i.e.,  $s \subset r$  and there is no other  $t \in R$  such that  $s \subset t \subset r$ .

$r$ ) of  $G$  a message function  $m_{pr}(\mathbf{x}_r)$ , which is initialized to 1. For each  $r \in R$ , define its descendants  $D(r) := \{s \in R : s \subset r\}$ , and its  $G$ -parents,  $\mathcal{P}_G(r) := \{s \in R : (s \rightarrow r)$  an edge in  $G\}$ .

For each  $r \in R$ , the belief function  $b_r(\mathbf{x}_r)$  is defined in terms of a local neighborhood of messages, as follows:

$$b_r(\mathbf{x}_r) = k\beta_r(\mathbf{x}_r) \left( \prod_{p \in \mathcal{P}_G(r)} m_{pr}(\mathbf{x}_p) \right) \times \left( \prod_{d \in D(r)} \prod_{p' \in \mathcal{P}_G(d) \setminus (\{r\} \cup D(r))} m_{p'd}(\mathbf{x}_d) \right) \quad (8)$$

where constant  $k$  is chosen to normalize  $b_r$  so it will sum to 1, and  $\beta_r(\mathbf{x}_r) := \prod_{s \in R, s \subset r} \alpha_s(\mathbf{x}_s)$ . Note that the beliefs need only be calculated upon the termination of the algorithm.

At each iteration of the algorithm, the message  $m_{pr}(\mathbf{x}_r)$  corresponding to an edge ( $p \rightarrow r$ ) of the  $G$  is updated to satisfy the edge constraint  $\sum_{\mathbf{x}_p \setminus r} b_p(\mathbf{x}_p) - b_r(\mathbf{x}_r) = 0$ , as shown in (9) at the bottom of the page, where  $k'$  is any convenient constant. Note that the common terms from the numerator and denominator of (9) can be cancelled, but we will not write the explicit general form here.

As shown in [12] and [8], the fixed points of (8) and (9) are precisely the stationary points of the constrained minimization problem (4). Also, it should again be noted that if the poset  $R$  of Kikuchi regions has only two levels, the above algorithm coincides exactly with the standard BP algorithm on the graphical representation  $G$  of  $R$ .

As is the case with the conventional BP algorithm, the convergence behavior of the algorithm may depend on the schedule chosen to update messages according to (9). However, the fixed points of (8) and (9) are clearly independent of the updating schedule, as long as each message is scheduled to be updated frequently. Possible updating schedules include *fully parallel updating*, where at each iteration, all messages are updated simultaneously based on the values of the messages at the previous iteration; and *fully serial updating*, where a full iteration consists of a sequential updating of all the messages according to a predefined order. In either case, the algorithm is run up to a maximum number of iterations, or until an explicit test of convergence of the messages is satisfied, e.g., until a distance function between the beliefs from one iteration to the next is smaller than a fixed constant.

### III. JOINT DECODING OF LDPC CODE AND PR CHANNEL

Consider a PR channel precoded by an LDPC code, as depicted in Fig. 1.

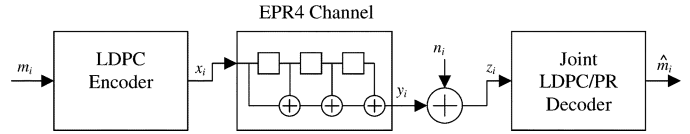


Fig. 1. Block diagram of the serial concatenation of an LDPC code and a PR channel.

We identify the LDPC code by its parity-check matrix  $H_{m \times N}$  where  $N$  is the blocklength of the code, and  $m$  is the number of parity checks. Therefore,  $\mathbf{x} = (x_0, \dots, x_{N-1})$  is a codeword iff it satisfies  $H \cdot \mathbf{x} = \mathbf{0}$  in modulo 2 arithmetics. The PR channel is identified by a transfer polynomial  $h(D) := \sum_{k=0}^{\nu} h_k D^k$ , where  $\nu$  is the degree (memory) of the channel. For example, the EPR4 channel depicted is identified by  $h(D) = 1 + D - D^2 - D^3$ . Therefore, the output of the channel is related to its input by  $y(D) = h(D)x(D)$  in the Z-transfer domain. We will assume an additive white Gaussian noise (AWGN) with variance  $\sigma_n^2$ . The objective is to find the maximum-likelihood estimates of the transmitted code symbols  $x_i$ 's given the noisy observations  $\mathbf{z} = (z_0, \dots, z_{N-1})$ .

It is clear that this problem can be described as that of finding the marginals of a product function, in this case, a joint distribution  $P(\mathbf{x})$ , as posed in Section II.

Let  $P(\mathbf{x})$  denote the joint distribution of the codewords  $\mathbf{x}$  given the observations. Then

$$P(\mathbf{x}) = \frac{1}{Z} \prod_{j \in [m]} 1(H_j \cdot \mathbf{x} = 0) \prod_{i \in [N]} p(z_i | \mathbf{x}) \quad (10)$$

$$= \frac{1}{Z} \prod_{j \in [m]} 1(H_j \cdot \mathbf{x} = 0) \prod_{i \in [N]} p_n(z_i - \sum_{k=0}^{\nu} h_k x_{i-k}) \quad (11)$$

where  $H_j$  denotes the  $j$ th row of the parity-check matrix  $H$ ;  $1(\cdot)$  is the indicator function, taking value 1 if its argument is true, and 0, otherwise; and  $p_n$  is the probability density of the noise. In particular, for the AWGN of variance  $\sigma_n^2$ , we have  $p_n(n) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-n^2/2\sigma_n^2}$ .

The best-performing method discussed in [4] was the state-based message-passing algorithm. This corresponds to the standard BP/GBP algorithm performed on a junction graph of the type pictured in Fig. 2, where the BP algorithm on the upper bubble corresponds to the Gallager–Tanner decoding of the LDPC code (see [3]), and on the lower bubble, it amounts to the Bahl–Cocke–Jelinek–Raviv (BCJR) decoding of the PR channel (see [1]). There are four classes of nodes in this graph:

- the bit nodes corresponding to the bits  $x_i$  of the LDPC code; the regions associated with these nodes are  $\{i\}$  for  $i \in [N]$ ;

$$m_{pr}(\mathbf{x}_r) = k' \frac{\sum_{\mathbf{x}_p \setminus r} \beta_p(\mathbf{x}_p) \left( \prod_{s \in \mathcal{P}_G(p)} m_{sp}(\mathbf{x}_p) \right) \left( \prod_{d \in D(p)} \prod_{s' \in \mathcal{P}_G(d) \setminus (\{p\} \cup D(p))} m_{s'd}(\mathbf{x}_d) \right)}{\beta_r(\mathbf{x}_r) \left( \prod_{s \in \mathcal{P}_G(r) \setminus \{p\}} m_{sr}(\mathbf{x}_r) \right) \left( \prod_{d \in D(r)} \prod_{p' \in \mathcal{P}_G(d) \setminus (\{r\} \cup D(r))} m_{p'd}(\mathbf{x}_d) \right)} \quad (9)$$

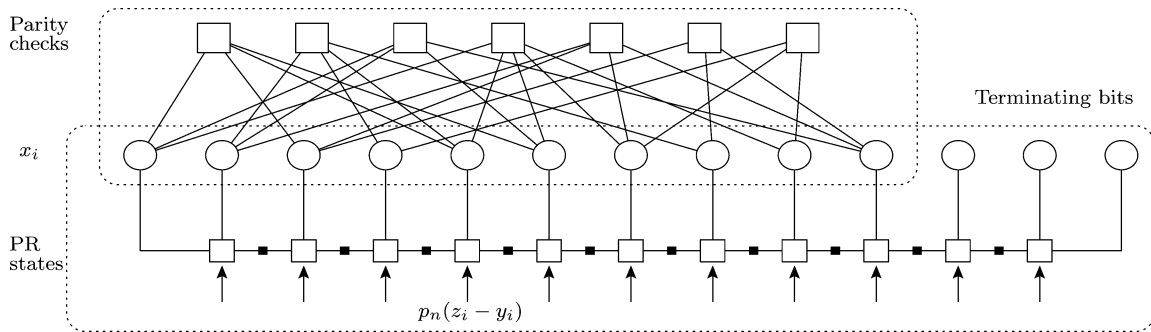


Fig. 2. Junction-graph model for joint BP decoding of the LDPC/PR problem.

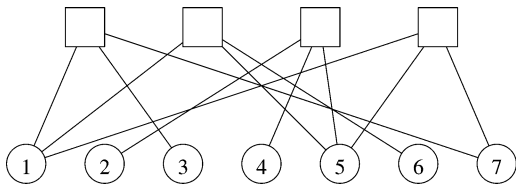


Fig. 3. Simple LDPC code for Example 1.

- the check nodes corresponding to the parity checks of the LDPC code; the corresponding regions are  $\{i : \mathbf{H}_{j,i} = 1\}$  for  $j \in [m]$ ;
- the PR-state nodes corresponding to the states of the PR channel; the associated regions are  $\{i, i-1, \dots, i-\nu\}$  for  $i \in [N]$  where  $\nu$  is the memory of the channel;
- and the PR-intersection nodes, depicted by small solid squares, corresponding to the intersection of neighboring PR-state nodes.

To apply the Kikuchi approximation method for this problem, we used the poset obtained by the *cluster variation method*, see [12]. Specifically, we appended all the intersections of the above regions to form the collection  $R$  of regions; as we discuss in [8], such collections that are closed under pairwise intersections have nice balance properties that are expected to improve the approximation. Notice that good LDPC codes do not have small loops of size four, in which case, no two check nodes can intersect in more than one index. However, the check nodes can have nontrivial intersections with the PR-state nodes. Indeed, it is precisely due to these nontrivial intersections that the Kikuchi free energy approximation is expected to improve upon the conventional BP approximation.

*Example 1:* Consider the simple LDPC code of Fig. 3, with seven bits and four parity checks, in conjunction with a PR channel with memory  $\nu = 2$ . Fig. 4 depicts the Hasse diagram for the associated Kikuchi problem. Specifically, we start with the four check-node regions  $\{1, 3, 7\}$ ,  $\{1, 5, 6\}$ ,  $\{2, 4, 5\}$ , and  $\{1, 5, 7\}$ ; and five PR-state node regions,  $\{1, 2, 3\}$ ,  $\{2, 3, 4\}$ ,  $\{3, 4, 5\}$ ,  $\{4, 5, 6\}$ , and  $\{5, 6, 7\}$ . We then add the seven bit-node regions  $\{1\}, \dots, \{7\}$ . Note that for simplicity, we have omitted the terminating bits in this example. Finally, we complete the poset by appending all the intersections of these regions; here there are four PR-intersection regions, which are the intersections of neighboring

PR-state regions  $\{2, 3\}$ ,  $\{3, 4\}$ ,  $\{4, 5\}$ , and  $\{5, 6\}$ . In addition, there are five other intersection regions, which are either intersections of a check-node and a PR-node region, or that of two check-node regions.<sup>3</sup> These nontrivial intersections are  $\{1, 7\}$ ,  $\{1, 3\}$ ,  $\{1, 5\}$ ,  $\{2, 4\}$ , and  $\{5, 7\}$ .

The full graph in Fig. 4 depicts the complete Hasse diagram for this poset. Here we have also indicated the “redundant” edges with dashed lines; these are the edges that can be removed without changing the fixed point of the Kikuchi problem, to obtain a “minimal graphical representation” for the problem.

Finally, we report the overcounting factors for this simple example. As always, the largest nodes, here, the check nodes and the PR-state nodes, have overcounting factors equal to one. Using straightforward calculation according to (7), for the intersection regions, we obtain  $c_{17} = c_{13} = c_{15} = c_{24} = c_{57} = c_{23} = c_{34} = -1$  and  $c_{45} = c_{56} = -2$ ; and finally for the bit-node regions, we have  $c_1 = c_2 = c_3 = c_6 = c_7 = 0$  and  $c_4 = c_5 = 1$ .

We considered a specific example from [4], with a rate-7/8 LDPC code with block length  $n = 495$ , and with an EPR4 channel. The resulting poset had 495 bit-node regions (singletons), 62 check-node regions (each of size 24, since the LDPC code is regular with 24 bits per check), 495 PR-state node regions (each of size 4, since the channel is EPR4), and a total of 659 nontrivial intersection regions, with nonzero overcounting factors. Of these 659 regions, 494 correspond to the intersections of neighboring PR-state node regions (each of size 3). The remaining 165 regions (with sizes 2 or 3) are the new regions that make the difference from the BP algorithm.

Simulation results are reported in Fig. 5. “BCJR+LDPC” data points are averaged over 500 simulation runs, with 8 iterations between the BCJR and LDPC algorithms, where the LDPC algorithm consisted of 20 iterations. GBP data points are averaged over 50 simulation runs, with 8 iterations of the GBP algorithm with a fully serial updating schedule for each run. For each case, the BER was calculated based on the belief function after the stated number of iterations of the corresponding algorithm.

These results suggest that, as expected, the GBP algorithm considered performs better than the BCJR+LDPC method. Our new technique appears to be particularly well-suited to the low-SNR regime, which is the one that is most important for current magnetic recording applications.

<sup>3</sup>Here the LDPC code does have cycles of length four, so two check nodes can have nontrivial intersections.

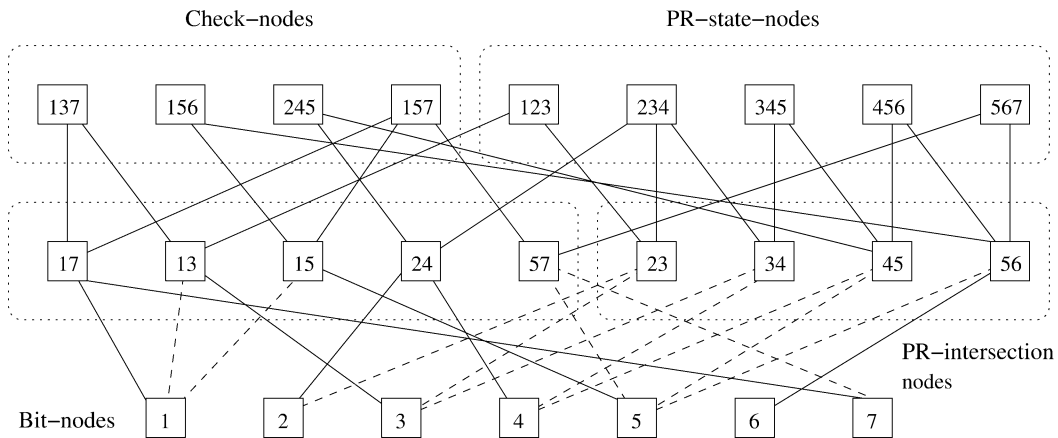


Fig. 4. Graphical representation of Kikuchi regions for Example 1.

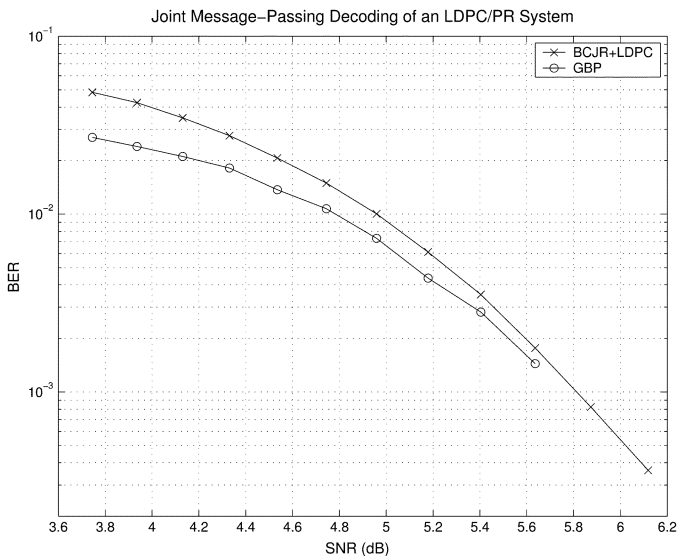


Fig. 5. Simulation results for joint decoding of LDPC/PR.

We conclude this letter with a few remarks on the complexity of the GBP as compared with the standard BP. Clearly, the complexity of GBP depends on the number of edges of the graphical representation. A full comparison would have to include a study of convergence rate of GBP versus BP, as well as the expected number of edges of the minimal graphical representation for a given marginalization problem. At the same time, one needs to develop low-complexity approximations to the full algorithm presented here, which may be more suitable for implementation. All this is beyond the scope of this letter. Here, we content ourselves with reporting the complexity of the specific example which was described above.

The full Hasse diagram for the above problem had 1711 vertices and 3973 edges. The minimal graph for this collection, obtained by removing the redundant edges as prescribed in [8],

has 1711 vertices and 2951 edges. For comparison, the corresponding graph for the problem, before addition of the intersection regions, which corresponds to the loopy BP algorithm, has 2476 edges. Therefore, each iteration of the GBP algorithm entails about as many message updates as the standard BP, although GBP update rules of (9) are admittedly more complex than those of the BP.

REFERENCES

- [1] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.
- [2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun.*, Geneva, Switzerland, May 1993, no. 2, pp. 1064–1070.
- [3] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
- [4] B. Kurkoski, P. Siegel, and J. Wolf, "Joint message-passing decoding of LDPC codes and partial-response channels," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1410–1422, Jun. 2002.
- [5] D. J. C. MacKay, "Good codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
- [6] R. J. McEliece and M. Yildirim, "Belief propagation on partially ordered sets," in *Mathematical Systems Theory in Biology, Communications, Computation, and Finance*. Berlin, Germany: Springer, 2003, pp. 275–300.
- [7] P. Pakzad and V. Anantharam, "Belief propagation and statistical physics," in *Proc. Conf. Inf. Sci. Syst.*, 2002, Paper No. 225, CD-ROM.
- [8] P. Pakzad and V. Anantharam, "Estimation and marginalization using Kikuchi approximation methods," *Neural Comput.*, vol. 17, no. 8, pp. 1836–1873, Aug. 2005.
- [9] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.
- [10] R. Stanley, *Enumerative Combinatorics*. Monterey, CA: Wadsworth & Brooks/Cole, 1986, vol. I.
- [11] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," *Adv. Neural Inf. Process. Syst.*, vol. 13, pp. 689–695, Dec. 2000.
- [12] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free energy approximations and generalized belief propagation algorithms" Mitsubishi Elect. Res. Lab. (MERL) Rep. TR2004-40, 2004.