We can recover j from $g(j)$ as follows:

$$j_n = g_n^j$$

$$j_k = j_{k+1} + g_k^j \bmod 2$$

which gives

$$j_{n-1} = g_n^j + g_{n-1}^j$$

$$j_{n-2} = g_n^j + g_{n-1}^j + g_{n-2}^j, \cdots.$$

Thus

$$j_k = \sum_{m=k}^{n} g_m^j \bmod 2.$$

We know that the integers i and i', where $0 \le i < 2^{n-1}$ and $i' = i + 2^{n-1}$, differ in only one digit, i.e., $i_n = 0, i_n' = 1$, $i_k = i_k', k = 1,2,\cdots, n-1$. Hence

$$g_k^i = i_k + i_{k+1} = g_k^{i'}, \qquad \text{if } k \le n-2. \tag{2}$$

Furthermore,

$$g_n^i + g_{n-1}^i = i_{n-1}$$

while

$$g_n^{i'} + g_{n-1}^{i'} = i_{n-1}' = g_n^i + g_{n-1}^i. \tag{3}$$

We have thus shown that, if we add together the first two columns of a 2^n-level Gray code and copy the remaining $n-2$ columns, the resulting $n-1$ columns contain two identical parts. It remains to be proved that each half is a 2^{n-1}-level Gray code. We denote the latter by $G(i)$, $0 \le i \le 2^{n-1} - 1$. Then

$$G_{n-1} = i_{n-1}, \qquad G_k = i_k + i_{k+1}, \qquad 0 \le k \le n-2.$$

The previous are identical to the expressions (2) and (3). Thus the $n-1$ columns do consist of two repetitions of 2^{n-1}-level Gray code. Now if we combine the first two columns again, we reduce each 2^{n-1}-level Gray code into two 2^{n-2}-level Gray codes, or, the complete array into four 2^{n-2}-level Gray codes. This can continue until we have only m columns, which would be 2^{n-m} repetitions of 2^m-level Gray code. We have thus derived the lemma.

References

[1] M. Gardner, "Mathematical games," *Sci. Amer.*, vol. 227, pp. 106–109, Aug. 1972.

Correction to "On the Error Probability for a Class of Binary Recursive Feedback Strategies"

J. PIETER M. SCHALKWIJK AND KAREL A. POST

In the above paper[1], p. 499, (2) should have read

$$p_{n+1}(\theta) = \begin{cases} \dfrac{(1-y_{n+1})p + y_{n+1}q}{(1-y_{n+1})[aq + (1-a)p] + y_{n+1}[(1-a)q + ap]} p_n(\theta), & \text{for } \theta > a_n \\[2ex] \dfrac{(1-y_{n+1})q + y_{n+1}p}{(1-y_{n+1})[aq + (1-a)p] + y_{n+1}[(1-a)q + ap]} p_n(\theta), & \text{for } \theta < a_n. \end{cases} \tag{2}$$

Manuscript received September 14, 1973.
The authors are with the Technological University, Eindhoven, The Netherlands.
[1] J. P. M. Schalkwijk and K. A. Post, *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 498–511, July 1973.

On the right side of p. 505, the fifth and sixth line from the bottom, the lower error exponent $\bar{E}^-(R)$ is valid for the 1 output and the upper error exponent $\bar{E}^+(R)$ for the 0 output.

Acknowledgment

The authors want to thank Dr. J. C. Tiernan for pointing out the mistake in (2).

Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate

L. R. BAHL, J. COCKE, F. JELINEK, AND J. RAVIV

Abstract—**The general problem of estimating the *a posteriori* probabilities of the states and transitions of a Markov source observed through a discrete memoryless channel is considered. The decoding of linear block and convolutional codes to minimize symbol error probability is shown to be a special case of this problem. An optimal decoding algorithm is derived.**

I. Introduction

The Viterbi algorithm is a maximum-likelihood decoding method which minimizes the probability of word error for convolutional codes [1], [2]. The algorithm does not, however, necessarily minimize the probability of symbol (or bit) error. In this correspondence we derive an optimal decoding method for linear codes which minimizes the symbol error probability.

We first tackle the more general problem of estimating the *a posteriori* probabilities (APP) of the states and transitions of a Markov source observed through a noisy discrete memoryless channel (DMC). The decoding algorithm for linear codes is then shown to be a special case of this problem.

The algorithm we derive is similar in concept to the method of Chang and Hancock [3] for removal of intersymbol interference. Some work by Baum and Petrie [4] is also relevant to this problem. An algorithm similar to the one described in this correspondence was also developed independently by McAdam *et al.* [5].

II. The General Problem

Consider the transmission situation of Fig. 1. The source is assumed to be a discrete-time finite-state Markov process. The M distinct states of the Markov source are indexed by the integer m, $m = 0,1,\cdots, M-1$. The state of the source at time t is denoted by S_t and its output by X_t. A state sequence of the source extending from time t to t' is denoted by $S_t^{t'} = S_t, S_{t+1}, \cdots, S_{t'}$, and the corresponding output sequence is $X_t^{t'} = X_t, X_{t+1}, \cdots, X_{t'}$.

The state transitions of the Markov source are governed by the transition probabilities

$$p_t(m \mid m') = \Pr\{S_t = m \mid S_{t-1} = m'\}$$

and the output by the probabilities

$$q_t(X \mid m', m) = \Pr\{X_t = X \mid S_{t-1} = m'; S_t = m\}$$

where X belongs to some finite discrete alphabet.

Manuscript received July 27, 1972; revised July 23, 1973. This paper was presented at the 1972 International Symposium on Information Theory, Asilomar, Calif. January 1972.
L. R. Bahl and J. Cocke are with the IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y.
F. Jelinek is with the IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y., on leave from Cornell University, Ithaca, N.Y.
J. Raviv was with the IBM Thomas J. Watson Research Center., Yorktown Heights, N.Y. He is now at the IBM Scientific Center, Haifa, Israel.

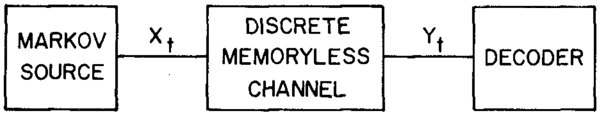

Fig. 1. Schematic diagram of transmission system.

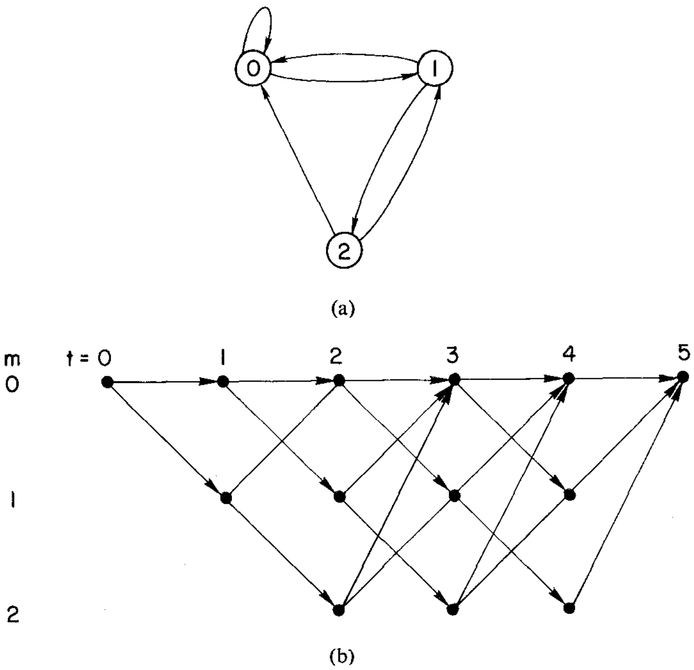

Fig. 2. (a) State transition diagram for 3-state Markov source. (b) Trellis diagram for source of Fig. 2(a).

The Markov source starts in the initial state $S_0 = O$, and produces an output sequence X_1^τ ending in the terminal state $S_\tau = O$. X_1^τ is the input to a noisy DMC whose output is the sequence $Y_1^\tau = Y_1, Y_2, \cdots, Y_\tau$. The transition probabilities of the DMC are defined by $R(\cdot \mid \cdot)$ so that for all $1 \leq t \leq \tau$

$$\Pr \{Y_1^t \mid X_1^t\} = \prod_{j=1}^{t} R(Y_j \mid X_j).$$

The objective of the decoder is to examine Y_1^τ and estimate the APP of the states and transitions of the Markov source, i.e., the conditional probabilities

$$\Pr \{S_t = m \mid Y_1^\tau\} = \Pr \{S_t = m; Y_1^\tau\} / \Pr \{Y_1^\tau\} \tag{1}$$

and

$$\Pr \{S_{t-1} = m'; S_t = m \mid Y_1^\tau\} = \Pr \{S_{t-1} = m'; S_t = m; Y_1^\tau\} / \Pr \{Y_1^\tau\}. \tag{2}$$

A graphical interpretation of the problem is quite useful. A time-invariant Markov source is generally represented by a state transition diagram of the type in Fig. 2(a). The nodes are the states and the branches represent the transitions having nonzero probabilities. If we index the states with both the time index t and state index m, we get the "trellis" diagram of Fig. 2(b). The trellis diagram shows the time progression of the state sequences. For every state sequence S_1^τ there is a unique path through the trellis diagram, and vice versa.

If the Markov source is time variant, then we can no longer represent it by a state-transition diagram; however, it is obvious that we can construct a trellis for its state sequences.

Associated with each node in the trellis is the corresponding APP $\Pr \{S_t = m \mid Y_1^\tau\}$ and associated with each branch in the trellis is the corresponding APP $\Pr \{S_{t-1} = m'; S_t = m \mid Y_1^\tau\}$. The objective of the decoder is to examine Y_1^τ and compute these APP.

For ease of exposition, it is simpler to derive the joint probabilities

$$\lambda_t(m) = \Pr \{S_t = m; Y_1^\tau\}$$

and

$$\sigma_t(m',m) = \Pr \{S_{t-1} = m'; S_t = m; Y_1^\tau\}.$$

Since, for a given Y_1^τ, $\Pr \{Y_1^\tau\}$ is a constant, we can divide $\lambda_t(m)$ and $\sigma_t(m',m)$ by $\Pr \{Y_1^\tau\}$ ($= \lambda_\tau(0)$, which is available from the decoder) to obtain the conditional probabilities of (1) and (2). Alternatively, we can normalize $\lambda_t(m)$ and $\sigma_t(m',m)$ to add up to 1 to obtain the same result. We now derive a method for obtaining the probabilities $\lambda_t(m)$ and $\sigma_t(m',m)$.

Let us define the probability functions

$$\alpha_t(m) = \Pr \{S_t = m; Y_1^t\}$$

$$\beta_t(m) = \Pr \{Y_{t+1}^\tau \mid S_t = m\}$$

$$\gamma_t(m',m) = \Pr \{S_t = m; Y_t \mid S_{t-1} = m'\}.$$

Now

$$\begin{aligned} \lambda_t(m) &= \Pr \{S_t = m; Y_1^t\} \cdot \Pr \{Y_{t+1}^\tau \mid S_t = m; Y_1^t\} \\ &= \alpha_t(m) \cdot \Pr \{Y_{t+1}^\tau \mid S_t = m\} \\ &= \alpha_t(m) \cdot \beta_t(m). \end{aligned} \tag{3}$$

The middle equality follows from the Markov property that if S_t is known, events after time t do not depend on Y_1^t.

Similarly,

$$\begin{aligned} \sigma_t(m',m) &= \Pr \{S_{t-1} = m'; Y_1^{t-1}\} \cdot \Pr \{S_t = m; Y_t \mid S_{t-1} = m'\} \\ &\quad \cdot \Pr \{Y_{t+1}^\tau \;\; S_t = m\} \\ &= \alpha_{t-1}(m') \cdot \gamma_t(m',m) \cdot \beta_t(m). \end{aligned} \tag{4}$$

Now for $t = 1,2,\cdots,\tau$

$$\begin{aligned} \alpha_t(m) &= \sum_{m'=0}^{M-1} \Pr \{S_{t-1} = m'; S_t = m; Y_1^t\} \\ &= \sum_{m'} \Pr \{S_{t-1} = m'; Y_1^{t-1}\} \cdot \Pr \{S_t = m; Y_t \mid S_{t-1} = m'\} \\ &= \sum_{m'} \alpha_{t-1}(m') \cdot \gamma_t(m',m). \end{aligned} \tag{5}$$

Again, the middle equality follows from the fact that events after time $t - 1$ are not influenced by Y_1^{t-1} if S_{t-1} is known. For $t = 0$ we have the boundary conditions

$$\alpha_0(0) = 1, \text{ and } \alpha_0(m) = 0, \qquad \text{for } m \neq 0. \tag{6}$$

Similarly, for $t = 1,2,\cdots, \tau - 1$

$$\begin{aligned} \beta_t(m) &= \sum_{m'=0}^{M-1} \Pr \{S_{t+1} = m'; Y_{t+1}^\tau \mid S_t = m\} \\ &= \sum_{m'} \Pr \{S_{t+1} = m'; Y_{t+1} \mid S_t = m\} \cdot \Pr \{Y_{t+2}^\tau \mid S_{t+1} = m'\} \\ &= \sum_{m'} \beta_{t+1}(m') \cdot \gamma_{t+1}(m,m'). \end{aligned} \tag{7}$$

The appropriate boundary conditions are

$$\beta_\tau(0) = 1, \text{ and } \beta_\tau(m) = 0, \qquad \text{for } m \neq 0. \tag{8}$$

Relations (5) and (7) show that $\alpha_t(m)$ and $\beta_t(m)$ are recursively obtainable. Now

$$\begin{aligned}\gamma_t(m',m) &= \sum_X \Pr\{S_t = m \mid S_{t-1} = m'\} \\ &\quad \cdot \Pr\{X_t = X \mid S_{t-1} = m', S_t = m\} \cdot \Pr\{Y_t \mid X\} \\ &= \sum_X p_t(m \mid m') \cdot q_t(X \mid m', m) \cdot R(Y_t, X) \qquad (9)\end{aligned}$$

where the summation in (9) is over all possible output symbols X.

We can now outline the operation of the decoder for computing $\lambda_t(m)$ and $\sigma_t(m',m)$.

1) $\alpha_0(m)$ and $\beta_\tau(m)$, $m = 0,1,\cdots, M-1$ are initialized according to (6) and (8).

2) As soon as Y_t is received, the decoder computes $\gamma_t(m',m)$ using (9) and $\alpha_t(m)$ using (5). The obtained values of $\alpha_t(m)$ are stored for all t and m.

3) After the complete sequence Y_1^τ has been received, the decoder recursively computes $\beta_t(m)$ using (7). When the $\beta_t(m)$ have been computed, they can be multiplied by the appropriate $\alpha_t(m)$ and $\gamma_t(m',m)$ to obtain $\lambda_t(m)$ and $\sigma_t(m',m)$ using (3) and (4).

We now discuss the application of this algorithm to the decoding of linear codes.

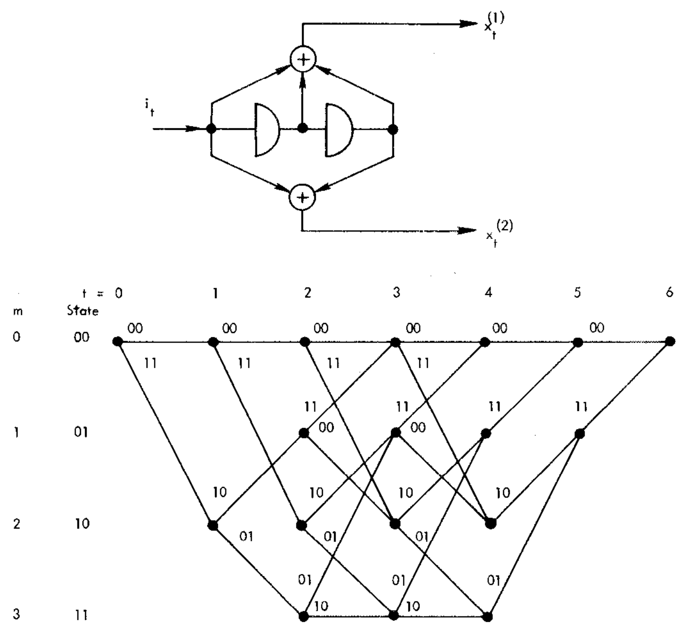

Fig. 3. Rate-1/2 encoder and its trellis diagram.

III. Application to Convolutional Codes

Consider a binary rate k_0/n_0 convolutional encoder of overall constraint length $k_0\nu$. The input to the encoder at time t is the block $I_t = (i_t^{(1)}, i_t^{(2)}, \cdots, i_t^{(k_0)})$ and the corresponding output is $X_t = (x_t^{(1)}, \cdots, x_t^{(n_0)})$. The encoder can be implemented by k_0 shift registers, each of length ν, and the state of the encoder is simply the contents of these registers, i.e., the ν most recent input blocks. Representing the state as a $k\nu$-tuple, we have

$$S_t = (s_t^{(1)}, s_t^{(2)}, \cdots, s_t^{(k_0\nu)}) = (I_t, I_{t-1}, \cdots, I_{t-\nu+1}). \qquad (10)$$

By convention, the encoder starts in state $S_0 = \boldsymbol{O}$. An information sequence $\boldsymbol{I}_1^T$ is the input to the encoder, followed by ν blocks of all-zero inputs, i.e., by $\boldsymbol{I}_{T+1}^\tau = \boldsymbol{O}, \boldsymbol{O}, \cdots, \boldsymbol{O}$ where $\tau = T + \nu$, causing the encoder to end in state $S_\tau = \boldsymbol{O}$. The trellis structure of such a convolutional code is well known [2] and we assume that the reader is familiar with it. As an example, we illustrate in Fig. 3 a rate-$\frac{1}{2}$ code with $\nu = 2$ and its trellis diagram for $\tau = 6$. The transition probabilities $p_t(m \mid m')$ of the trellis are governed by the input statistics. Generally, we assume all input sequences equally likely for $t \le T$, and since there are 2^{k_0} possible transitions out of each state, $p_t(m \mid m') = 2^{-k_0}$ for each of these transitions. For $t > T$, only one transition is possible out of each state, and this has probability 1. The output X_t is a deterministic function of the transition so that, for each transition, there is a $0-1$ probability distribution $q_t(X \mid m',m)$ over the alphabet of binary n-tuples. For time-invariant codes $q_t(\cdot \mid \cdot)$ is independent of t. If the output sequence is sent over a DMC with symbol transition probabilities $r(\cdot \mid \cdot)$, the derived block transition probabilities are

$$R(Y_t \mid X_t) = \prod_{j=1}^{n_0} r(y^{(j)} \mid x_t^{(j)})$$

where $Y_t = (y_t^{(1)}, \cdots, y_t^{(n_0)})$ is the block received by the receiver at time t. For instance, in a BSC with crossover probability p_c

$$R(Y_t \mid X_t) = (p_c)^d (1 - p_c)^{n-d}$$

where d is the Hamming distance between X_t and Y_t.

To minimize the symbol probability of error, we must determine the most likely input digits $i_t^{(j)}$ from the received sequence $\boldsymbol{Y}_1^\tau$. We assume that the $\lambda_t(m)$ have been computed as shown in the previous section. Let $A_t^{(j)}$ be the set of states S_t such that $s_t^{(j)} = 0$. Note that $A_t^{(j)}$ is not dependent on t. Then from (10) we have

$$s_t^{(j)} = i_t^{(j)}, \qquad j = 1,2,\cdots,k_0$$

which implies

$$\Pr\{i_t^{(j)} = 0; \boldsymbol{Y}_1^\tau\} = \sum_{S_t \in A_t^{(j)}} \lambda_t(m).$$

Normalizing by $\Pr\{\boldsymbol{Y}_1^\tau\} = \lambda_\tau(0)$ we have

$$\Pr\{i_t^{(j)} = 0 \mid \boldsymbol{Y}_1^\tau\} = \frac{1}{\lambda_\tau(0)} \sum_{S_t \in A_t^{(j)}} \lambda_t(m).$$

We decode $i_t^{(j)} = 0$ if $\Pr\{i_t^{(j)} = 0 \mid \boldsymbol{Y}_1^\tau\} \ge 0.5$, otherwise $i_t^{(j)} = 1$.

Sometimes it is of interest to determine the APP of the encoder output digits, i.e., $\Pr\{x_t^{(j)} = 0 \mid \boldsymbol{Y}_1^\tau\}$. One instance where such probabilities are needed is bootstrap hybrid decoding [6]. Let $B_t^{(j)}$ be the set of transitions $S_{t-1} = m' \to S_t = m$ such that the jth output digit $x_t^{(j)}$ on that transition is 0. $B_t^{(j)}$ is independent of t for time-invariant codes. Then

$$\Pr\{x_t^{(j)} = 0; \boldsymbol{Y}_1^\tau\} = \sum_{(m',m) \in B_t^{(j)}} \sigma_t(m',m)$$

which can be normalized to give $\Pr\{x_t^{(j)} = 0 \mid \boldsymbol{Y}_1^\tau\}$. We can obtain the probability of any event that is a function of the states by summing the appropriate $\lambda_t(m)$; likewise, the $\sigma_t(m',m)$ can be used to obtain the probability of any event which is a function of the transitions.

Unfortunately, the algorithm requires large storage and considerable computation. All the values of $\alpha_t(m)$ must be stored, which requires roughly $2^{k\nu_0} \cdot \tau$ storage locations. The storage size grows exponentially with constraint length and linearly with block length. The number of computations in determining the $\alpha_t(m)$ (or $\beta_t(m)$) for each t are $M \cdot 2^{k_0}$ multiplications and M additions of 2^{k_0} numbers each. The computation of the $\gamma_t(m',m)$ is quite simple and in practice is most easily accomplished by a

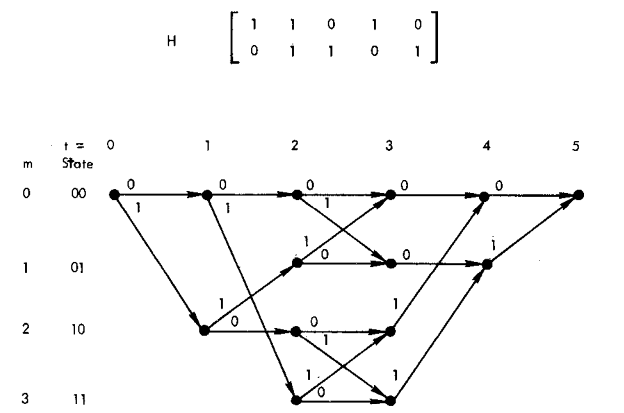

Fig. 4. Parity check matrix and trellis diagram for (5,3) block code.

table lookup. For this reason it is easier to recompute the $\gamma_t(m',m)$ in step 3) rather than to save them from step 2). Computing $\lambda_t(m)$ requires M multiplications for each t and computing the APP of the input digits requires $k_0M/2$ additions. In comparison, the Viterbi algorithm requires the calculation of a quantity essentially similar to $\gamma_t(m',m)$ with $M \cdot 2^{k_0}$ additions and $M2^{k_0}$-way compares for each t. In view of the complexity of the algorithm, it is practical only for short constraint lengths and short block lengths.

IV. Application to Block Codes

The results of Section II can be applied to any code for which a coding trellis can be drawn. We now show how a trellis may be obtained for a linear block code.

Let H be the parity check matrix of a linear (n,k) code, and let h_i, $i = 1,2,\cdots,n$ be the column vectors of H. Let $C = (c_1,c_2,\cdots,c_n)$ be a codeword. We define the states S_t, $t = 0,1,\cdots,n$ pertaining to C as follows:

$$S_0 = O$$

and

$$S_t = S_{t-1} + c_t h_t = \sum_{j=1}^{t} c_i h_i, \qquad t = 1,2,\cdots,n. \tag{11}$$

Obviously, $S_n = O$ for all codewords and the current state S_t is a function of the preceding state S_{t-1} and the current input c_t.

Equation (11) can be used to draw a trellis diagram for a block code with at most 2^r states at each level where $r = n - k$. Each transition is labeled with the appropriate codeword symbol c_t. As an example, a trellis for a block code with

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

is shown in Fig. 4. The structure of the trellis is irregular in comparison to the trellis of a convolutional code, since a block code is equivalent to a time-varying Markov source whereas a convolutional code is a stationary Markov source.

Forney (in a private communication) has pointed out that the number of states needed in the trellis can be reduced to less than 2^r by rearrangement of the code bits. The interesting question of what is the minimum number of states needed is not dealt with here.

The algorithm derived here shows that any parity check code with r parity bits can be decoded with complexity $\sim 2^r$ on an arbitrary memoryless channel. This result had previously only been known for the BSC (using syndromes and table-lookup decoding).

V. Comments and Generalizations

A brute-force approach to minimizing word or symbol error probability would work as follows: given the received sequence Y_1^τ we could compute the APP $\Pr\{X_1^\tau \mid Y_1^\tau\}$ for each codeword X_1^τ. To minimize word error probability, we would pick the codeword having maximum value of $\Pr\{X_1^\tau \mid Y_1^\tau\}$ among all codewords. To minimize the symbol error probability of the jth input digit, we compute $\Sigma \Pr\{X_1^\tau \mid Y_1^\tau\}$, where the sum is over all codewords having jth input digit 0; if this sum ≥ 0.5, we decode the jth input digit as 0. In the case of linear codes we can avoid the calculation of $\Pr\{X_1^\tau \mid Y_1^\tau\}$ for each possible codeword by taking advantage of the state structure of the code. The complexity of the brute-force method is proportional to the number of codewords, i.e., $\sim 2^k$. In convolutional codes $k = k_0T \gg k_0\nu$ which makes the trellis decoding approach attractive. In block codes, the trellis method is advantageous as long as $r < k$, i.e., for high-rate codes.

The algorithm derived in this correspondence cannot be considered as an attractive alternative to Viterbi decoding, because of its increased complexity. Even though Viterbi decoding is not optimal in the sense of bit error rate, in most applications of interest the performance of both algorithms would be effectively identical. The main virtue of the algorithm is in the fact that the APP of the information and channel digits are obtained, which can be useful in applications such as bootstrap decoding [6].

Many interesting generalizations of the algorithm are possible. We point out a few. First, the restriction that the starting and terminal states of the source be known can be removed by changing the initial conditions for $\alpha_0(m)$ and $\beta_\tau(m)$. Second, the algorithm can be made applicable to all finite-state channels by expanding the state-space to be the cross-product of the encoder states and the channel states. Finally, the extension to nonbinary codes is quite obvious.

Acknowledgment

We are very grateful to Dr. G. D. Forney, Jr., for many helpful comments and suggestions. Most of the notation used in this correspondence is taken from a paper of his [7]. Also, he pointed out the isomorphism between this algorithm and the method of Chang and Hancock for intersymbol interference. We are also indebted to the other reviewer of this paper for his useful suggestions.

References

[1] A. J. Viterbi, "Error bounds for convolutional codes and an asymtotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, Apr. 1967.
[2] G. D. Forney, Jr., "Review of random tree codes," NASA Ames Research Center, Moffett Field, Calif., Appendix A of Final Report on Contract NAS2-3637, NASA CR73176, Dec. 1967.
[3] R. W. Chang and J. C. Hancock, "On receiver structures for channels having memory," *IEEE Trans. Inform. Theory.*, vol. IT-12, pp. 463–468, Oct. 1966.
[4] L. E. Baum and T. Petrie, "Statistical interference for probabilistic functions of finite state markov chains," *Ann. Math. Statist.*, vol. 37, pp. 1559–1563, 1966.
[5] P. L. McAdam, L. R. Welch, and C. L. Weber, "M.A.P. bit decoding of convolutional codes," presented at the 1972 Int. Symp. Information Theory, Asilomar, Calif.
[6] F. Jelinek and J. Cocke, "Bootstrap hybrid decoding for symmetrical binary input channels," *Inform. Contr.*, vol. 18, pp. 261–298, Apr. 1971.
[7] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.