Low Complexity Markov Chain Monte Carlo Detector for Channels with Intersymbol Interference

Rong-Hui Peng, Rong-Rong Chen and Behrouz Farhang-Boroujeny
Dept. of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT 84112
Email: {peng, rchen, farhang}@eng.utah.edu

Abstract— **In this paper, we propose a novel low complexity soft-in soft-out (SISO) equalizer using the Markov chain Monte Carlo (MCMC) technique. Direct application of MCMC to SISO equalization (reported in a previous work) results in a sequential processing algorithm that leads to a long processing delay in the communication link. Using the tool of factor graph, we propose a novel parallel processing algorithm that reduces the processing delay by orders of magnitude. Numerical results show that, both the sequential and parallel processing SISO equalizers perform similarly well and achieve a performance that is only slightly worse than the optimum SISO equalizer. The optimum SISO equalizer, on the other hand, has a complexity that grows exponentially with the size of the memory of the channel, while the complexity of the proposed SISO equalizers grows linearly.**

I. Introduction

The increasing demand for high speed wireless products has motivated a significant amount of research to combat the intersymbol interference (ISI) resulting from multipath transmission. Early developments dates back to 1960s and 1970s when linear and decision feedback equalizers were developed, [1], [2]. These traditional methods face the problem of noise enhancement, in the case of linear equalizer, or error propagation, in the case of decision feedback equalizers. Furthermore, these methods are based on hard decisions and thus cannot benefit from the modern coding techniques that can approach the channel capacity by making use of soft information. To achieve near capacity performance, modern communication systems operate based on the principle of the maximum a posteriori (MAP) detection. The detection is performed in two steps. In the first step, pilot symbols are used to estimate the channel impulse response. A MAP detector is then used to find the best bit/symbol stream that matches the received signal, given the received signal, channel impulse response, and the constraints imposed by the channel coding. The result is called MAP equalizer and it is optimal in the sense of minimizing the probability of bit errors. However, the computational complexity of this method, even with the use of efficient implementations such as the BCJR algorithm [3] is exponential with respect to length of the channel impulse response and the size of symbol constellation and thus may be prohibitive in many cases. To resolve this problem, approximations to the MAP equalizer have been developed [4] [5], and remain an active area of research in future.

In this paper, we develop a novel low complexity approximation to the MAP equalizer based on Markov chain Monte Carlo (MCMC) principles, [6]. Throughout this paper we use the name soft-in soft-out (SISO) equalizer to refer to MAP equalizer and its approximations. We show that direct application of MCMC to SISO equalization is inappropriate as it results in a sequential processing algorithm that leads to a long processing delay in the communication link. Using the tool of factor graph, we propose a novel parallel processing algorithm that reduces the processing delay by orders of magnitude. We present numerical results that show desirable behavior of the proposed algorithms. The proposed algorithms are also compared with the optimum SISO (i.e., MAP) equalizer which is implemented using the BCJR algorithm [3]. We find that at the cost of a slight performance degradation, the proposed SISO equalizers result in a reduction of the complexity from exponential (with respect to the channel memory) to linear.

In a recent work, Kashif et al [7] have also explored the use of MCMC for SISO equalization. The emphasis of [7] is on non-linear channels with very short memory - the channels considered have memory of two samples only. It also explores a number of MCMC detection algorithms from the literature, including the one presented in [8]. This paper, on the other hand, explores linear channels with longer memory length. Also, based on the results in [7] and our findings, we note that the least complex and most stable MCMC algorithm reported so far is the one proposed in [8]. We thus concentrate on the use of this algorithm and our developments will be in this context. Moreover, we note that the work in [7] is limited to sequential MCMC processing. Hence, the parallel MCMC detection algorithm proposed in this paper is novel.

This paper is organized as follows. In Section II, we introduce the system model, the optimum MAP equalization, and factor graph representation of ISI channels. Section III includes detailed descriptions of the proposed SISO equalizers. Simulation results are presented in Section IV and the conclusions are drawn in Section V.

In this paper, the following notations are used. Vectors are denoted by lowercase bold letters and are in column form. Matrices are denoted by uppercase bold letters. The superscripts T is used to denote matrix or vector transpose. To differentiate between bit and symbol indices, k is consistently used for bits and i is used for the symbols. The notations $P(\cdot)$ and $p(\cdot)$ denote the probability of discrete random variables and the probability density of continuous random variables, respectively.

II. System description

A. System model

We consider the communication system depicted in Fig. 1. A sequence of binary information bits $\{a_k\}$ is first encoded by a channel encoder of rate R. The coded bites are passed through an interleaver with the output $\{b_k\}$. We assume a packetized data transmission system in which each packet consists of KN coded bits. The modulator maps each set of K bits of $\{b_k\}$ to a data symbol x_i and constructs the transmit signal vector $\mathbf{x} = [x_0, x_1, \cdots, x_{N-1}]^{\mathrm{T}}$. This vector is passed through an ISI channel with impulse response $\mathbf{h} = [h_0, h_1, \cdots, h_L]^{\mathrm{T}}$, where L is channel memory. The channel output is the vector $\mathbf{y}$. More details of $\mathbf{y}$ is given later.

At the receiver, SISO equalization and channel decoding are performed in a turbo loop. The SISO equalizer finds the log-likelihood ratio (LLR) of the coded bits, $\{\lambda_{1,k}\}$, using the available information which are the channel impulse response, $\mathbf{h}$, the received signal samples, $\mathbf{y}$, and possibly some extrinsic information from the channel decoder, $\{\lambda^{\mathrm{e}}_{2,k}\}$. The extrinsic (i.e., the new portion of) LLR values from the SISO equalizer given by $\lambda^{\mathrm{e}}_{1,k} = \lambda_{1,k} - \lambda^{\mathrm{e}}_{2,k}$ are passed to the channel decoder that ameliorates the results using the known correlation between the coded bits $\{b_k\}$ that has been induced by the encoder. After a few cycles of the exchange of information between the SISO equalizer and channel decoder, the channel decoder make a final decision on the transmitted uncoded bits $\{a_k\}$.

The channel is characterized by the equation

$$y_n = \sum_{l=0}^{L} h_l x_{n-l} + z_n, \quad \text{for } n = 0, 1, \cdots, N-1, \tag{1}$$

where $\{z_n\}$ is the channel noise, a complex Gaussian random process with zero mean and one-sided variance $\sigma^2 = N_0/2$.

The set of equations (1) can be represented in matrix form as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z} \tag{2}$$

where $\mathbf{y} = [y_0, y_1, \cdots, y_{N+L-1}]^{\mathrm{T}} \in \mathcal{C}^{(N+L)\times 1}$ and $\mathbf{z} = [z_0, z_1, \cdots, z_{N+L-1}]^{\mathrm{T}} \in \mathcal{C}^{(N+L)\times 1}$ are the received signal vector and the channel noise vector, respectively, and

$$\mathbf{H} = \begin{bmatrix} h_0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ 0 & h_L & \cdots & h_0 & \cdots & 0 \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & h_L \end{bmatrix} \tag{3}$$

is the channel matrix. The samples of channel impulse response, $\{h_n\}$, are also complex-valued.

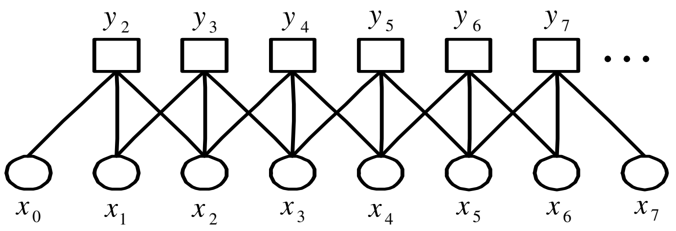

Fig. 2. Factor graph representation of an ISI channel with $L = 2$. Square denotes constraint nodes and circle denotes factor nodes.

B. SISO equalizer

Given $\mathbf{y}$, the extrinsic LLR value of a particular bit b_k is given by

$$\begin{aligned} \lambda_{1,k} &= \ln \frac{P(b_k = 0|\mathbf{y}, \boldsymbol{\lambda}^e_2)}{P(b_k = 1|\mathbf{y}, \boldsymbol{\lambda}^e_2)} \\ &= \ln \frac{\sum_{\mathbf{b}_{-k}} P(b_k = 0, \mathbf{b}_{-k}|\mathbf{y}, \boldsymbol{\lambda}^e_2)}{\sum_{\mathbf{b}_{-k}} P(b_k = 1, \mathbf{b}_{-k}|\mathbf{y}\boldsymbol{\lambda}^e_2)} \end{aligned} \tag{4}$$

where $\mathbf{b}_{-k} = (b_0, \cdots b_{k-1}, b_{k+1}, \cdots, b_{KN-1})$; $b_j \in \{0, 1\}$. Using Bayes' Rule, we can write (4) as

$$\lambda^e_{1,k} = \ln \frac{\sum_{\mathbf{b}_{-k}} p(\mathbf{y}|b_k = 0, \mathbf{b}_{-k}) \prod_{k'} P(b_{k'})}{\sum_{\mathbf{b}_{-k}} p(\mathbf{y}|b_k = 1, \mathbf{b}_{-k}) \prod_{k'} P(b_{k'})} - \lambda^e_{2,k} \tag{5}$$

Computation of each summation in (5) is over all combinations of $\mathbf{b}_{-k}$ which is equal to 2^{KN-1}. For typical values of KN which are in the order of at least few hundreds, this clearly is a prohibitive complexity. By utilizing the trellis structure of the ISI channel, the BCJR algorithm [3] may be used to reduce this complexity significantly. However, even this reduced complexity grows exponentially with the multiplication of the length of the channel, $L + 1$, and the number of bits per symbol, K. We use MCMC to reduce this complexity to a more affordable level. As we will show the complexity of MCMC SISO equalizer grows linearly with $K(L + 1)$.

C. Factor graph representation

The ISI channel model may be represented by a factor graph, [9], as in Fig. 2. The constraint nodes denote the channel equation (1), and the factor nodes denote the transmitting symbols $\{x_i\}$. Accordingly, the sum-product algorithm can be applied for computation of the a posteriori probability terms in (5). Details of the sum-product algorithms for ISI channels can be found in [9].

III. MCMC equalizer

The complexity of the optimum SISO (i.e., MAP) equalizer comes from the exponential growth of the number of combinations of $\mathbf{b}_{-k}$ which leads to 2^{NK-1} summation terms in the numerator and denominator of (5). However, we note that in practice there are always only small subsets of selections of $\mathbf{b}_{-k}$ that contribute to the final results of the summations. Let

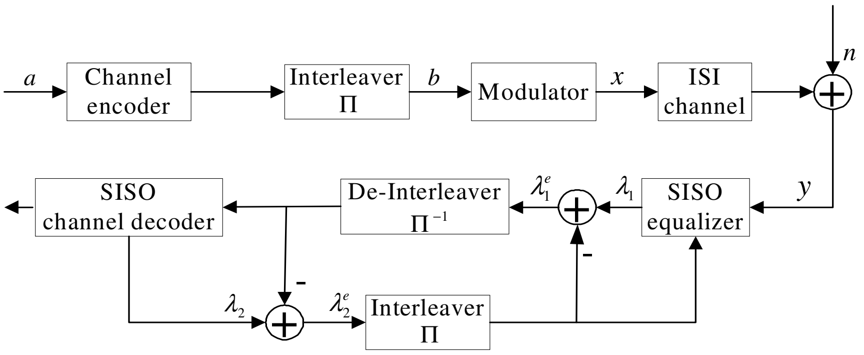

Fig. 1. Turbo equalizer system model. **Changes: channel encoder input: a, Interleaver output: b.**

us call such subsets *important sets* and use $\mathcal{I}_{k,0}$ and $\mathcal{I}_{k,1}$ to denote the important sets associated with the cases of $b_k = 0$ and $b_k = 1$, respectively.

An ideal important set is the set which includes only the significant probability terms under the summation in (5). The MCMC is a search method that finds the desired significant terms by browsing through the choices of $\mathbf{b}_{-k}$ in an efficient manner. We refer to these choices as samples. By increasing the number of samples of MCMC, one can improve the detector performance. In other word, MCMC allows one to trade between the complexity and performance.

A. Gibbs sampler

One widely used method for obtaining the samples in MCMC is the Gibbs sampler. In the context of this paper, the Gibbs sampler examines the bits of $\mathbf{b}_{-k}$ one at a time and, in a probabilistic manner, and selects a value for the bit that on average increases the respective probability terms under the summations in (5).

The Gibbs sampler operates as follows:

```
Generate an initial $\mathbf{b}^{(0)}$
for $n = 1$ to $I$
 generate $b_0^{(n)}$ from distribution
    $P(b_0 = b | b_1^{(n-1)}, b_2^{(n-1)}, \cdots, b_{KN-1}^{(n-1)}, \mathbf{y})$
 generate $d_1^{(n)}$ from distribution
    $P(b_1 = b | b_0^{(n)}, b_2^{(n-1)}, \cdots, b_{KN-1}^{(n-1)}, \mathbf{y})$
    $\vdots$
 generate $b_{KN-1}^{(n)}$ from distribution
    $P(b_{KN-1} = b | b_0^{(n)}, b_1^{(n)}, \cdots, b_{KN-2}^{(n)}, \mathbf{y})$
end for
```

Here, $\mathbf{b}^{(0)}$ is the initial bit sequence, $b_k^{(n)}$ is the k-th bit generated during the n-th iteration, and I is the number of iterations for the Gibbs sampler.

Note that the Gibbs sampler presented above sequentially iterates over all bits within each packet before returning for the next iteration. To facilitate an extension of this sequential algorithm to a more practical parallel processing algorithm, we introduce a factor graph interpretation of the Gibbs sampler from [10].

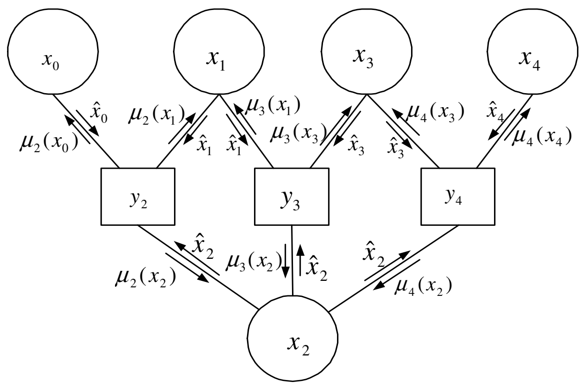

Fig. 3. Gibbs sampling as message passing over factor graph.

According to [10], Gibbs sampling is effectively a message-passing in a factor graph. This may be explained as follows (see Fig. 3 and compare it with Fig. 2):

1) Select a factor node, say x_2 in Fig. 2.
2) The factor nodes $\{x_0, x_1, x_3, x_4\}$ send their samples as outgoing messages to adjacent constraint nodes $\{y_2, y_3, y_4\}$.
3) At the constraint nodes, using the incoming message from $\{x_0, x_1, x_3, x_4\}$, soft messages of x_2 are computed as

$$\begin{aligned} \mu_2(x_2) &= P(x_2 | y_2, x_0, x_1) \\ \mu_3(x_2) &= P(x_2 | y_3, x_1, x_3) \\ \mu_4(x_2) &= P(x_2 | y_3, x_3, x_4) \end{aligned} \tag{6}$$

   and sent to factor node x_2.
4) At factor node x_2, a new sample is selected from the distribution

$$f(x_2) = \frac{\mu_2(x_2)\mu_3(x_2)\mu_4(x_2)}{\sum_{x_2} \mu_2(x_2)\mu_3(x_2)\mu_4(x_2)} \tag{7}$$

   and returned to the constraint nodes.

One may find that these steps are equivalent to the Gibbs sampler procedure presented above when x_is are binary bits. In particular, the set of equations (6) and (7) are equivalent to forming the conditional probabilities in the Gibbs sampler. In the more general case where x_is are symbols, each carrying multiple bits, (6) should be modified to extract marginal probabilities of each bit, and (7) should be replaced by the probability distribution of a bit.

B. Application to Equalization

We use samples from the Gibbs sampler to derive an approximation to the MAP equalizer and call it MCMC equalizer. The MCMC equalizer is implemented in two steps. In the first step, the Gibbs sampler is used to generate the sample sets of $\mathbf{b}_{-k}$, for $k = 0, 1, \cdots, NK-1$. The sample sets form the important sets. In the second step, the *a posteriori* LLR values are computed using the important sets. The details are as follows:

1) Generating the sample sets: In the Gibbs sampler presented above, the selection of the bit b_k is from the conditional distribution

$$P(b_k^{(n)} = b | b_0^{(n)}, \cdots, b_{k-1}^{(n)}, b_{k+1}^{(n-1)}, \cdots, b_{KN-1}^{(n-1)}, \mathbf{y}),$$

for $b \in \{0, 1\}$. Since the transmitter sends the symbols $\{x_i\}$, instead of the coded bits $\{b_k\}$, and

$$\begin{aligned} &P(b_k^{(n)} = b | b_0^{(n)}, \cdots, b_{k-1}^{(n)}, b_{k+1}^{(n-1)}, \cdots, b_{KN-1}^{(n-1)}, \mathbf{y}) \\ &\quad \propto p(\mathbf{y} | b_0^{(n)}, \cdots, b_k^{(n)}, b_{k+1}^{(n-1)}, \cdots, b_{KN-1}^{(n-1)}) P(b_k^{(n)} = b) \\ &\quad = p(\mathbf{y} | x_0^{(n)}, \cdots, x_i^{(n)}, x_{i+1}^{(n-1)}, \cdots, x_{N-1}^{(n-1)}) P(b_k^{(n)} = b), \end{aligned} \tag{8}$$

for implementing of the Gibbs sampler, we use the right-hand side of (8) instead of

$$P(b_k^{(n)} = b | b_0^{(n)}, \cdots, b_{k-1}^{(n)}, b_{k+1}^{(n-1)}, \cdots, b_{KN-1}^{(n-1)}, \mathbf{y}).$$

Moreover, we note that due to finite memory of the channel,

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=0}^{N-1} p(y_i | x_{i-L:i}) \tag{9}$$

where $x_{i-L:i}$ denotes $\{x_{i-L}, x_{i-L+1}, \cdots, x_i\}$. Also, using the channel model of (1), we have

$$p(y_i | x_{i-L:i}) \propto \exp\left(-\frac{1}{2\sigma^2} |y_i - \sum_{l=0}^{L} h_l x_{i-l}|^2 \right). \tag{10}$$

2) Computing the a posteriori *LLR values:* After collecting sufficient samples, the *a posteriori* LLR values are computed as

$$\lambda_{1,k}^e = \ln \frac{\sum_{\mathcal{I}_{k,0}} p(\mathbf{y} | b_k = 0, \mathbf{b}_{-k}) \prod_{k'} P(b_{k'})}{\sum_{\mathcal{I}_{k,1}} p(\mathbf{y} | b_k = 1, \mathbf{b}_{-k}) \prod_{k'} P(b_{k'})} - \lambda_{2,k}^e \tag{11}$$

Note that (11) is similar to (5) except that the summations are over the important sets $\mathcal{I}_{k,0}$ and $\mathcal{I}_{k,1}$ whose size may be much smaller than the set defined by $\mathbf{b}_{-k}$. In most cases, the number of elements in $\mathcal{I}_{k,0}$ and in $\mathcal{I}_{k,1}$ are unequal. However, to have a pair of balanced estimates of the probabilities in the numerator and denominator of the first term on the right-hand side of (11), the number of samples in $\mathcal{I}_{k,0}$ and in $\mathcal{I}_{k,1}$ should be equal. We resolve this problem by combining the samples from $\mathcal{I}_{k,0}$ and in $\mathcal{I}_{k,1}$ and forming a new pair of sample sets with equal number of samples as follows. We form the set $\mathcal{I}_k = \mathcal{I}_{k,0} \cup \mathcal{I}_{k,1}$ and introduce an expanded set $\mathcal{I}_k^{\rm e}$ that contains all the vectors in $\mathcal{I}_k$, as well as the vectors that differ from the vectors in $\mathcal{I}_k$ by only one bit. We then substitute $\mathcal{I}_{k,0}$ and $\mathcal{I}_{k,1}$ by $\mathcal{I}_{k,0}^{\rm e}$ and $\mathcal{I}_{k,1}^{\rm e}$ which are the same size subsets of $\mathcal{I}_k^{\rm e}$.

Strictly speaking, (11) suggests that each sample in the important sets $\mathcal{I}_{k,0}$ and $\mathcal{I}_{k,1}$ has a length of NK bits. However, the bits that interact significantly with a particular bit, say, b_k, are those that are within a limited distance from it. More particularly, if b_k belongs to the symbol x_i, then the samples of the received signal that are affected by b_k are $y_{i-L:i+L}$. So, when computing $\lambda_{1,k}^{\rm e}$, we truncate the important sets $\mathcal{I}_{k,0}$ and $\mathcal{I}_{k,1}$ to include only the bits b_l for $K(i-2L) \leq l \leq K(i+2L)$.

The performance of the MCMC equalizer is dependent on the quality of samples in the important sets. In practice, Gibbs sampler may require many iterations to converge to its stationary distribution. This is called burn-in period. As a result, including the burn-in period in the implementation of the Gibbs sampler may increase the complexity significantly. In [8], it has been shown empirically that the formulas such as (11) still work well if the stationary distribution of the underlying Markov chain is replaced by a uniform distribution over the significant samples (see Fig. 2 of [8]). To obtain samples with this uniform distribution, it has also been noted in [8] that a set of parallel Gibbs samplers with no burn-in period and small number of iterations are more effective than using a single Gibbs sampler with many iterations. We have followed this implementation in this paper.

C. Serial updating and parallel updating

In Gibbs sampler, the bit sequence $\{b_k\}$ is updated sequentially. In terms of factor graph terminology, this is a serial scheduling. From an implementation point of view, this is a sequential processing that runs over each packet multiple times (once for each iteration of the Gibbs sampler). Obviously, sequential processing requires large number of clock cycles, proportional to the packet length, to complete. This in turn means a long processing time or, equivalently, long delay in the detection path. This, of course, is undesirable and should avoided if possible.

We solve the above problem by introducing a parallel processing scheme in which the transmitted symbol vector $\mathbf{x}$ is divided into a number of partitions and each partition is processed separately on a different processor. Fig. 4 depicts an example of such partitioning. From a factor graph point of view, we have divided the original graph into a number of sub-graphs. Note that that there are constraint nodes that share among adjacent sub-graphs. For instance, in Fig. 4, the constraint nodes y_4 and y_5 are common to the two sub-graphs. While running Gibbs samplers, the contents of the factor nodes are being updated. On the other hand, each sub-graph/partition is processed by a different Gibbs sampler. Message is exchanged between adjacent sub-graphs by sending the samples of factor nodes in one sub-graph to the the factor nodes in the other sub-graph through the shared constraint nodes. A pair of Gibbs samplers will collide if they attempt to visit the neighboring bits (equivalent to symbols or factor nodes) simultaneously. To avoid this problem, we assume that the processing of each sub-graph is from left to right and the sub-graphs are synchronized such that they begin simultaneously from the left-side. It should be also noted that to avoid collisions it is necessary to select the length of each sub-graph to be larger than the channel memory, L.

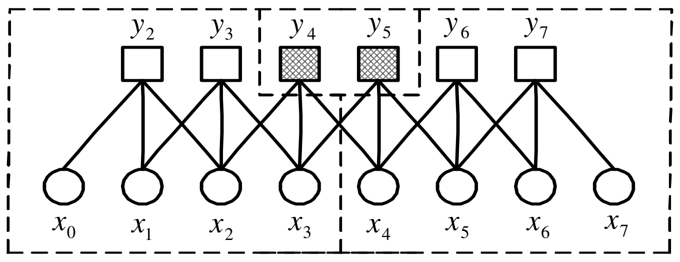

Fig. 4. Factor graph representation of ISI channel with $L = 2$ and partitioned into two sub-graphs. The received signal $y_4, 5_4$ are shared constraint nodes among the two sub-graphs.

IV. Simulation results

In this section, we present simulation results of the proposed MCMC equalizer and make comparisons with the optimal BCJR equalizer. The performance coded AWGN channel is also given as a reference. The basic system setup is as follows. We use a turbo code of rate $1/2$. The coded bit sequence is mapped to a sequence of 8-PSK symbols using gray mapping. The length of the coded bit sequence is 7800 bits, and the length of the symbol sequence is $7800/3 = 2600$ symbols. The soft-in-soft-out optimal MAP decoder is used for turbo decoding. We perform five outer iterations of joint equalization and channel decoding. For each outer iteration, channel equalization is performed once followed by eight inner iterations of turbo decoding. The impulse response of ISI channel is given by

$$\begin{aligned}h[n] &= (2 + 0.4i)\delta[n] + (1.5 + 1.8i)\delta[n-1] + \delta[n-2] \\ &+ (1.2 - 1.3i)\delta[n-3] + (0.8 + 1.6)\delta[n-4]\end{aligned}$$

For the MCMC equalizer, we consider a 10x10 MCMC equalizer (10 parallel Gibbs samplers and 10 iterations per Gibbs sampler) and a 5x5 equalizer, using either a parallel updating schedule, or a serial updating schedule. For the parallel updating schedule, since we have a total of $2600 = 520 \times 5$ received signals, we can break them into 520 segments such that each segment contains only 5 signals. The proposed MCMC equalizer can be applied to these 520 segments in parallel.

Fig. 5 shows the BER performance after 1, 2, 5 iterations of turbo equalization. We note that the MCMC equalizer using the parallel updating schedule achieves similar performance as the one that uses serial updating. The BCJR equalizer performs the best, followed by 10x10 MCMC equalizer and 5x5 MCMC equalizer. After 5 iterations, at BER $= 10^{-4}$, the 10x10 MCMC equalizer is only 0.23 dB worse than the BCJR equalizer. The 5x5 MCMC equalizer performs slightly worse than 10x10 MCMC equalizer by about 0.15 dB. In terms of simulation time, we observe that the system that employs the BCJR equalizer runs about 25 times slower than the system that uses 5x5 MCMC with serial updating, and it runs about 5 times slower than the system that uses the 10x10 MCMC with serial updating. The confirms that the MCMC detector indeed achieves near optimal performance at a greatly reduced complexity.

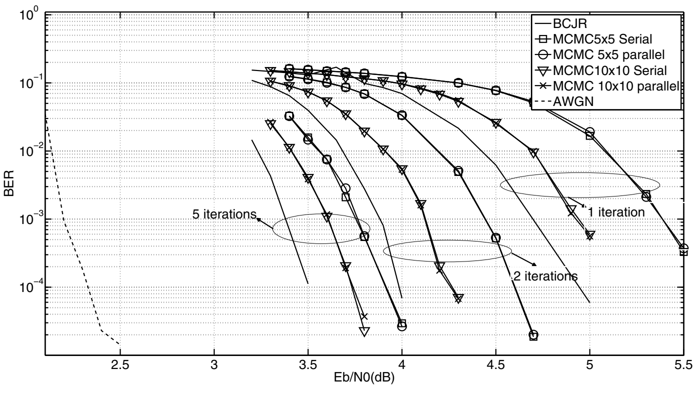

Fig. 5. Performance comparisons of the MCMC equalizer with the optimal equalizer over an ISI channel.

V. Conclusion

In this paper, we propose a novel low-complexity MCMC detector for ISI channels. The proposed detector performs closely to the optimal BCJR detector with a much reduced complexity. A distinguished feature of the MCMC equalizer proposed in this work enables parallel processing, which makes it an attractive candidate for practical implementation. Extensions of this work to multiple-input multiple-output (MIMO) ISI channel is the subject of ongoing research.

References

[1] R. Lucky, "Automatic equalization for digital communication," *Bell System Tech. J.*, pp. 547–588, April 1965.

[2] M. E. Austin, "Decision feedback equalization for digital communication over dispersive channels," MIT Lincoln Lab., Lexington, MA, Tech. Rep., Aug. 1967.

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

[4] G. Colavolpe, G. Ferrari, and R. Raheli, "Reduced-state BCJR-type algorithms," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 848–859, May 2001.

[5] M. Sikora and J. Costello, D. J., "A new SISO algorithm with application to turbo equalization," in *Proc. International Symposium on Information Theory ISIT 2005*, 4–9 Sept. 2005, pp. 2031–2035.

[6] C. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer-Verlag, 1999.

[7] F. M. Kashif, H. Wymeersch, and M. Z. Win, "Monte carlo equalization for nonlinear dispersive satellite channels," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 2, pp. 245–255, February 2008.

[8] B. Farhang-Boroujeny, H. Zhu, and Z. Shi, "Markov chain Monte Carlo algorithms for CDMA and MIMO communication systems," *IEEE Trans. Signal. Process.*, vol. 54, no. 5, pp. 1896–1909, May 2006.

[9] 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.

[10] J. Dauwels, S. Korl, and H. A. Loeliger, "Particle Methods as Message Passing," in *Proc. IEEE Int. Symp. Information Theory(ISIT'06)*, July 2006, pp. 2052–2056.