Two Trellis Coding Schemes for Large Free Distances

Mao-Chao Lin, *Member, IEEE*, Yeong-Luh Ueng, and Jia-Yin Wang

***Abstract*—A trellis code encoded by using the encoder of a convolutional code C with a short constraint length followed by an additional processing unit is equivalent to a trellis code with a large constraint length. In 1993, Hellstern proposed a trellis coding scheme for which the processing unit consists of a delay processor and a signal mapper. With Hellstern's scheme, trellis codes with large free distances can be constructed. In this paper, we propose two trellis coding schemes. For the first scheme, the processing unit is composed of multiple pairs of delay processors and signal mappers. For the second scheme, the processing unit is composed of a convolutional processor and a signal mapper, where a convolutional processor is a rate 1 convolutional code. The trellis code constructed from each of the proposed schemes can be suboptimally decoded by using the trellis of the convolutional code C with some feedback information. Either of the proposed schemes can produce a trellis code that has a larger bound on free distance and better error performance as compared to the trellis code constructed from Hellstern's scheme based on the same convolutional code C.**

***Index Terms*—Convolutional codes, trellis-coded modulation, trellis codes.**

I. Introduction

In 1993, Hellstern [1] proposed a coding scheme to construct trellis-coded modulation (TCM) with large free distances. The encoding of Hellstern's scheme is implemented by inserting a multilevel delay processor between the convolutional encoder C and the signal mapper required by the encoding for Ungerboeck's TCM [2]. Hellstern's scheme can also be used to construct binary convolutional codes with large free distances. In this paper, we classify both binary convolutional codes and TCM as trellis codes. Suppose that $v_1(t)$, $v_2(t)$, ..., $v_m(t)$ form the m-bit output of the convolutional encoder at the tth time unit. Then, the bit $v_j(t)$ (the code bit for the jth coding level), $1 \le j \le m$, is delayed by $\tau_j = (m-j)\lambda$ time units before it is fed into the signal mapper, where λ is a delay constant. If for each time unit the output of the signal mapper is a binary m-tuple, then the resultant trellis code is a binary convolutional code. If for each time unit the output of the signal mapper is a signal point of a signal constellation (such as $MPSK$, $M = 2^m$), then the resultant trellis code is a TCM. The introduction of the multilevel delay processor increases the constraint length of the trellis code. By properly designing the delay processor and the signal mapper, a large free distance for the trellis code can be achieved. The trellis code can be suboptimally decoded by using the trellis of C and some previously recovered information. In this way, good error performance can be achieved with moderate decoding complexity.

Hellstern's scheme was generalized in [3] by using a more general delay processor. With the generalization in [3], we have more flexibility of controlling the decoding delay and sometimes have better error performance for low signal-to-noise ratio conditions.

In this paper, we propose two trellis coding schemes. Using the proposed schemes, codes with very large free distances can be constructed. In Section II, we briefly review the trellis coding scheme proposed in [1] and [3]. In Section III, we propose the first coding scheme, for which the encoder of a convolutional code C is followed by multiple pairs of delay processors and signal mappers. In Section IV, we propose the second trellis coding scheme, for which the encoder of a convolutional code C is followed by a convolutional processor and a signal mapper, where a convolutional processor is a rate 1 convolutional code. Both the proposed schemes can be suboptimally decoded by using the trellis of C. Simulation for various trellis codes has been implemented without using interleaving and iterative decoding. The superiority of the proposed schemes over Hellstern's scheme can be observed from the calculated lower bounds on free distance for examples provided in Sections II–IV and simulation results presented in Section V. Comparison of the proposed trellis codes with the conventional binary convolutional code, Ungerboeck's TCM, and turbo code will be given in Section V. Finally, concluding remarks are given in Section VI.

II. Trellis Coding Using a Delay Processor and a Signal Mapper

In this section, we briefly review the design of trellis codes given in [1] and [3]. The encoding is given in Fig. 1. The encoder for a rate r/m convolutional code C converts an input message sequence $\overline{u} = \{\ldots, \hat{u}(0), \hat{u}(1), \ldots\}$ to a sequence $\overline{v} = \{\ldots, \hat{v}(0), \hat{v}(1), \ldots\}$, where $\hat{u}(t) = (u_1(t), \ldots, u_r(t))$ is an r-bit symbol and $\hat{v}(t) = (v_1(t), \ldots, v_m(t))$ is an m-bit symbol. The sequence $\overline{v}$ is then fed into the delay processor which produces a sequence $\overline{s} = \{\ldots, \hat{s}(0), \hat{s}(1), \ldots\}$, where $\hat{s}(t) = (s_1(t), \ldots, s_m(t))$ is an m-bit symbol. The relation between $v_j(t)$ and $s_j(t)$ can be described by

$$s_j(t) = v_j\left(t - \sum_{i=j}^{m} \lambda_i\right) = v_j(t - \tau_j), \qquad \text{for } 1 \le j \le m \tag{1}$$

Paper approved by T. Aulin, the Editor or Coding and Communication Theory of the IEEE Communications Society. Manuscript received February 9, 1999; revised June 21, 1999, January 14, 2000, and January 20, 2000. This work was supported in part by the National Science Council of the R.O.C. under Grant NSC 87-2219-E-002-003. This paper was presented in part at the 1997 IEEE International Symposium on Information Therory, Ulm, Germany, June 29–July 4, 1997, and in part at the 10th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'99), Osaka Japan, September 12–15, 1999.

M.-C. Lin is with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: mclin@cc.ee.ntu.edu.tw).

Y.-L. Ueng is with the Graduate Institute of Communication Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: uyl@eagle.ee.ntu.edu.tw).

J.-Y. Wang is with the Broadband Network Systems Department, Computer and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan 310, R.O.C. (e-mail: jyw@panda.ccl.itri.org.tw).

Publisher Item Identifier S 0090-6778(00)07108-7.

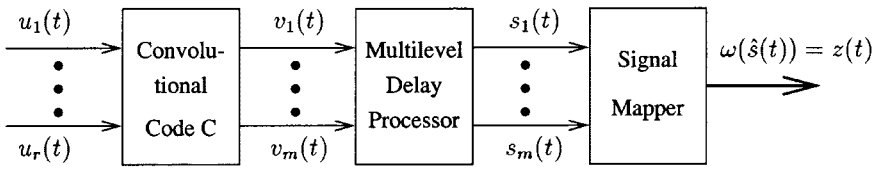


Fig. 1. Encoding structure of trellis codes in [1] and [3].

where $\tau_j = \sum_{i=j}^{m} \lambda_i$, $\lambda_m = 0$, and $\lambda_i \geq 0$ for $1 \leq i \leq m-1$. The sequence $\overline{s}$ is then fed into the signal mapper to yield an output symbol sequence $\overline{z} = \{\ldots, z(0), z(1), \ldots\}$, where $z(t) = \omega(\hat{s}(t)) \in \Omega$ and Ω is a signal space consisting of 2^m signal points. The signal space Ω can be a signal constellation, such as MPSK ($M = 2^m$), or a collection of binary m-tuples, such as $\{0, 1\}^m$.

In [1], constant delays are used in the delay processor and in [3] delays of various values are used. For simplification of presentation, we only consider constant delays here. Hence, we have $\lambda_j = \lambda$ for $1 \leq j \leq m-1$ and $\tau_j = (m-j)\lambda$ for $1 \leq j \leq m$. We can construct an m-level partition chain $\Omega_0/\Omega_1/\Omega_2/\cdots/\Omega_m$ such that every signal point z in $\Omega = \Omega_0$ corresponds to a unique binary m-tuple $\hat{s} = (s_1, s_2, \ldots, s_m)$, i.e., $z = \omega(\hat{s})$, where $z \in \Omega$ and s_i, $1 \leq i \leq m$, is the coset label of Ω_{i-1}/Ω_i [1], [4]. Let $\Delta(z, z')$ denote a distance measure between signal points $z, z' \in \Omega$. If Ω is a signal constellation, then $\Delta(z, z')$ is the squared Euclidean distance between z and z'. If $\Omega = \{0, 1\}^m$, then $\Delta(z, z')$ is the Hamming distance between the binary representations of z and z'. We define Δ_j to be the least one of all the possible $\Delta(\omega(\hat{s}), \omega(\hat{s}'))$, where $\omega(\hat{s})$ and $\omega(\hat{s}')$ are in an arbitrary coset of Ω_{j-1}, $\omega(\hat{s})$ is in a coset of Ω_j labeled by $s_j = 1$ and $\omega(\hat{s}')$ is in a coset of Ω_j labeled by $s'_j = 0$. If Ω is the 8-PSK signal constellation as given in [2], then $\{\Delta_1, \Delta_2, \Delta_3\} = \{0.586, 2, 4\}$. If $\Omega = \{0, 1\}^m$ and the mapping ω: $\Omega \to \Omega$ is linear, then ω can be represented by an $m \times m$ matrix K. That means $z = \omega(\hat{s}) = \hat{s}K$. It can be checked that $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \{1, 2, 2, 4\}$, if $\Omega = \{0, 1\}^4$ and

$$K = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{2}$$

Let $\overline{z}$ and $\overline{z}'$ be the output symbol sequences associated with $\overline{v} = \{\ldots, \hat{v}(0), \hat{v}(1), \ldots\}$ and $\overline{v}' = \{\ldots, \hat{v}'(0), \hat{v}'(1), \ldots\}$, respectively. Assume $\hat{v}(t) = \hat{v}'(t)$ for $t < 0$ and $\hat{v}(0) \neq \hat{v}'(0)$. The pairwise distance measure between sequences $\overline{z}$ and $\overline{z}'$ is lower bounded [3] by

$$\Delta_{LB}((\overline{v}, \overline{v}'), \lambda, \Delta_j) = \sum_{j=1}^{m} \sum_{t=0}^{\lambda-1} (v_j(t) \oplus v'_j(t))\Delta_j. \tag{3}$$

Define Δ_{free} as the free distance of the trellis code (binary convolutional code) if $\Omega = \{1, 0\}^m$, and as the squared free distance of the trellis code (TCM) if Ω is a signal constellation. Then, we have the following theorem [3].

Theorem 1: For the trellis code shown in Fig. 1 with $\lambda_j = \lambda$, $1 \leq j \leq m-1$, its Δ_{free} is lower bounded by

$$\Delta_{LB}(C, \lambda, \Delta_j) = \min_{\overline{v} \in C, \hat{v}(0) \neq \hat{0}} \sum_{j=1}^{m} \sum_{t=0}^{\lambda-1} v_j(t)\Delta_j \tag{4}$$

where $\hat{0}$ is the zero m-tuple. □

Example 1: Let $r = 2$, $m = 3$ and Ω be the 8-PSK signal set [2] with $\{\Delta_1, \Delta_2, \Delta_3\} = \{0.586, 2, 4\}$. The resultant code is a TCM for which its coding rate is 2 bits per 8-PSK signal point. Let ν be the number of memory bits in the encoder of C. Consider the following two cases: a) $\nu = 2$ and b) $\nu = 4$. The generator matrices are, respectively, given by

$$G_E = \begin{pmatrix} 1 & 1 & 1 \\ 5 & 7 & 0 \end{pmatrix} \quad \text{and} \quad G_E = \begin{pmatrix} 4 & 7 & 3 \\ 5 & 3 & 4 \end{pmatrix}.$$

From (4), we can calculate the squared free distance D^2_{free} of this TCM. D^2_{free} is at least 6.34 if $\lambda \geq 11$ for case a) and at least 8.93 if $\lambda \geq 23$ for case b). □

Example 2: Let $r = 2$, $m = 4$, and $\Omega = \{1, 0\}^4$ with the mapping described in (2) and $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \{1, 2, 2, 4\}$. The resultant code is a rate 1/2 convolutional code. Consider the following two cases: a) $\nu = 2$ and b) $\nu = 4$. The generator matrices are, respectively, given by

$$G_E = \begin{pmatrix} 0 & 3 & 3 & 2 \\ 3 & 3 & 2 & 1 \end{pmatrix} \quad \text{and} \quad G_E = \begin{pmatrix} 0 & 2 & 7 & 7 \\ 7 & 5 & 7 & 2 \end{pmatrix}.$$

The free distance d_{free} is at least 12 if $\lambda \geq 5$ for case a) and at least 16 if $\lambda \geq 9$ for case b). □

A suboptimum decoding which only needs the trellis of C can be designed [1], [3]. Let $y(t)$ be the received symbol which is the possibly error-corrupted form of $z(t)$. For the tth time unit, we calculate the bit metric for $v_j(t)$ based on the received $y(t+\tau_j)$ and the previously recovered code bit, $v_i(t-(j-i)\lambda)$, $1 \leq i < j$. Then, the bit metrics for $v_1(t), \ldots, v_m(t)$ are summed up to form the branch metric for $\hat{v}(t)$. With the branch metrics for all the possible $\hat{v}(t-i)$, $i \geq 0$, the Viterbi decoder of C can recover $\hat{u}(t-\lambda+1)$ and $\hat{v}(t-\lambda+1)$, where λ is also used as the truncation length of decoding. The decoding delay is $m\lambda-1$ time units.

III. Trellis Coding Using Multiple Pairs of Delay Processors and Signal Mappers

As indicated in [1] and [3], we observe that a delay processor and a signal mapper following the encoder of a convolutional code C can result in a convolutional code C' of a large free distance. It is natural to consider once again applying a delay processor and a signal mapper to the output of the convolutional code C' to achieve a possibly larger free distance.

Consider a trellis code with its encoder given in Fig. 2. The input message sequence $\overline{u}$ is first converted to a sequence $\overline{v}^{(1)}$ through an encoder of a convolutional code C. The sequence $\overline{v}^{(1)}$ is then repeatedly processed by L pairs of delay processors (denoted as $Q^{(1)}, \ldots, Q^{(L)}$, respectively, in Fig. 2) and signal mappers with mapping functions of $\omega^{(1)}, \ldots, \omega^{(L)}$, respectively, to produce the output symbol sequence $\overline{z}$. Let $\overline{v}^{(\ell)} = (\ldots, \hat{v}^{(\ell)}(0), \hat{v}^{(\ell)}(1), \ldots)$ be the input sequence for $Q^{(\ell)}$ and $\overline{s}^{(\ell)} = (\ldots, \hat{s}^{(\ell)}(0), \hat{s}^{(\ell)}(1), \ldots)$ be the output sequence for

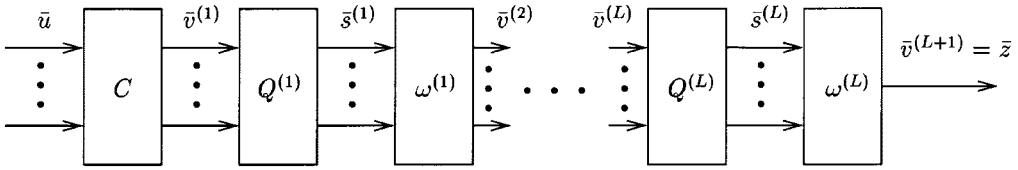


Fig. 2. Encoding of a trellis code using multiple pairs of delay processors and signal mappers.

$Q^{(\ell)}$, where $1 \le \ell \le L$. The ℓth signal mapper maps the binary m-tuple $\hat{s}^{(\ell)}(t) = (s_1^{(\ell)}(t), \ldots, s_m^{(\ell)}(t))$ to

$$\hat{v}^{(\ell+1)}(t) = \omega^{(\ell)}(\hat{s}^{(\ell)}(t)), \qquad \text{for } 1 \le \ell \le L. \tag{5}$$

For $1 \le \ell \le L$, $\hat{v}^{(\ell)}(t) = (v_1^{(\ell)}(t), \ldots, v_m^{(\ell)}(t))$ is a binary m-tuple, while $\hat{v}^{(L+1)}(t) = z(t) = \omega^{(L)}(\hat{s}^{(L)}(t)) \in \Omega$ is the output symbol which may be a binary m-tuple or a signal point of a signal constellation. The relation between $s_j^{(\ell)}(t)$ and $v_j^{(\ell)}(t)$ is given by

$$\begin{aligned} s_j^{(\ell)}(t) &= v_j^{(\ell)}\left(t - (m-j)\lambda^{(\ell)}\right) \\ &= v_j^{(\ell)}\left(t - \tau_j^{(\ell)}\right), \qquad \text{for } 1 \le j \le m;\ 1 \le \ell \le L \end{aligned} \tag{6}$$

where $\tau_j^{(\ell)} = (m-j)\lambda^{(\ell)}$. Suppose that the mapping function $\omega^{(\ell)}$ is linear and invertible for $1 \le \ell \le L-1$. Then $\omega^{(\ell)}$ can be represented by an $m \times m$ nonsingular matrix $K^{(\ell)}$. That is

$$\hat{v}^{(\ell+1)}(t) = \hat{s}^{(\ell)}(t)K^{(\ell)}, \qquad \text{for } 1 \le \ell \le L-1. \tag{7}$$

The mapping function $\omega^{(L)}$ can also be represented by a nonsingular matrix $K^{(L)}$ if $\Omega = \{0, 1\}^m$ and $\omega^{(L)}$ is linear and invertible.

Let $C^{(L)}$ be the collection of $\overline{v}^{(L)}$ resultant from all the possible $\overline{u}$. By Theorem 1, Δ_{free} of the trellis code shown in Fig. 2 is lower bounded by

$$\begin{aligned} &\Delta_{LB}(C^{(L)}, \lambda^{(L)}, \Delta_j) \\ &\quad = \min_{\overline{v}^{(L)} \in C^{(L)}, \hat{v}^{(L)}(0) \ne \hat{0}} \sum_{j=1}^{m} \sum_{t=0}^{\lambda^{(L)}-1} v_j^{(L)}(t)\Delta_j. \end{aligned} \tag{8}$$

It is difficult to calculate the lower bound of the Δ_{free} based on (8). Hence, we will resort to another approach to calculate the lower bound. In the following, we will assume $\lambda^{(\ell)} \ge m\lambda^{(\ell-1)}$ for $2 \le \ell \le L$. Let $\delta_j^{(L)} = \Delta_j$, $1 \le j \le m$. Define

$$d_w^{(\ell)}\left(\hat{v}^{(\ell)}\right) = \sum_{j=1}^{m} v_j^{(\ell)}\delta_j^{(\ell)}, \qquad \text{for } 1 \le \ell \le L \tag{9}$$

and

$$\delta_j^{(\ell)} = \min d_w^{(\ell+1)}\left(\omega^{(\ell)}\left(\hat{s}^{(\ell)}\right)\right), \qquad \text{for } 1 \le \ell \le L-1 \tag{10}$$

where the minimization is over all the possible $\hat{s}^{(\ell)}$ with $\hat{s}_j^{(\ell)} \ne \hat{0}$ and $s_i^{(\ell)} = 0$ for $i < j$. With the initially given $\{\delta_1^{(L)}, \ldots, \delta_m^{(L)}\}$, we can recursively calculate $\{\delta_1^{(\ell)}, \ldots, \delta_m^{(\ell)}\}$ for $\ell = L-1, \ldots, 1$. From Appendix A, we have the following theorem.

Theorem 2: If $\lambda^{(\ell+1)} \ge m\lambda^{(\ell)}$ for $1 \le \ell < L$, then Δ_{free} of the trellis code shown in Fig. 2 is lower bounded by

$$\begin{aligned} \Delta_{LB}\left(C, \lambda^{(1)}, \delta_j^{(1)}\right) &= \min_{\overline{v}^{(1)} \in C,\, \hat{v}^{(1)}(0) \ne \hat{0}} \sum_{t=0}^{\lambda^{(1)}-1} d_w^{(1)}\left(\hat{v}^{(1)}(t)\right) \\ &= \min_{\overline{v}^{(1)} \in C,\, \hat{v}^{(1)}(0) \ne \hat{0}} \sum_{t=0}^{\lambda^{(1)}-1} \sum_{j=1}^{m} v_j^{(1)}(t)\delta_j^{(1)}. \end{aligned} \tag{11}$$

□

For arbitrarily chosen nonsingular matrices $K^{(\ell)}$, $1 \le \ell \le L-1$, it is not necessarily true that $\delta_j^{(1)} \ge \Delta_j$ for all j. In this paper, we use the following design procedure to choose $K^{(\ell)}$.

Step 1) Let $I = \{1, 2, \ldots, m\}$ be the index set. Suppose that $\delta_{i_1}^{(L)} = \delta_{i_1+1}^{(L)} = \cdots = \delta_{i_2-1}^{(L)} < \delta_{i_2}^{(L)} = \delta_{i_2+1}^{(L)} = \cdots = \delta_{i_3-1}^{(L)} < \delta_{i_3}^{(L)} = \delta_{i_3+1}^{(L)} = \cdots = \delta_{i_p-1}^{(L)} < \delta_{i_p}^{(L)} = \delta_{i_p+1}^{(L)} = \cdots = \delta_{i_q-1}^{(L)} < \delta_{i_q}^{(L)} = \delta_{i_q+1}^{(L)} = \cdots = \delta_m^{(L)}$, where $J \equiv \{i_1, \ldots, i_q\} \subseteq I$ and $1 = i_1 < i_2 < \cdots < i_q \le m$. For a nonzero m-tuple, $\hat{v} = (v_1, \ldots, v_m)$, define $R(\hat{v})$ to be the largest index i such that $v_i \ne 0$. Let $C_{j,m}^{(L-1)}$ be the code generated by the last $(m-j+1)$ rows of a nonsingular matrix $K^{(L-1)}$. We choose $K^{(L-1)}$ such that a) for $i_p \le j < i_{p+1}$, $p = 1, 2, \ldots, q$

$$\min_{\hat{v} \in C_{j,m}^{(L-1)}, \hat{v} \ne \hat{0}} R(\hat{v}) \ge i_p,$$

where $i_p \in J$ and $i_{q+1} = m+1$; b) there is at least one j such that the minimum Hamming distance of $C_{j,m}^{(L-1)}$ is at least 2. Note that (9) and (10) imply

$$\delta_j^{(L-1)} = \min_{\hat{v} \in C_{j,m}^{(L-1)}, \hat{v} \ne \hat{0}} \sum_{i=1}^{m} v_i\delta_i^{(L)}. \tag{12}$$

From condition 1), we have $\delta_j^{(L-1)} \ge \delta_j^{(L)}$ for $1 \le j \le m$ and $\delta_1^{(L-1)} \le \delta_2^{(L-1)} \cdots \le \delta_m^{(L-1)}$. With the additional condition 2), we further have $\delta_j^{(L-1)} > \delta_j^{(L)}$ for at least one j.

Step 2) Choose $K^{(\ell)}$, $\ell = L-2, \ldots, 1$, in a way similar to Step 1). Then we have $\delta_j^{(\ell)} \ge \delta_j^{(\ell+1)}$ for $\ell = L-2, \ldots, 1$, and $1 \le j \le m$. Hence $\delta_j^{(1)} \ge \Delta_j$ for $1 \le j \le m$ and $\delta_j^{(1)} > \Delta_j$ for at least one j.

Step 3) For a given C, we compute the lower bound of Δ_{free} according to (11). This can be done in a way similar to the computation of the free distance of C except that the weighting factors $\delta_1^{(1)}, \ldots, \delta_{m-1}^{(1)}$ and $\delta_m^{(1)}$ must be considered.

Using the above procedure, we can construct trellis codes with large free distances.

Example 3: Let C be a rate 2/3 binary convolutional code, $L = 2$ and Ω be the 8-PSK signal set [2] with $\{\Delta_1, \Delta_2, \Delta_3\} = \{0.586, 2, 4\}$. Choose

$$K^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

The resultant code is a TCM for which its coding rate is 2 bits per 8-PSK signal point. We have $\{\delta_1^{(2)}, \delta_2^{(2)}, \delta_3^{(2)}\} = \{0.586, 2, 4\}$. From (12), we have $\{\delta_1^{(1)}, \delta_2^{(1)}, \delta_3^{(1)}\} = \{0.586, 2.586, 6.586\}$. We consider the following two cases: a) $\nu = 2$ and b) $\nu = 4$. The associated generator matrices are the same as those used in Example 1. From (11), we can calculate the squared free distance D_{free}^2 of this TCM. For case a), D_{free}^2 is at least 7.52 if $\lambda^{(1)} \geq 12$ and $\lambda^{(2)} \geq 36$. For case b), D_{free}^2 is at least 10.69 if $\lambda^{(1)} \geq 25$ and $\lambda^{(2)} \geq 75$. □

Example 4: Let C be a rate 2/4 binary convolutional code, $L = 2$ and $\Omega = \{0, 1\}^4$. Let $K^{(2)}$ be the same as the matrix K given in (2). Then, $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \{1, 2, 2, 4\}$. Choose

$$K^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

The resultant code is a rate 1/2 binary convolutional code. We have $\{\delta_1^{(2)}, \delta_2^{(2)}, \delta_3^{(2)}, \delta_4^{(2)}\} = \{1, 2, 2, 4\}$. From (12), we have $\{\delta_1^{(1)}, \delta_2^{(1)}, \delta_3^{(1)}, \delta_4^{(1)}\} = \{1, 3, 3, 9\}$. Consider the following two cases: a) $\nu = 2$ and b) $\nu = 4$. The associated generator matrices are the same as those used in Example 2. The free distance d_{free} of the constructed binary trellis code is at least 17 if $\lambda^{(1)} \geq 5$ and $\lambda^{(2)} \geq 20$ for case a) and is at least 24 if $\lambda^{(1)} \geq 10$ and $\lambda^{(2)} \geq 40$ for case b). □

A suboptimum decoding for the trellis code can be implemented as follows. Let $\|y\text{–}z\|^2$ represent the squared Euclidean distance between symbol y and symbol z. If y or z is a binary m-tuple, then a bit "0" must be replaced by "−1" and a bit "1" remains to be "1." Let $y(t) = \hat{v}^{*(L+1)}(t)$ be the received symbol that is the possibly error-corrupted form of $z(t) = \hat{v}^{(L+1)}(t)$. For $\ell = L, \ldots, 1$, $1 \leq j \leq m$, define the log-likelihood-ratio [1] for bit $v_j^{(\ell)}(t)$ by

$$\begin{aligned}\mu_j^{(\ell)}(t) &= \ln \frac{p\left(\hat{v}^{*(\ell+1)}\left(t+\tau_j^{(\ell)}\right) \middle| v_j^{(\ell)}(t) = 1\right)}{p\left(\hat{v}^{*(\ell+1)}\left(t+\tau_j^{(\ell)}\right) \middle| v_j^{(\ell)}(t) = 0\right)} \\ &\approx \min_{x \in A_0} \frac{\left\|\hat{v}^{*(\ell+1)}\left(t+\tau_j^{(\ell)}\right) - \omega^{(\ell)}(x)\right\|^2}{N_0} \\ &\quad - \min_{x \in A_1} \frac{\left\|\hat{v}^{*(\ell+1)}\left(t+\tau_j^{(\ell)}\right) - \omega^{(\ell)}(x)\right\|^2}{N_0} \end{aligned} \tag{13}$$

where $\hat{v}^{*(\ell+1)}(p) = (\mu_1^{(\ell+1)}(p), \ldots, \mu_m^{(\ell+1)}(p))$ for $\ell = L-1, \ldots, 1$ and $A_i = \{\omega^{(\ell)}(s_1^{(\ell)}(t+\tau_j^{(\ell)}), \ldots, s_{j-1}^{(\ell)}(t+\tau_j^{(\ell)}), i, x_{j+1}, \ldots, x_m) \mid x_q \in \{0, 1\}$ for $j < q \leq m\}$ for $i = 0, 1$. Let $\eta^{(1)} = 0$, $\eta^{(\ell)} = \sum_{i=1}^{\ell-1} \tau_1^{(i)}$ and $\lambda^{(\ell)} \geq m\lambda^{(\ell-1)}$ for $\ell \geq 2$. Note that $\lambda^{(2)} \geq \lambda^{(1)} + \eta^{(2)}$. Since $\lambda^{(\ell)} \geq \lambda^{(1)} + \eta^{(\ell)}$ implies that $\lambda^{(\ell+1)} \geq \lambda^{(\ell)} + \tau_1^{(\ell)} \geq \lambda^{(1)} + \eta^{(\ell)} + \tau_1^{(\ell)} = \lambda^{(1)} + \eta^{(\ell+1)}$, then $\lambda^{(\ell)} \geq \lambda^{(1)} + \eta^{(\ell)}$ for $\ell \geq 2$. Assume that $\hat{v}^{(\ell)}(t-l)$ has been correctly recovered for $l \geq \lambda^{(1)}$ and $1 \leq \ell \leq L$. The suboptimum decoding can be implemented as follows.

Step 1) For $\ell = L, \ldots, 1$, we use (13) to calculate $\mu_j^{(\ell)}(t + \eta^{(\ell)})$ for $v_j^{(\ell)}(t + \eta^{(\ell)})$. Note that for $1 \leq p < j$, $s_p^{(\ell)}(t+\tau_j^{(\ell)}+\eta^{(\ell)}) = v_p^{(\ell)}(t-(j-p)\lambda^{(\ell)}+\eta^{(\ell)})$ has already been recovered since $(j-p)\lambda^{(\ell)} \geq \lambda^{(1)} + \eta^{(\ell)}$.

Step 2) Using $\mu_j^{(1)}(t) = \mu_j^{(1)}(t + \eta^{(1)})$, $j = 1, \ldots, m$, we calculate the branch metrics for all the possible $\hat{v}(t)$, which are fed to the Viterbi decoder of C to recover $\hat{u}(t-\lambda^{(1)}+1)$ and $\hat{v}^{(1)}(t-\lambda^{(1)}+1)$, where $\lambda^{(1)}$ is used as the truncation length of the Viterbi decoder of C.

Step 3) The recovered $\hat{v}^{(1)}(t-\lambda^{(1)}+1)$ is used to recover $\hat{s}^{(1)}(t-\lambda^{(1)}+1)$ which is then used to recover $\hat{v}^{(\ell)}(t-\lambda^{(1)}+1)$ and $\hat{s}^{(\ell)}(t-\lambda^{(1)}+1)$ for $\ell = 2, \ldots, L$. Then we increase t by 1 and go to step 1.

The error performance of the suboptimum decoding can be further improved by using SOVA [5], since the assumption of correct recovery of $\hat{v}^{(\ell)}(t-l)$ is not always true. Let the log-likelihood-ratio obtained by SOVA for $s_j^{(\ell)}(t)$ be denoted by $\Lambda(s_j^{(\ell)}(t))$. Then, the parameter $\mu_j^{(\ell)}(t)$ given in (13) is modified to be (14), shown at the bottom of the next page, where $B_i = \{\omega^{(\ell)}(s_1^{(\ell)}(t+\tau_j^{(\ell)}), \ldots, s_{j-1}^{(\ell)}(t+\tau_j^{(\ell)}), i, x_{j+1}, \ldots, x_m) | s_q^{(\ell)}(t+\tau_j^{(\ell)}) \in \{0, 1\}$ for $1 \leq q < j$, $x_q \in \{0, 1\}$ for $j < q \leq m\}$ for $i = 0, 1$. The total decoding delay is $\lambda^{(1)} + \eta^{(L+1)} - 1$ time units. If $\lambda^{(\ell)} = m\lambda^{(\ell-1)}$ for $\ell \geq 2$, then $\lambda^{(1)} + \eta^{(L+1)} - 1 = m^L\lambda^{(1)} - 1$.

IV. Trellis Coding Using a Convolutional Processor and a Signal Mapper

The trellis coding shown in Fig. 2 can be described in a different way. Let D represent the operator of one unit time delay. Define an $m \times m$ diagonal matrix $Q_M^{(\ell)}$ for $1 \leq \ell \leq L$, for which the entry at the ith row and the ith column is $D^{\tau_i^{(\ell)}}$. The function of the delay processor $Q^{(\ell)}$ can be represented by $\hat{s}^{(\ell)}(t) = \hat{v}^{(\ell)}(t)Q_M^{(\ell)}$ for $1 \leq \ell \leq L$. Then, we have

$$\begin{aligned}\hat{s}^{(L)}(t) &= \hat{v}^{(L)}(t)Q_M^{(L)} \\ &= \hat{s}^{(L-1)}(t)K^{(L-1)}Q_M^{(L)} \\ &= \cdots \\ &= \hat{v}^{(1)}(t)Q_M^{(1)}K^{(1)}Q_M^{(2)}\cdots K^{(L-1)}Q_M^{(L)} \\ &= \hat{v}^{(1)}(t)P \end{aligned} \tag{15}$$

where $P = Q_M^{(1)}K^{(1)}Q_M^{(2)}\cdots K^{(L-1)}Q_M^{(L)}$. We may regard P as the transfer function matrix for a rate 1 convolutional code with input sequence $\overline{v}^{(1)}$ and output sequence $\overline{s}^{(L)}$. This suggests a new class of trellis coding with encoding configuration shown in Fig. 3, in which an input message sequence $\overline{u}$ is fed into the encoder of a convolutional code C followed by a convolutional processor and a signal mapper to produce the output symbol sequence $\overline{z}$. The convolutional processor is the encoder of a rate 1 convolutional code with transfer function matrix P.

In the following, we will propose a special design of the convolutional processor P such that the $(j+1)$th level input bit will

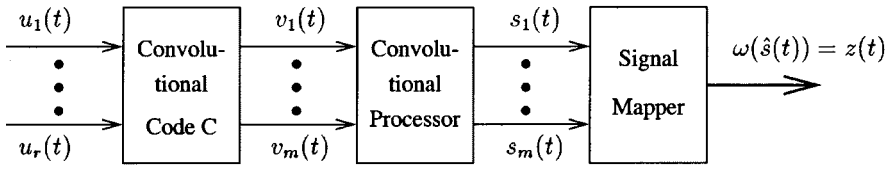


Fig. 3. Encoding structure of a trellis code with a convolutional processor.

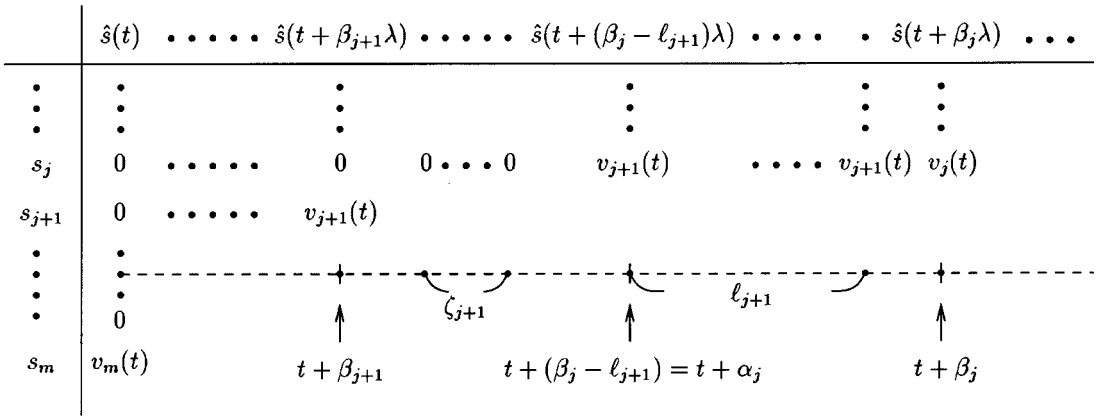


Fig. 4. Relation between sequences $\overline{s}$ and $\overline{v} = (\ldots, \hat{0}, \hat{0}, \hat{v}(t), \hat{0}, \hat{0}, \ldots)$ described by (19), where $\alpha_j = \beta_j - \ell_{j+1}$.

affect both the $(j+1)$th and the jth level output bits. Then $P_{i,j}$, the entry at the ith row and the jth column of the $m \times m$ matrix P, will be zero except for the diagonal elements and $P_{j+1,j}$, where $j = 1, 2, \ldots, m-1$. For $j = 1, 2, \ldots, m$, set

$$P_{j,j} = D^{\beta_j \lambda} \tag{16}$$

and for $j = 1, 2, \ldots, m-1$, set

$$P_{j+1,j} = \begin{cases} 0, & \text{if } \ell_{j+1} = 0 \\ \left(D^{(\beta_j - 1)\lambda} + \cdots + D^{(\beta_j - \ell_{j+1})\lambda}\right), & \text{if } \ell_{j+1} > 0. \end{cases} \tag{17}$$

Then, we have

$$s_m(t) = v_m(t - \beta_m \lambda) \tag{18}$$

and for $j = 1, 2, \ldots, m-1$

$$s_j(t) = \begin{cases} v_j(t-\beta_j\lambda), & \text{if } \ell_{j+1} = 0 \\ v_j(t-\beta_j\lambda) \oplus v_{j+1}(t-(\beta_j-1)\lambda) \\ \oplus \cdots \oplus v_{j+1}(t-(\beta_j-\ell_{j+1})\lambda), & \text{if } \ell_{j+1} > 0. \end{cases} \tag{19}$$

To insure that $v_{j+1}(t)$, $j < m$, can affect $\ell_{j+1} + 1$ different output symbols, we must have $\beta_j - \beta_{j+1} \geq \ell_{j+1} + 1$. If it is desired that the pairwise distance measure between sequences $\overline{z}$ and $\overline{z}'$ is lower bounded by

$$\Delta_{LB}((\overline{v}, \overline{v}'), \lambda, (\Delta_j + \ell_j \Delta_{j-1})) = \sum_{j=1}^{m} \sum_{t=0}^{\lambda-1} (v_j(t) \oplus v_j'(t))(\Delta_j + \ell_j \Delta_{j-1}) \tag{20}$$

we require that $\beta_m = 0$ and for $j = m-1, \ldots, 1$

$$\beta_j - \beta_{j+1} = \ell_{j+1} + \zeta_{j+1} + 1 \tag{21}$$

where $\zeta_{j+1} \geq 0$ is an integer for which the constraint will be given in Theorem 3. The relation between $\overline{s}$ and $\overline{v} = (\hat{0}, \ldots, \hat{0}, \hat{v}(t), \hat{0}, \ldots, \hat{0})$ is illustrated in Fig. 4.

Example 5: Let $\ell_1 = 0$, $\ell_2 = 1$, $\ell_3 = 0$, $\zeta_3 = 0$, $\zeta_2 = 1$, and $\zeta_1 = 0$. It follows from (21) that $\beta_3 = 0$, $\beta_2 = 1$, $\beta_1 = 4$. Hence, by (16) and (17), we have

$$P = \begin{pmatrix} D^{4\lambda} & 0 & 0 \\ D^{3\lambda} & D^{\lambda} & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

We have $s_1(t) = v_1(t - 4\lambda) \oplus v_2(t - 3\lambda)$, $s_2(t) = v_2(t - \lambda)$, and $s_3(t) = v_3(t)$ by (18) and (19). The relation between $\overline{s}$ and $\overline{v}$ is shown in Fig. 5. Suppose that $\hat{v}(t) = \hat{v}'(t)$ for $t < 0$ and $\hat{v}(0) \neq \hat{v}'(0)$. We see that $\Delta(z(0), z'(0)) \geq (v_3(0) \oplus v_3'(0))\Delta_3$, $\Delta(z(\lambda), z'(\lambda)) \geq (v_2(0) \oplus$

$$\mu_j^{(\ell)}(t) \approx \min_{x \in B_0} \left\{ \frac{\left\| \hat{v}^{*(\ell+1)}\left(t + \tau_j^{(\ell)}\right) - \omega^{(\ell)}(x) \right\|^2}{N_0} - \frac{1}{2} \sum_{i=1}^{j-1} \left[2 s_i^{(\ell)}\left(t + \tau_j^{(\ell)}\right) - 1 \right] \Lambda\left(s_i^{(\ell)}\left(t + \tau_j^{(\ell)}\right) \right) \right\}$$
$$- \min_{x \in B_1} \left\{ \frac{\left\| \hat{v}^{*(\ell+1)}\left(t + \tau_j^{(\ell)}\right) - \omega^{(\ell)}(x) \right\|^2}{N_0} - \frac{1}{2} \sum_{i=1}^{j-1} \left[2 s_i^{(\ell)}\left(t + \tau_j^{(\ell)}\right) - 1 \right] \Lambda\left(s_i^{(\ell)}\left(t + \tau_j^{(\ell)}\right) \right) \right\} \tag{14}$$

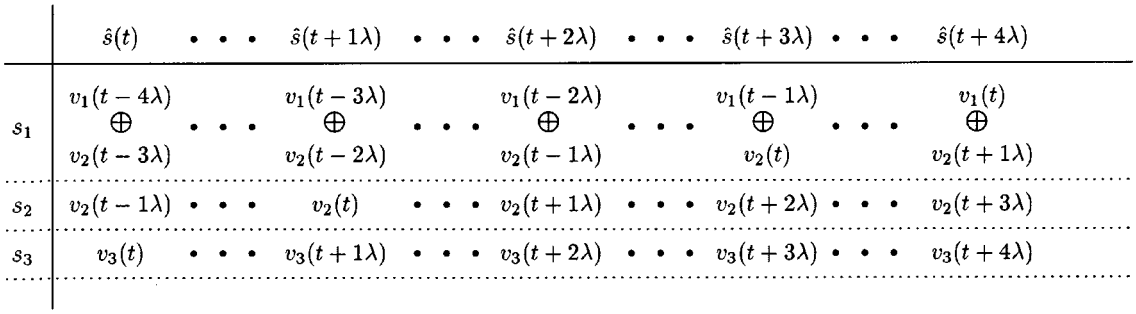


Fig. 5. Relation between sequences $\bar{s}$ and $\bar{v}$ for Example 5.

$v_2'(0))\Delta_2$, $\Delta(z(3\lambda), z'(3\lambda)) \geq (v_2(0) \oplus v_2'(0))\Delta_1$, and $\Delta(z(2\lambda), z'(2\lambda)) + \Delta(z(4\lambda), z'(4\lambda)) \geq (v_1(0) \oplus v_1'(0))\Delta_1$ if $\Delta_2 \geq \Delta_1$. Hence (20) is satisfied. □

From Appendix B, we have the following theorem.

Theorem 3: Let $\ell_j \leq \lfloor \Delta_j/\Delta_{j-1} \rfloor$. If $\zeta_p \geq \zeta_{p-1} + \ell_p$ for $\ell_p > 0$ (and $\zeta_p = 0$ for $\ell_p = 0$), then Δ_{free} of the proposed trellis code is lower bounded by

$$\Delta_{LB}(C, \lambda, (\Delta_j + \ell_j\Delta_{j-1})) = \min_{\bar{v}\in C,\, \hat{v}(0)\neq\hat{0}} \sum_{j=1}^{m} \sum_{t=0}^{\lambda-1} v_j(t)(\Delta_j + \ell_j\Delta_{j-1}) \quad (22)$$

where $\Delta_0 = 0$ and $\ell_1 = 0$. □

For this scheme, we use the following design procedure to construct trellis codes.

Step 1) Select $\ell_1, \ldots, \ell_m$ and $\zeta_1, \ldots, \zeta_m$ which are subject to the constraint given in Theorem 3. Calculate $\beta_1, \ldots, \beta_m$ by (21). Then we have P with entries described by (16) and (17).

Step 2) For a given C, we compute the lower bound of Δ_{free} according to (22). This can be done in a way similar to the computation of free distance of C except that the weighting factors $(\Delta_1 + \ell_1\Delta_0) = \Delta_1, \ldots, (\Delta_m + \ell_m\Delta_{m-1})$ must be considered.

For this scheme, increasing ℓ_j may result in increased Δ_{free} as indicated in (22). However, increasing ℓ_j may not necessarily yield improved error performance. This may result from the increased error coefficient. Moreover, decoding delay is also increased. Hence, a large ℓ_j is not necessarily desired.

Example 6: Let $r = 2, m = 3$, and Ω be the 8-PSK signal set [2] with $\{\Delta_1, \Delta_2, \Delta_3\} = \{0.586, 2, 4\}$. Let P be the matrix used in Example 5. Let d_i denote $\sum_{t=0}^{\lambda-1} v_i(t)$ for $i = 1, 2, 3$. The squared free distance of the constructed TCM is $D^2_{\text{free}} \geq \min_{\bar{v}\in C,\, \hat{v}(0)\neq\hat{0}} \{d_1\cdot\Delta_1 + d_2\cdot(\Delta_2 + \Delta_1) + d_3\cdot\Delta_3\}$. Consider the following two cases: a) $\nu = 2$ and b) $\nu = 4$. The associated generator matrices are the same as those used in Example 1. We have $D^2_{\text{free}} \geq 7.17$ if $\lambda \geq 11$ for case a) and $D^2_{\text{free}} \geq 9.52$ if $\lambda \geq 23$ for case b). □

We can modify Example 6 by using $\ell_3 = 1$ instead of $\ell_3 = 0$. In this way, a larger lower bound on free distance can be achieved. However, the error performance remains similar.

Example 7: Let $r = 2$, $m = 4$, and $\Omega = \{1, 0\}^4$ with the mapping described in (2) and $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \{1, 2, 2, 4\}$. Let $\ell_4 = 0$, $\ell_3 = 1$, $\ell_2 = 1$, $\ell_1 = 0$, $\zeta_4 = 0$, $\zeta_3 = 2$, $\zeta_2 = 1$, $\zeta_1 = 0$. We have

$$P = \begin{pmatrix} D^{8\lambda} & 0 & 0 & 0 \\ D^{7\lambda} & D^{5\lambda} & 0 & 0 \\ 0 & D^{4\lambda} & D^{\lambda} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Then, we have a rate 1/2 binary trellis code with free distance $d_{\text{free}} \geq \min_{\bar{v}\in C,\, \hat{v}(0)\neq\hat{0}} \{d_1\cdot\Delta_1 + d_2\cdot(\Delta_2 + \Delta_1) + d_3\cdot(\Delta_3 + \Delta_2) + d_4\cdot\Delta_4\}$. Consider the following two cases a) $\nu = 2$ and b) $\nu = 4$. The associated generator matrices are the same as those used in Example 2. Then, we have $d_{\text{free}} \geq 14$ if $\lambda \geq 5$ for case a) and $d_{\text{free}} \geq 20$ if $\lambda \geq 7$ for case b). □

With the special design of P, the proposed trellis code can be suboptimally decoded by using the trellis of C. As an illustration, we describe the decoding procedure for Example 6 as follows. We assume that $\hat{v}(t-l)$ has been correctly recovered for $l \geq \lambda$.

Step 1) For $j \in \{0, 1\}$, we calculate the bit metric for $v_1(t) = j$. From Fig. 5, we see that $s_1(t + 4\lambda) = v_1(t) \oplus v_2(t + \lambda)$ contains the information of $v_1(t)$. The bit $v_2(t + \lambda)$ is not yet recovered and also appears in $\hat{s}(t + 2\lambda)$, where $s_1(t + 2\lambda) = v_1(t - 2\lambda) \oplus v_2(t - \lambda)$ is already recovered at earlier time units. Hence, $v_1(t)$ can be estimated from the received symbols $y(t + 4\lambda)$ and $y(t + 2\lambda)$. The bit metric for $v_1(t) = j$ is calculated to be

$$\min_{\hat{s}(t+4\lambda),\, \hat{s}(t+2\lambda)} \{||y(t+4\lambda) - \omega(\hat{s}(t+4\lambda))||^2 + ||y(t+2\lambda) - \omega(\hat{s}(t+2\lambda))||^2\}$$

where $\hat{s}(t + 4\lambda) = (j \oplus x_1, x_3, x_4)$, $\hat{s}(t + 2\lambda) = (v_1(t - 2\lambda) \oplus v_2(t - \lambda), x_1, x_2)$, and $x_1 = v_2(t + \lambda) \in \{0, 1\}$, $x_2 = v_3(t + 2\lambda) \in \{0, 1\}$, $x_3 = v_2(t + 3\lambda) \in \{0, 1\}$, $x_4 = v_3(t + 4\lambda) \in \{0, 1\}$ (since $v_2(t+\lambda), v_3(t+2\lambda), v_2(t+3\lambda)$ and $v_3(t+4\lambda)$ are not yet recovered).

Step 2) For $j \in \{0, 1\}$, we calculate the bit metric for $v_2(t) = j$. We see that $s_2(t + \lambda) = v_2(t)$ and $s_1(t+3\lambda) = v_1(t-\lambda) \oplus v_2(t)$ contain the information of $v_2(t)$, where $v_1(t - \lambda)$ is already recovered. Hence, $v_2(t)$ can be estimated by received symbols $y(t+\lambda)$ and $y(t+3\lambda)$. The bit metric for $v_2(t) = j$ is calculated to be

$$\min_{\hat{s}(t+\lambda)} \{||y(t+\lambda) - \omega(\hat{s}(t+\lambda))||^2\} + \min_{\hat{s}(t+3\lambda)} ||y(t+3\lambda) - \omega(\hat{s}(t+3\lambda))||^2\}$$

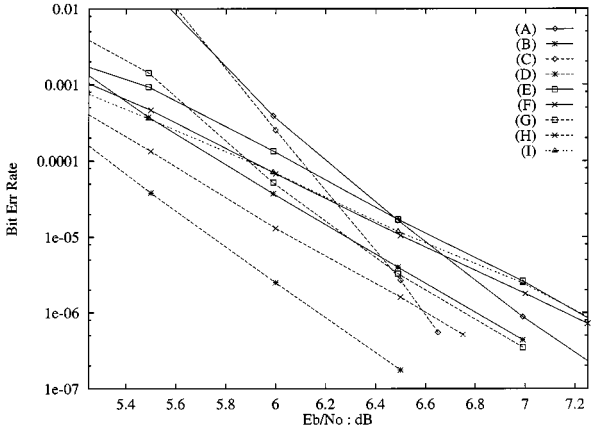


Fig. 6. Simulation results for Examples 1 and 3. (A): 3a, without SOVA, $\nu = 2$, $\lambda^{(1)} = 30$, $\lambda^{(2)} = 90$. (B): 3a, with SOVA, $\nu = 2$, $\lambda^{(1)} = 30$, $\lambda^{(2)} = 90$. (C): 3b, without SOVA, $\nu = 4$, $\lambda^{(1)} = 60$, $\lambda^{(2)} = 180$. (D): 3b, with SOVA, $\nu = 4$, $\lambda^{(1)} = 60$, $\lambda^{(2)} = 180$. (E): 1a, without SOVA, $\nu = 2$, $\lambda = 20$. (F): 1a, with SOVA, $\nu = 2$, $\lambda = 20$. (G): 1b, without SOVA, $\nu = 4$, $\lambda = 40$. (H): 1b, with SOVA, $\nu = 4$, $\lambda = 40$. (I): Ungerboeck's TCM, $\nu = 4$, $\lambda = 24$.

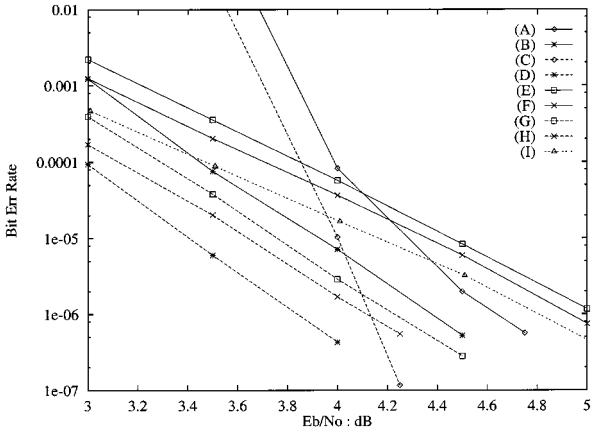


Fig. 7. Simulation results for Examples 2 and 4.(A): 4a, without SOVA, $\nu = 2$, $\lambda^{(1)} = 20$, $\lambda^{(2)} = 80$. (B): 4a, with SOVA, $\nu = 2$, $\lambda^{(1)} = 20$, $\lambda^{(2)} = 80$. (C): 4b, without SOVA, $\nu = 4$, $\lambda^{(1)} = 40$, $\lambda^{(2)} = 160$. (D): 4b, with SOVA, $\nu = 4$, $\lambda^{(1)} = 40$, $\lambda^{(2)} = 160$. (E): 2a, without SOVA, $\nu = 2$, $\lambda = 20$. (F): 2a, with SOVA, $\nu = 2$, $\lambda = 20$. (G): 2b, without SOVA, $\nu = 4$, $\lambda = 40$. (H): 2b, with SOVA, $\nu = 4$, $\lambda = 40$. (I): rate 1/2 conventional convolutional code, $\nu = 6$, $\lambda = 36$.

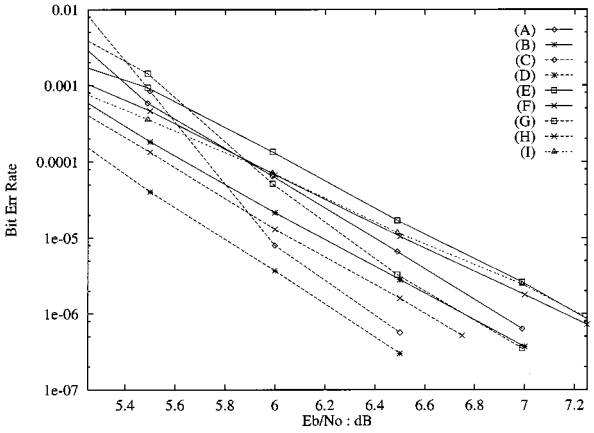


Fig. 8. Simulation results for Examples 1 and 6. (A): 6a, without SOVA, $\nu = 2$, $\lambda = 40$. (B): 6a, with SOVA, $\nu = 2$, $\lambda = 40$. (C): 6b, without SOVA, $\nu = 4$, $\lambda = 80$. (D): 6b, with SOVA, $\nu = 4$, $\lambda = 80$. (E): 1a, without SOVA, $\nu = 2$, $\lambda = 20$. (F): 1a, with SOVA, $\nu = 2$, $\lambda = 20$. (G): 1b, without SOVA, $\nu = 4$, $\lambda = 40$. (H): 1b, with SOVA, $\nu = 4$, $\lambda = 40$. (I): Ungerboeck's TCM, $\nu = 4$, $\lambda = 24$.

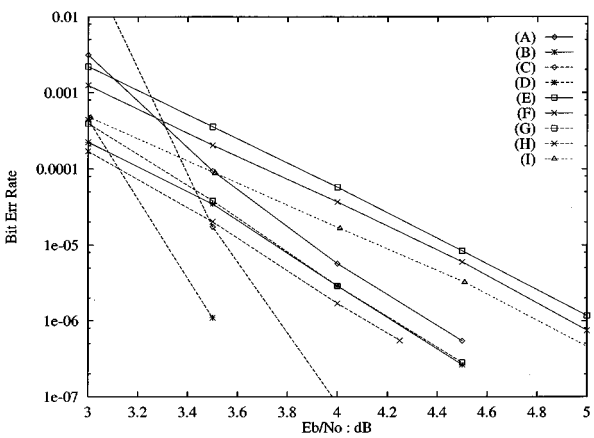


Fig. 9. Simulation results for Examples 2 and 7.(A): 7a, without SOVA, $\nu = 2$, $\lambda = 20$.(B): 7a, with SOVA, $\nu = 2$, $\lambda = 20$.(C): 7b, without SOVA, $\nu = 4$, $\lambda = 40$.(D): 7b, with SOVA, $\nu = 4$, $\lambda = 40$.(E): 2a, without SOVA, $\nu = 2$, $\lambda = 20$.(F): 2a, with SOVA, $\nu = 2$, $\lambda = 20$.(G): 2b, without SOVA, $\nu = 4$, $\lambda = 40$.(H): 2b, with SOVA, $\nu = 4$, $\lambda = 40$.(I): rate 1/2 conventional convolutional code, $\nu = 6$, $\lambda = 36$.

where $\hat{s}(t+\lambda) = (v_1(t-3\lambda) \oplus v_2(t-2\lambda), j, x_1)$, $\hat{s}(t+3\lambda) = (v_1(t-\lambda) \oplus j, x_2, x_3)$, and $x_1 = v_3(t+\lambda) \in \{0,1\}$, $x_2 = v_2(t+2\lambda) \in \{0,1\}$, $x_3 = v_3(t+3\lambda) \in \{0,1\}$.

Step 3) For $j \in \{0,1\}$, the bit metric for $v_3(t) = j$ is calculated to be $\min_{\hat{s}(t)} \{||y(t) - \omega(\hat{s}(t))||^2\}$, where $\hat{s}(t) = (v_1(t-4\lambda) \oplus v_2(t-3\lambda), v_2(t-\lambda), j)$.

Step 4) Calculate the metric for $\hat{v}(t)$ as the sum of bit metrics for $v_1(t)$, $v_2(t)$, and $v_3(t)$.

Step 5) With the branch metric for $\hat{v}(t-i)$, $i \geq 0$, we use the Viterbi decoder with truncation length λ for the convolutional code C to recover $\hat{u}(t-\lambda+1)$ and $\hat{v}(t-\lambda+1)$. Then, we increase t by 1 and proceed to step 1). The decoding delay is $5\lambda - 1$ time units.

In the general case, the decoding is similar to that of Example 6. Suppose that for a nonnegative integer i, $s_p(t+i\lambda)$ contains the information of $v_j(t)$. That means $s_p(t+i\lambda)$ is the sum of $v_j(t)$, $v_{p'}(t+i'\lambda)$ and some other bits. Then, the received symbol $y(t+i\lambda)$ needs to be used in the estimation of $v_j(t)$. If $i' > 0$, and $s_{p''}(t+i''\lambda)$ contains the information of $v_{p'}(t+i'\lambda)$, then the received symbol $y(t+i''\lambda)$ also needs to be used in the estimation of $v_j(t)$. If $s_{p''}(t+i''\lambda)$ is the sum of $v_{p'}(t+i'\lambda)$, $v_{p'''}(t+i'''\lambda)$ and some other bits, where $i''' > 0$, then more received symbols need to be used in the estimation of $v_j(t)$. The bit metrics used in the above mentioned decoding can be modified by introducing log-likelihood ratios obtained by SOVA in a way similar to that described in Section III.

V. Performance Evaluation

Using the proposed coding schemes, we can construct trellis codes with large free distances. However, the suboptimum decoding for each scheme may yield a large error coefficient. Therefore, simulation is needed to verify the error performance. In [1], interleaving is used. However, we do not use interleaving here, since large interleaving size requires large decoding delay. Simulation results over the AWGN channel for Examples 3,

TABLE I
DATA OF THE TRELLIS CODES GIVEN IN EXAMPLES 1–4, 6, AND 7

Example	Ω	$\frac{r}{m}$	ν	Δ_{LB}	(add, comp) (without SOVA)	delay (time units)	λ ($\lambda^{(1)}$)	delay (message bits)	E_b/N_0 for BER=10^{-6}	
									without SOVA(dB)	with SOVA(dB)
1a	8PSK	$\frac{2}{3}$	2	6.34	(32, 20)	$3\lambda - 1$	20	118	7.2	7.15
3a	8PSK	$\frac{2}{3}$	2	7.52	(48, 28)	$9\lambda^{(1)} - 1$	30	538	6.95	6.8
6a	8PSK	$\frac{2}{3}$	2	7.17	(38, 30)	$5\lambda - 1$	40	398	6.9	6.75
1b	8PSK	$\frac{2}{3}$	4	8.93	(80, 56)	$3\lambda - 1$	40	238	6.75	6.6
3b	8PSK	$\frac{2}{3}$	4	10.69	(96, 64)	$9\lambda^{(1)} - 1$	60	1078	6.6	6.2
6b	8PSK	$\frac{2}{3}$	4	9.52	(86, 66)	$5\lambda - 1$	80	798	6.4	6.25
2a	$\{0,1\}^4$	$\frac{2}{4}$	2	12	(64, 34)	$4\lambda - 1$	20	158	5.0	4.9
4a	$\{0,1\}^4$	$\frac{2}{4}$	2	17	(112, 56)	$16\lambda^{(1)} - 1$	20	638	4.65	4.35
7a	$\{0,1\}^4$	$\frac{2}{4}$	2	14	(78, 70)	$9\lambda - 1$	20	358	4.35	4.2
2b	$\{0,1\}^4$	$\frac{2}{4}$	4	16	(112, 70)	$4\lambda - 1$	40	318	4.2	4.1
4b	$\{0,1\}^4$	$\frac{2}{4}$	4	24	(160, 92)	$16\lambda^{(1)} - 1$	40	1278	4.1	3.85
7b	$\{0,1\}^4$	$\frac{2}{4}$	4	20	(126, 106)	$9\lambda - 1$	40	718	3.75	3.5

4, 6, and 7 using the suboptimum decoding (with and without SOVA) are given in Figs. 6–9, respectively. Simulation results for trellis codes constructed from Hellstern's scheme (Examples 1 and 2) and the 64-state conventional binary convolutional code and the 16-state Ungerboeck TCM are also given in these figures. The data related to the simulation for Examples 1–4, 6, and 7 are listed in Table I.

The numbers of additions and comparisons per symbol (add, comp) can be used as one measure of decoding complexity. In Table I, (add, comp) are computed based on the suboptimal decoding using hard feedback, i.e., without SOVA. If SOVA is used, the sum of add and comp will be about twice of that of not using SOVA. From Fig. 8 and Table I, we see that Example 6b has similar complexity and better error performance as compared to Example 1b. In addition to (add, comp), the number of trellis states can serve as another measure of decoding complexity. For a high-speed Viterbi decoder using many parallel processors, the number of trellis states will be a dominant factor of complexity [6]. In this case, either Example 3b or 6b has similar complexity and better error performance as compared to Example 1b. Similar comparison can be made between other examples such as Examples 3a and 1a, Examples 2a and 4a, ..., etc. We can conclude that either of the proposed schemes yields error performance better than Hellstern's scheme based on the same C (or similar decoding complexity). We note that either the four-state Example 3a or four-state Example 6a requires lower E_b/N_o (to achieve $\mathrm{BER} = 10^{-6}$) than that required by the 16-state Ungerboeck's TCM. Moreover, the four-state Example 4a or four-state Example 7a requires lower E_b/N_o (to achieve $\mathrm{BER} = 10^{-6}$) than that required by the 64-state conventional binary convolutional code. Hence, we can conclude that either of the proposed schemes has better error performance and lower decoding complexity as compared to the conventional trellis code. According to Table I, we see that the decoding delay of either of the proposed schemes is longer than that of Hellstern's scheme.

It is interesting to compare the proposed scheme with the turbo code [7]. A rate 1/2 turbo code using SOVA decoding with interleaving size of 900 message bits and $\nu = 2$ can achieve $\mathrm{BER} = 10^{-6}$ at $E_b/N_o \approx 4$ dB after six iterations. Example 7a can achieve $\mathrm{BER} = 10^{-6}$ at $E_b/N_o \approx 4.2$ dB, which is slightly worse than that obtained from the turbo code. However, Example 7a requires significantly less decoding delay than the turbo code.

Finally, by comparing Examples 6 and 7 with Examples 3 and 4, we see that if the same C is used, using the second scheme we can achieve similar error performance with less decoding delay as compared to using the first scheme, even though using the first scheme can achieve a larger lower bound on free distance.

VI. CONCLUDING REMARKS

In this paper, we concentrate on constructing trellis codes with large constraint lengths and hence large free distances. Two schemes for constructing such trellis codes are proposed. A suboptimum decoding with moderate complexity is proposed for each coding scheme. With the suboptimum decoding, the error coefficient is expected to be very large which may reduce the coding gain achieved by the large free distance. In contrast, for

the well-known turbo code, the mechanism of reduced BER majorly comes from its very low error coefficient [8], [9]. Hence, the proposed schemes can achieve good error performance at moderate to high E_b/N_o, while turbo codes can achieve good error performance at low to high E_b/N_o. Even though the decoding delay of each of the proposed schemes is longer than that of the conventional TCM or convolutional codes, it is much shorter than that of the turbo codes.

Error performance of the proposed schemes can be improved by using interleaving and iterative decoding. Such a design will significantly increase the decoding delay and complexity, and hence is not considered in this paper.

APPENDIX
PROOFS OF THEOREMS 2 AND 3

A. Proof of Theorem 2

Consider the righthand side of (8). The condition of $\lambda^{(L)} - 1 \geq m\lambda^{(L-1)} - 1$ implies that

$$\sum_{t=0}^{\lambda^{(L)}-1}\sum_{j=1}^{m} v_j^{(L)}(t)\Delta_j = \sum_{t=0}^{\lambda^{(L)}-1} d_w^{(L)}\left(\hat{v}^{(L)}(t)\right) \geq \sum_{t=0}^{m\lambda^{(L-1)}-1} d_w^{(L)}(\hat{v}^{(L)}(t)). \quad \text{(A1)}$$

Let $t = \tau_j^{(L-1)} + \rho = (m-j)\lambda^{(L-1)} + \rho$. Then the summation index t for $0 \leq t \leq m\lambda^{(L-1)} - 1$ can be replaced by (ρ, j) for $0 \leq \rho < \lambda^{(L-1)}, 1 \leq j \leq m$. For $i < j$, we have $t - \tau_i^{(L-1)} = \rho - (j-i)\lambda^{(L-1)} < 0$. Since $\hat{v}^{(L-1)}(p) = \hat{0}$ for $p < 0$, then $v_i^{(L-1)}(t - \tau_i^{(L-1)}) = 0$. Hence $s_i^{(L-1)}(t) = 0$. It follows from (10) that

$$\begin{aligned} d_w^{(L)}\left(\hat{v}^{(L)}(t)\right) &= d_w^{(L)}\left(\omega^{(L-1)}\left(\hat{s}^{(L-1)}(t)\right)\right) \\ &\geq s_j^{(L-1)}(t)\delta_j^{(L-1)} \\ &= v_j^{(L-1)}(\rho)\delta_j^{(L-1)}. \end{aligned} \quad \text{(A2)}$$

Then

$$\begin{aligned} \sum_{t=0}^{\lambda^{(L)}-1}\sum_{j=1}^{m} v_j^{(L)}(t)\Delta_j &\geq \sum_{\rho=0}^{\lambda^{(L-1)}-1}\sum_{j=1}^{m} v_j^{(L-1)}(\rho)\delta_j^{(L-1)} \\ &= \sum_{t=0}^{\lambda^{(L-1)}-1} d_w^{(L-1)}\left(\hat{v}^{(L-1)}(t)\right). \end{aligned} \quad \text{(A3)}$$

We can similarly derive that $\sum_{t=0}^{\lambda^{(L)}-1}\sum_{j=1}^{m} v_j^{(L)}(t)\Delta_j \geq \sum_{t=0}^{\lambda^{(\ell)}-1} d_w^{(\ell)}(\hat{v}^{(\ell)}(t))$ for $\ell = L-2, \ldots, 1$. Then, $\Delta_{\text{free}} \geq \Delta_{LB}(C^{(L)}, \lambda^{(L)}, \Delta_j) \geq \min_{\overline{v}^{(1)} \in C, \hat{v}^{(1)}(0) \neq \hat{0}} \sum_{t=0}^{\lambda^{(1)}-1} d_w^{(1)}(\hat{v}^{(1)}(t))$. □

B. Proof of Theorem 3

Consider the sequence $\overline{v}$ with $\hat{v}(0) \neq \hat{0}$ and $\hat{v}(t) = \hat{0}$ for $t < 0$. We may divide the sequence $\overline{v}$ into λ disjoint subsequences, $\overline{v}(0), \overline{v}(1), \ldots, \overline{v}(\lambda-1)$, where $\overline{v}(h) = \{\ldots, \hat{v}(-\lambda+h), \hat{v}(h), \hat{v}(\lambda+h), \ldots\}$, $h = 0, 1, \ldots, \lambda-1$. Let $\overline{s}$ and $\overline{z}$ be the corresponding sequences for $\overline{v}$. We then divide each of $\overline{s}$ and $\overline{z}$ into λ subsequences similarly. The subsequence $\overline{z}(h)$ will only depend on $\overline{v}(h)$ among all the λ subsequences of $\overline{v}$. Now we will consider the distance property related to subsequences $\overline{v}(0)$, $\overline{s}(0)$ and $\overline{z}(0)$. Let $z_0 = \omega(\hat{0})$ and $\overline{z}_0 = \{\ldots, z_0, z_0, \ldots\}$. We will prove that $\Delta(\overline{z}(0), \overline{z}_0)$, which is the distance measure between $\overline{z}(0)$ and $\overline{z}_0$, is lower bounded by $\sum_{j=1}^{m} v_j(0)(\Delta_j + \ell_j\Delta_{j-1})$. Then, we can similarly show that the distance measure between $\overline{z}(h)$ and $\overline{z}_0$ is lower bounded by $\sum_{j=1}^{m} v_j(h)(\Delta_j + \ell_j\Delta_{j-1})$. Then, it is easy to see that the bound given in (22) is true. In the following, we first assume $\ell_j > 0$ for $2 \leq j \leq m$. The case for $\ell_j = 0$ will be considered later. Let $\alpha_j = \beta_j - \ell_{j+1}$ for $1 \leq j \leq m-1$, $\alpha_0 = \beta_1 + 1$ and $\alpha_m = 0$. It can be checked that $\alpha_{j-1} = \beta_j + \zeta_j + 1$ for $1 \leq j \leq m$. It is evident that

$$\begin{aligned} \Delta(\overline{z}(0), \overline{z}_0) &\geq \sum_{t=0}^{\beta_1} \Delta(z(t\lambda), z_0) \\ &= \sum_{t=\alpha_m}^{\alpha_{m-1}-1} \Delta(z(t\lambda), z_0) + \sum_{t=\alpha_{m-1}}^{\alpha_{m-2}-1} \Delta(z(t\lambda), z_0) \\ &\quad + \cdots + \sum_{t=\alpha_1}^{\alpha_0-1} \Delta(z(t\lambda), z_0). \end{aligned} \quad \text{(B1)}$$

Consider $\alpha_j \leq t < \alpha_{j-1}$ for $2 \leq j \leq m$. Since $t < \alpha_{j-1}$, then $t - \beta_i < t - (\beta_i - 1) < \cdots < t - (\beta_i - \ell_{i+1}) = t - \alpha_i < 0$ for $i < j$. Since $\hat{v}(p) = 0$ for $p < 0$, from (19), we have $s_i(t\lambda) = v_i((t-\beta_i)\lambda) \oplus v_{i+1}((t-(\beta_i-1))\lambda) \oplus \cdots \oplus v_{i+1}((t-\alpha_i)\lambda) = 0$ for $1 \leq i < j$. Hence $\Delta(z(t\lambda), z_0) \geq s_j(t\lambda)\Delta_j$ for $2 \leq j \leq m$. Moreover, it is clear that $\Delta(z(t\lambda), z_0) \geq s_1(t\lambda)\Delta_1$ for $\alpha_1 \leq t < \alpha_0$. Let

$$Y_j = \sum_{t=\alpha_j}^{\alpha_{j-1}-1} s_j(t\lambda)\Delta_j, \qquad \text{for } 1 \leq j \leq m. \quad \text{(B2)}$$

Then, (B1) implies $\Delta(\overline{z}(0), \overline{z}_0) \geq Y_m + Y_{m-1} + \cdots + Y_1$.

For $p \in \{0, 1\}$, and $q_1, q_2, \ldots, q_k \in \{0, 1\}$, if $k \leq \ell_j$, the condition of $\Delta_j \geq \ell_j\Delta_{j-1}$ implies $p\Delta_j + q_1\Delta_{j-1} + q_2\Delta_{j-1} + \cdots + q_k\Delta_{j-1} \geq (p \oplus q_1)\Delta_{j-1} + \cdots + (p \oplus q_k)\Delta_{j-1}$. Moreover, it can be checked that for $p_i, q_j \in \{0, 1\}$, if $k \leq h$, then

$$\sum_{i=1}^{h} p_i\Delta_j + \sum_{i=1}^{k} q_i\Delta_{j-1} \geq \sum_{i=1}^{k}(q_i \oplus p_i \oplus p_{i-1} \oplus \cdots \oplus p_l)\Delta_{j-1} \quad \text{(B3)}$$

where $l = \max\{i - (\ell_j - 1), 1\}$. Note that each p_i appears in the summation of the right-hand side of (B3) at most ℓ_j times.

Let $W_j = \sum_{i=j}^{m} Y_i$ and $Y'_j = \sum_{i=1}^{\zeta_j} v_j(i\lambda)\Delta_j$ for $1 \le j \le m$. From (18) and $\beta_m = 0$, we have

$$\begin{aligned} W_m = Y_m &= \sum_{t=\alpha_m}^{\alpha_{m-1}-1} s_m(t\lambda)\Delta_m = \sum_{t=0}^{\zeta_m} v_m(t\lambda)\Delta_m \\ &= v_m(0)\Delta_m + Y'_m. \end{aligned} \tag{B4}$$

Then, $W_{m-1} = Y_m + Y_{m-1} = v_m(0)\Delta_m + Y'_m + Y_{m-1}$. From (B3) and the condition of $\zeta_m \ge \zeta_{m-1} + \ell_m = \zeta_{m-1} + \beta_{m-1} - \alpha_{m-1} = (\alpha_{m-2} - 1) - \alpha_{m-1}$, we have

$$\begin{aligned} &Y'_m + Y_{m-1} \\ &\quad = \sum_{t=1}^{\zeta_m} v_m(t\lambda)\Delta_m + \sum_{t=\alpha_{m-1}}^{\alpha_{m-2}-1} s_{m-1}(t\lambda)\Delta_{m-1} \\ &\quad = s_{m-1}(\alpha_{m-1}\lambda)\Delta_{m-1} + \sum_{t=1}^{\zeta_m} v_m(t\lambda)\Delta_m \\ &\qquad + \sum_{t=1}^{\zeta_{m-1}+\ell_m} s_{m-1}((\alpha_{m-1}+t)\lambda)\Delta_{m-1} \\ &\quad \ge h_0\Delta_{m-1} + h_1\Delta_{m-1} + \cdots + h_{\ell_m+\zeta_{m-1}}\Delta_{m-1} \end{aligned} \tag{B5}$$

where

$$h_0 = s_{m-1}(\alpha_{m-1}\lambda) \tag{B6}$$

and for $1 \le i \le \ell_m + \zeta_{m-1}$

$$\begin{aligned} h_i = s_{m-1}((\alpha_{m-1}+i)\lambda) &\oplus v_m(i\lambda) \\ &\oplus v_m((i-1)\lambda) \oplus \cdots \oplus v_m(l\lambda) \end{aligned} \tag{B7}$$

where $l = \max\{i - (\ell_m - 1), 1\}$. It follows from (19) that

$$\begin{aligned} s_{m-1}((\alpha_{m-1}+i)\lambda) &= v_{m-1}((\alpha_{m-1}+i-\beta_{m-1})\lambda) \\ &\quad \oplus v_m((\alpha_{m-1}+i-(\beta_{m-1}-1))\lambda) \\ &\quad \oplus \cdots \oplus v_m((\alpha_{m-1}+i-\alpha_{m-1})\lambda) \\ &= v_m(i\lambda) \oplus \cdots \oplus v_m((i-(\ell_m-1))\lambda) \\ &\quad \oplus v_{m-1}((i-\ell_m)\lambda). \end{aligned} \tag{B8}$$

Since $\hat{v}(p) = 0$ for $p < 0$, from (B6)–(B8), we have

$$h_i = \begin{cases} v_m(0) & \text{for } i = 0, 1, \ldots, \ell_m - 1, \\ v_{m-1}((i-\ell_m)\lambda), & \text{for } i = \ell_m, \ldots, \ell_m + \zeta_{m-1}. \end{cases} \tag{B9}$$

Therefore

$$\begin{aligned} Y'_m + Y_{m-1} &\ge (\ell_m v_m(0) + v_{m-1}(0))\Delta_{m-1} \\ &\quad + \sum_{t=1}^{\zeta_{m-1}} v_{m-1}(t\lambda)\Delta_{m-1} \\ &= (\ell_m v_m(0) + v_{m-1}(0))\Delta_{m-1} + Y'_{m-1}. \end{aligned} \tag{B10}$$

In general, we can replace m in (B5)–(B9) by j for $2 \le j \le m$ to yield

$$Y'_j + Y_{j-1} \ge (\ell_j v_j(0) + v_{j-1}(0))\Delta_{j-1} + Y'_{j-1}. \tag{B11}$$

Thus

$$\begin{aligned} W_1 &= Y_m + Y_{m-1} + \cdots + Y_1 \\ &\ge v_m(0)\Delta_m + Y'_m + Y_{m-1} + \cdots + Y_1 \\ &\ge v_m(0)\Delta_m + (\ell_m v_m(0) + v_{m-1}(0))\Delta_{m-1} \\ &\quad + Y'_{m-1} + Y_{m-2} + \cdots + Y_1 \\ &\ge v_m(0)\Delta_m + (\ell_m v_m(0) + v_{m-1}(0))\Delta_{m-1} \\ &\quad + \cdots + (\ell_2 v_2(0) + v_1(0))\Delta_1 \\ &= v_m(0)(\Delta_m + \ell_m\Delta_{m-1}) + \cdots \\ &\quad + v_2(0)(\Delta_2 + \ell_2\Delta_1) + v_1(0)\Delta_1. \end{aligned} \tag{B12}$$

Note that in the above proof, we assume $\ell_j > 0$ for $2 \le j \le m$. If $\ell_j = 0$ for some j, $2 \le j \le m$, then $\zeta_j = 0$ and $s_{j-1}(t) = v_{j-1}(t - \beta_{j-1}\lambda)$. We have $\alpha_{j-1} = \beta_{j-1} - \ell_j = \beta_{j-1}$ and $\alpha_{j-2} = \beta_{j-2} - \ell_{j-1} = \beta_{j-1} + \zeta_{j-1} + 1$. Hence

$$\begin{aligned} Y'_j + Y_{j-1} &= \sum_{i=1}^{\zeta_j} v_j(i\lambda)\Delta_j + \sum_{i=\beta_{j-1}}^{\beta_{j-1}+\zeta_{j-1}} s_{j-1}(i\lambda)\Delta_{j-1} \\ &= 0 + \sum_{i=0}^{\zeta_{j-1}} v_{j-1}(i\lambda)\Delta_{j-1} \\ &= v_{j-1}(0)\Delta_{j-1} + Y'_{j-1}. \end{aligned} \tag{B13}$$

Hence (B11) and (B12) still hold. Therefore, $\Delta(\overline{z}(0), \overline{z}_0)$ is lower bounded by $\sum_{j=1}^{m} v_j(0) \cdot (\Delta_j + \ell_j\Delta_{j-1})$. Then, Δ_{free} is lower bounded by (22). □

REFERENCES

[1] G. Hellstern, "Coded modulation with feedback decoding trellis codes," in *Proc. IEEE Int. Conf. on Communications*, Geneva, Switzerland, May 1993, pp. 1071–1075.

[2] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–66, Jan. 1982.

[3] J. Y. Wang and M. C. Lin, "On constructing trellis codes with large free distances and low decoding complexities," *IEEE Trans. Commun.*, vol. 45, pp. 1017–1020, Sept. 1997.

[4] G. Pottie and D. Taylor, "Multilevel codes based on partitioning," *IEEE Trans. Inform. Theory*, vol. 35, pp. 87–98, Jan. 1989.

[5] J. Hagenauer, "Source-controlled channel decoding," *IEEE Trans. Commun.*, vol. 43, pp. 2449–2457, Sept. 1995.

[6] C. Chang and K. Yao, "Systolic array processing of the Viterbi algorithm," *IEEE Trans. Inform. Theory*, vol. 35, pp. 76–86, Mar. 1989.

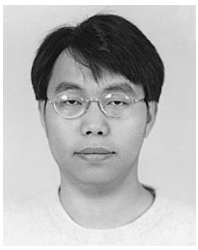
[7] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, pp. 1261–1271, Oct. 1996.

[8] L. C. Perez, J. Segher, and D. J. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Nov. 1996.

[9] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.

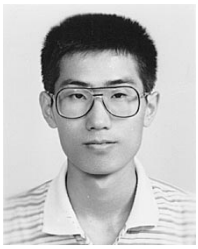
Mao-Chao Lin (S'86–M'87) was born in Taipei, Taiwan, R.O.C., on December 24, 1954. He received the B.S. and M.S. degrees in electrical engineering from National Taiwan University (NTU), Taipei, in 1977 and 1979, respectively, and the Ph.D. degree in electrical engineering from the University of Hawaii, Manoa, in 1986.

From 1979 to 1982, he was an Assistant Scientist at the Chung-Shan Institute of Science and Technology, Lung-Tan, Taiwan. He is currently with the Department of Electrical Engineering, NTU, Taipei, as a Professor. His research interest is in the area of coding theory.



Yeong-Luh Ueng was born in Taiwan, R.O.C., on July 2, 1975. He received the B.S. degree in electrical engineering and the M.S. degree in communication engineering from National Taiwan University (NTU), Taipei, in 1997 and 1999, respectively. He is now pursuing the Ph.D. degree at the Graduate Institute of Communication Engineering, NTU.

His research interest is in the area of coding theory.



Jia-Yin Wang was born in Tainan, Taiwan, R.O.C., on December 31, 1968. He received the B.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, in 1991 and 1997, respectively.

He is currently with the Broadband Network Systems Department, Computer and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan, as a Section Manager. His research interests include coding theory, information theory and voice over IP technology.