Soft-Output Detector for Partial Response Channels Using Vector Quantization

Brian M. Kurkoski[1], *Member, IEEE*, Paul H. Siegel[2], *Fellow, IEEE*, and Jack K. Wolf[2], *Life Fellow, IEEE*

[1]University of Electro-Communications, 182-8585 Tokyo, Japan
[2]Center for Magnetic Recording Research, University of California at San Diego, La Jolla, CA 92093-0401 USA

An implementation of the Bahl–Cocke–Jelinek–Raviv algorithm for partial response detection which performs the usual forward and backward state metric recursions and the log-likelihood computation using only lookup tables is proposed. The state metrics are vector quantized, which requires fewer quantization points than the conventional approach of quantizing each state metric individually. This soft-output implementation is suitable for use in partial response systems using turbo equalization.

***Index Terms*—Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm, forward–backward algorithm, partial response channel, turbo equalization.**

I. Introduction

The Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm [1] is often used as the soft-output detector in turbo equalization for partial response channels. When used with an appropriate outer error correcting code, turbo equalization has excellent performance for magnetic recording channels and permits operation at low signal-to-noise ratios (SNRs). However, the soft-output BCJR and related algorithms are difficult to implement effectively at the bit rates and power consumption levels required for hard drives. In conventional implementations, the state metrics are individually or scalar quantized.

In this paper, we consider an implementation of the BCJR algorithm where the state metrics are vector quantized and the operations are performed using lookup tables. The proposed implementation uses off-line computation to generate vector quantizers and lookup tables, and the only run-time operations are table lookups. Lookup tables are readily implemented in both very large-scale integration (VLSI) logic and computer memories.

II. System Description and BCJR Algorithm

The following partial response communication system is assumed. Bits x_t at time t are passed into a partial response channel with impulse response $h(D) = h_0 + h_1 D + \cdots + h_\nu D^\nu$. The noiseless output of the channel is $c_t = \sum_{i=0}^{\nu} h_i x_{t-i}$. An M-state trellis, with input transition labels x and output transition labels c, describes the partial response channel. We consider the combination of the even-mark modulation constraint, which allows ones to occur only in pairs, and the partial response class 1 channel, with $h(D) = 1 + D$; the EMM-PR1 trellis section is shown in Fig. 1. The state metrics of three-state trellises are easily visualized in two dimensions. We also consider the partial response class 2 (PR2) channel, with $h(D) = (1 + D)^2$. Unlike the partial response class 4 channel, the PR2 channel cannot be represented as two simpler interleaved channels. Although these are optical channel models, the proposed implementation is applicable to magnetic recording channels as well.

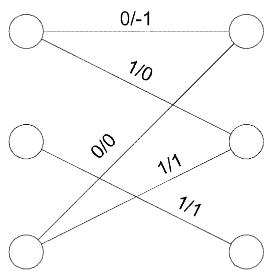

Fig. 1. EMM-PR1 trellis. Trellis labels are input x/output c.

The receiver observes N symbols $y_t = c_t + n_t, t = 1, \ldots, N$, where n_t is additive white Gaussian noise (AWGN) with known variance σ^2. It is assumed that the trellis is in the zero state at $t = 1$ and after $t = N$. The channel SNR is E_s/N_0, where E_s is the average power in c_t and $N_0 = 2\sigma^2$. For the EMM-PR1 channel, the observed symbols y_t are hard limited to $-2 \leq y_t \leq 2$; further, it is assumed that the sequence x_t satisfies the EMM constraint. For the PR2 channel, $c_t \in \{0, 1, 2, 3, 4\}$ and the observed symbols y_t are hard limited to $-1 \leq y_t \leq 5$.

The BCJR algorithm computes the log-likelihood ratio

$$U_t = \log \frac{P\left(x_t = 1 \middle| y_1^N\right)}{P\left(x_t = 0 \middle| y_1^N\right)}.$$

The algorithm has three steps: the forward recursion, the backward recursion, and the computation of the log-likelihood ratio. The forward recursion computes the state metrics $A_t(m)$ for states $m = 0, \ldots, M-1$ at time t. Let m and m' be two distinct states at time t which have transitions to state m'' at time $t+1$. The BCJR transition metric, for a trellis edge with label $c(m, m'')$, is the Euclidean distance $G_t(m, m'') = (y_t - c(m, m''))^2$, in which case the BCJR state metric recursion can be written as

$$A_{t+1}(m'') = \min(A_t(m) + G_t(m, m''), A_t(m') + G_t(m', m'')) - 2\sigma^2 \log(1 + e^{-|A_t(m) + G_t(m,m'') - A_t(m') - G_t(m',m'')|/2\sigma^2}). \quad (1)$$

Digital Object Identifier 10.1109/TMAG.2005.854452

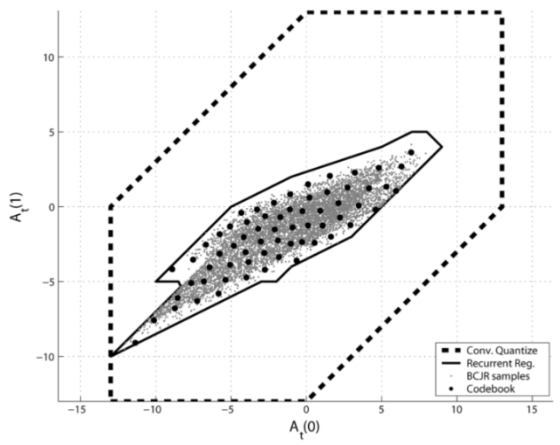

Fig. 2. Recurrent region for EMM-PR1 channel under Viterbi detection and samples for BCJR detection. x axis is $A_t(0)$, and y axis is $A_t(1)$.

In the limit as $\sigma^2 \to 0$, the correction term $2\sigma^2 \log(\cdots)$ goes to zero, and (1) becomes the Viterbi state metric recursion.

The state metrics at time t can be written as a vector $\mathbf{A}_t = (A_t(0), \ldots, A_t(M-1))$. The state metric recursion (1) can be represented as a function f_A: $\mathbf{A}_{t+1} = f_A(\mathbf{A}_t, y_t)$. Similarly, the backward computed state metrics and corresponding recursion are $\mathbf{B}_t = f_B(\mathbf{B}_{t+1}, y_t)$. The computation of the log-likelihood ratio is represented as a function f_U: $U_t = f_U(\mathbf{A}_t, y_t, \mathbf{B}_{t+1})$.

III. Recurrent Region and Vector Quantization

A. Recurrent Region

The recurrent region is the space of reachable state metrics.

Definition: If at time $t = 1$ the state metrics are initialized to $\mathbf{A}_1 = (0, 0, \ldots, 0)$, then the *forward BCJR recurrent region* is the set of state metrics which can be reached by an arbitrary number of forward steps of the BCJR algorithm.

For Viterbi detection of the EMM-PR1 channel, bounds on the recurrent region are known [2] and are plotted in Fig. 2. Also plotted are 10^5 samples of the BCJR algorithm's forward state recursion metric computed using (1). We have observed that most, but not all, samples of the BCJR state metrics fall within the Viterbi algorithm's recurrent region. In Fig. 2, and in what follows, the state metrics are normalized so that the state metric $A_t(M-1)$ is zero.

In a conventional implementation, the state metrics $A_t(m)$ are individually quantized. If the maximum difference between any two state metrics is bounded by Δ, then it is sufficient to quantize the state metrics in the range $(0, \Delta)$. In Fig. 2, this region of conventional quantization is shown ($\Delta = 13$ for the EMM-PR1 channel). State metrics outside the recurrent region are quantized in the conventional implementation, even though they can never occur.

B. Vector Quantization

Consider vector quantization of the state metrics $\mathbf{A}_t$ and $\mathbf{B}_t$ and the scalar quantization of the received symbol y_t and the output U_t. A vector quantizer Q_A is a mapping from a k-dimensional space to a set $\mathcal{C}_A$ containing I_A codepoints, where $\mathcal{C}_A = \{a_1, \ldots, a_{I_A}\}$ and $a_i \in \mathbb{R}^k$ for $i \in \mathcal{I}_A = \{1, 2, \ldots, I_A\}$. Let $\alpha_t \in \mathcal{I}_A$ denote the index value at time t. The encoder function is $\alpha_t = \mathcal{E}_A(\mathbf{A}_t)$. Correspondingly, $\hat{\mathbf{A}}_t$ represents one of the codepoints a_i at time t; that is, $\alpha_t = i$ is the same as $\hat{\mathbf{A}}_t = a_i$. The distortion measure is Euclidean distance

$$D(\mathbf{A}_t, \hat{\mathbf{A}}_t) = \sum_{m=0}^{k-1} (A_t(m) - \hat{A}_t(m))^2.$$

Although nominally the dimension of the quantizer is $k = M$, a $k = M - 1$ dimension quantizer is sufficient because the metric for state $M - 1$ is normalized to zero.

A similar but independent quantizer for the backward state metrics is required. The vector quantizer Q_B has a codebook $\mathcal{C}_B$ with size I_B and codepoints b_i. We let $\beta_t \in \mathcal{I}_B = \{1, \ldots, I_B\}$ denote the index at time t and let $\hat{\mathbf{B}}_t \in \mathcal{C}_B$ denote the codepoint at time t. The encoder function is $\beta_t = \mathcal{E}_B(\mathbf{B}_t)$.

Quantization of the received symbols y_t is required. The scalar quantizer Q_Y has a codebook $\mathcal{C}_Y$ with size I_Y and codepoints $\bar{y}_i$. We let $\eta_t \in \mathcal{I}_Y = \{1, \ldots, I_Y\}$ denote the index at time t and let $\hat{y}_t \in \mathcal{C}_Y$ denote the codepoint at time t. The encoder function is $\eta_t = \mathcal{E}_Y(y_t)$.

Also, the output U_t is quantized. The scalar quantizer Q_U has a codebook $\mathcal{C}_U$ with size I_U and codepoints u_i. We let $\omega_t \in \mathcal{I}_U = \{1, \ldots, I_U\}$ denote the index at time t and let $\hat{U}_t \in \mathcal{C}_U$ denote the codepoint at time t. The encoder function is $\omega_t = \mathcal{E}_U(U_t)$, and the decoder function is $\hat{u}_t = \mathcal{D}_U(\omega_t)$.

C. Minimax Lloyd Algorithm

The quantizer codebooks $\mathcal{C}_A$ and $\mathcal{C}_B$ are designed using a variation of the Lloyd algorithm [3], where the centroid condition is replaced by a minimize-the-maximum (minimax) condition. The objective of this variation is to produce codepoints which are equally spaced where there is training data and no codepoints elsewhere. This algorithm generates I_A codepoints using a training set of N_{ts} vectors $\mathbf{A}_1, \ldots, \mathbf{A}_{N_{\text{ts}}}$.

Minimax Lloyd Algorithm:

1) *Initialize*. Choose an initial codebook by randomly selecting I_A points from the training set.

2) *Nearest Neighbor*. Partition the training set into I_A nonoverlapping regions R_i such that

$$R_i = \{\mathbf{A}_k \colon D(\mathbf{A}_k, a_i) < D(\mathbf{A}_k, a_j), \text{for all } j \neq i\}.$$

3) *Minimax*. Choose a new codepoint a_i for R_i

$$a_i = \arg\min_x \max_{\mathbf{A}_k \in R_i} (D(\mathbf{A}_k, x)).$$

4) Repeat Steps 2) and 3) for a fixed number of iterations.

The codebook generated by the minimax Lloyd algorithm for the EMM-PR1 channel is shown in Fig. 2, generated using $N_{\text{ts}} = 10^4, I_A = 64, E_s/N_0 = 1$ dB. The codebook stabilized after 50 iterations.

IV. Lookup Table Implementation

The proposed implementation has two stages: a setup stage and a detection stage. In the setup stage, performed off-line, a vector quantizer is selected, and this is used to build three lookup tables: $f_{A,Q}, f_{B,Q}$, and $f_{U,Q}$ for the forward recursion, backward recursion, and computation of the log-likelihood ratio, respectively. In the detection stage, which is the run-time portion, these lookup tables are used to perform the detection operation, as normally would be performed by the BCJR algorithm.

Lookup Table Implementation—Setup:

1) Select vector quantizers Q_A and Q_B using a technique such as the minimax Lloyd algorithm.
2) Build the forward lookup table. For each $i \in \mathcal{I}_A$ and for each $j \in \mathcal{I}_Y$:
   a) Compute $\mathbf{A} = f_A(a_i, \bar{y}_j)$.
   b) Make the lookup table entry $f_{A,Q}(i,j) = \mathcal{E}_A(\mathbf{A})$.
3) Build the backward lookup table. For each $k \in \mathcal{I}_B$ and for each $j \in \mathcal{I}_Y$:
   a) Compute $\mathbf{B} = f_B(b_k, \bar{y}_j)$.
   b) Make the lookup table entry $f_{B,Q}(k,j) = \mathcal{E}_B(\mathbf{B})$.
4) Build the log-likelihood ratio output lookup table. For each $i \in \mathcal{I}_A$, each $j \in \mathcal{I}_Y$, and each $k \in \mathcal{I}_B$:
   a) Compute $u = f_U(a_i, \bar{y}_j, b_k)$.
   b) Make the lookup table entry $f_{Q,U}(i,j,k) = \mathcal{D}_U(\mathcal{E}_U(u))$.

Lookup Table Implementation—Detection: Assume that the input y_t has been quantized to the sequence $\eta_t = \mathcal{E}_Y(y_t)$. Initialize $\alpha_1 = \mathcal{E}_A((0,\infty,\ldots,\infty))$ and $\beta_{N+1} = \mathcal{E}_B((0,\infty,\ldots,\infty))$.

1) Forward recursion. For each $t \in \{1,\ldots,N-1\}$, lookup $\alpha_{t+1} = f_{A,Q}(\alpha_t, \eta_t)$.
2) Backward recursion. For each $t \in \{N,\ldots,2\}$, lookup $\beta_t = f_{B,Q}(\beta_{t+1}, \eta_t)$.
3) Log-likelihood output. For each $t \in \{1,\ldots,N\}$, lookup $\hat{U}_t = f_{U,Q}(\alpha_t, \eta_t, \beta_{t+1})$.

The sequence $\hat{U}_t$ is the lookup-table implementation's quantized output.

A. Simulation Results

The performance of the look-up table implementation was simulated on the PR2 channel at SNRs of 2 and 5 dB. Fig. 3 shows the mean-squared quantization error as a function of the number of bits of resolution of the state metrics. The reduction in state metric storage from that of the conventional implementation (also shown) is between 4 and 5 bits per state. For the lookup-table implementation, the vector quantizers Q_A, Q_B were generated using the minimax Lloyd algorithm, with $I_A = I_B$. The quantizer Q_Y has $I_Y = 32$ points, uniformly spaced between -1 and 5. The quantizer Q_U has $I_U = 64$ points, uniformly spaced between -15 and 15.

B. Lookup Table Size

As the number of codepoints I_A and I_B increases, quantization error decreases, and the accuracy of the lookup-table implementation increases. Accordingly, the size of the lookup tables $f_{A,Q}, f_{B,Q}$ and $f_{U,Q}$ increases. Table I shows the general size of the lookup tables and lists some examples for the PR2 channel, with $I_Y = 32$ and $I_U = 64$.

The lookup table $f_{U,Q}$ has three arguments and is much larger than $f_{A,Q}$ and $f_{B,Q}$, which have two arguments each. A partial implementation of the lookup-table algorithm is possible which uses lookup-tables for the forward and backward recursion only and then computes the function f_U in the usual fashion using the state metrics which have been decoded from their values α_t, β_{t+1}.

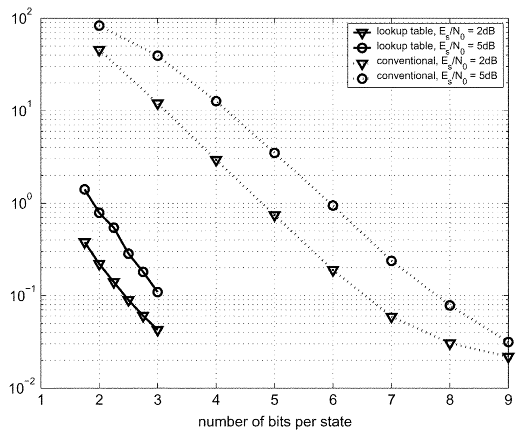

Fig. 3. Mean-squared error in U_t for the PR2 channel detector.

TABLE I
LOOKUP TABLE SIZES FOR PR2 CHANNEL. kB DENOTES 8192 BITS

I_A, I_B	$f_{A,Q}, f_{B,Q}$	$f_{U,Q}$
—	$I_Y I_A \log_2 I_A$	$I_Y I_A I_B \log_2 I_U$
128	3.5 kB ($2^{12} \times 7$)	384 kB ($2^{19} \times 6$)
256	8 kB ($2^{13} \times 8$)	1536 kB ($2^{21} \times 6$)
512	18 kB ($2^{14} \times 9$)	6144 kB ($2^{23} \times 6$)
1024	40 kB ($2^{15} \times 10$)	24576 kB ($2^{25} \times 6$)

V. CONCLUSION

We have proposed a lookup-table implementation of the BCJR algorithm for partial response channels. For the PR2 channel considered, vector quantization results in a substantial reduction in the storage used for the state metrics, and the lookup tables are a practical size. Lookup table operations, both in VLSI and computer simulations, can be much faster than conventional state-by-state computations.

ACKNOWLEDGMENT

This work was supported by the Ministry of Education, Science, Sports and Culture under Grant-in-Aid for JSPS Fellows, number 00004908, 2004 and by NSF Grant CCR-0219582, part of the Information Technology Research Program.

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. 20, no. 2, pp. 284–287, Mar. 1974.
[2] R. Calderbank, P. Fishburn, and P. Siegel, "State-space characterization of Viterbi detector path metric differences," in *26th Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, Oct. 1992, pp. 940–944.
[3] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer, 1992.

Manuscript received February 7, 2005.