Reprinted from pp. 148-157 of Proceedings of the Int. Conf. on Info. Th. and Systems, NTG-Fachberichte vol. 65, Berlin, Sept. 18-20, 1978.

FOUNDATION AND METHODS OF CHANNEL ENCODING

James L. Massey
Department of System Science
University of California, Los Angeles

1. Introduction

Perhaps no other discovery has given such impetus to the practical applications of coding as has that of the Viterbi algorithm [1] for the maximum-likelihood decoding of convolutional codes. But the full understanding of this decoding procedure awaited Forney's characterization of the code words by a "trellis." More recently, versions of the Viterbi algorithm and trellises have appeared in such disparate applications as the removing of intersymbol interference and the phase-continuous forms of frequency modulation. It is becoming apparent that "trellises" are much more fundamental than anyone had originally expected — even if no one has at yet said precisely what a "trellis" is.

Although this paper is intended as an "overview" of channel coding techniques, it is also intended to argue the novel thesis that the "trellis" is the foundation for channel coding in the sense that the trellis associated with a code is the primary determiner of the necessary effort for maximum-likelihood decoding. To support this thesis, we must of course say precisely what we mean by a "trellis" and by a "trellis code"; this we do in the next two sections. Then, in Section 4, we show how to bring various types of codes into their trellis formulations. Along with old ideas, we offer a few surprises that suggest that the power of the trellis approach is greater than generally supposed. Our aim has been to be as general as necessary to gain the full advantage offered by "trellises," but without seeking generality as an end in itself.

2. Trellis Definitions

We now give a precise definition of what past researchers seem to have meant by a "trellis." A trellis is a directed graph such that: (1) there is a unique node (the root) having no entering branches; (2) every other node can be reached by at least one directed path from the root; (3) all the directed paths from the root to any other node have the same length (i.e., the same number of branches); and (4) properties (1), (2) and (3) also hold when all branch directions are reversed. We shall call the root node of the reversed graph the toor node of the trellis. The graph in Fig. 1(a) is a trellis; but that in Fig. 1(b) is not because it violates (3), and that in Fig. 1(c) is not because it violates (4).

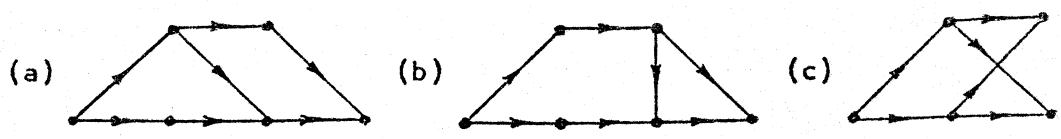

Fig. 1 Examples of: (a) a trellis; (b) and (c) non-trellises.

Because we shall be interested only in directed paths, we shall hereafter simply say "path" but shall always mean directed path. By the depth of a node in a trellis, we mean the length of the paths from the root to that node. We shall also follow, hereafter, the usual convention of drawing trellises so that all the nodes at each depth are arranged along a vertical line and so that all branch directions are from left to right with the direction arrows omitted.

We now specialize to the type of trellis that is the foundation for "trellis codes." For simplicity, we consider only binary codes and their corresponding trellises in this paper. The appropriate trellis for q-ary codes is defined merely by replacing 2 with q in the following definition. An (L,T) binary trellis is a trellis such that: (i) the toor node is at depth L + T; (ii) exactly two branches leave each node at depth i for $0 \leq i < L$; (iii) exactly one branch leaves each node at depth i for $L \leq i < L + T$; and (iv) properties (ii) and (iii) also hold when all branch directions are reversed. We shall refer to L as the dividing length of the trellis and to T as the tail size of the trellis. The trellis in Fig. 1(a) is not an (L,T) binary trellis; but those in Fig. 2(a) and Fig. 2(b) are (L,T) binary trellises for (L,T) = (3,1) and (L,T) = (3,2), respectively.

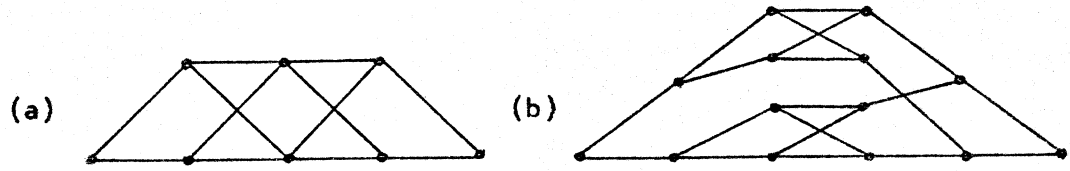

Fig. 2. Examples of: (a) a (3,1) trellis; (b) a (3,2) trellis.

The following are some easy consequences of our definition of an (L,T) binary trellis:

Proposition 1: $T \leq L$.

Proposition 2: There are 2^i nodes at depth i and at depth L + T − i for $0 \leq i < T$; there are 2^T nodes at depth i for $T \leq i < L$.

Proposition 3: There are 2^L distinct paths (i.e., paths differing in at least one branch) from the root to the toor.

3. Trellis Codes

3.1. Code Definitions

Let n_j, $1 \leq j \leq L + T$, be nonnegative integers. Suppose that each branch stemming from each node at depth j − 1 in an (L,T) binary trellis is labelled with n_j digits so that no two paths from the root node to the toor node are labelled with the same sequence of $n = n_1 + n_2 + \ldots + n_{L+T}$ binary digits. Then the 2^L sequences of length n on the 2^L paths from the root node to the toor node form a block code of rate R = L/n. We define such a code to be an $(L,T;n_1,n_2,\ldots n_{L+T})$ trellis code.

Figures 3(a), 3(b), and 3(c) give examples of a (2,1;1,2,1) trellis code, a (2,2;0,4,0,0) trellis code, and a (2,2;1,1,1,1) trellis code, respectively. But, in fact, all three codes are the same block code, namely the linear (parity check) code with generator matrix

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

We see then that there is some freedom as to what type of trellis code we choose to consider a given code to be. We shall soon examine the significance of such freedom. The trellis code in Fig. 3(b) may appear "trivial." but it does illustrate the fact we do permit $n_j = 0$ — moreover, we shall see that the "empty

branches" have decoding significance.

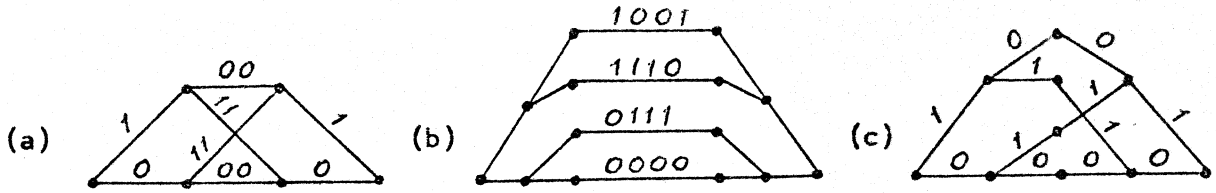

Fig. 3. Examples of: (a) a (2,1;1,2,1) trellis code; (b) a (2,2;0,4,0,0) trellis code; (c) a (2,2;1,1,1,1) trellis code.

3.2. Decoding Implications

The principal significance of the trellis formulation of a code is that the trellis specifies how the code can be decoded by the Viterbi algorithm (VA) — the somewhat parochial name by which coding theorists refer to "dynamic programming" in honor of the fact that Viterbi was the first to apply the algorithm in a coding application, namely to the decoding of convolutional codes [1]. The significance of the VA is that it is a maximum-likelihood decoding method that can be used with "soft-decision" demodulation on any memoryless channel (or any channel whose memory is destroyed by interleaving, at some cost in achievable performance).

Since several tutorial treatments of the VA are available [2]-[4], we limit ourselves here to a brief discussion of the VA, emphasizing the implications of the trellis structure for the necessary decoding effort.

Let $\underline{y} = [y_1, y_2, \ldots y_n]$ represent the n-tuple of demodulator output symbols, and let $\underline{x} = [x_1, x_2, \ldots x_n]$ denote an arbitrary code word in an $(L,T;n_1,n_2,\ldots n_{L+T})$ trellis code. Suppose also that one can define a "metric," $m(x,y)$, between the binary modulating bit x and the demodulator output digit y such that the decoding rule, "choose the code word $\underline{x}$ that maximizes $m(x_1,y_1)+m(x_2,y_2)+m(x_n,y_n)$," is maximum-likelihood (ML). [This is always the case for a memoryless channel where it suffices to take $m(x,y) = \log P(y|x)$. More generally for a memoryless channel, one may take $m(x,y) = c_1 \log P(y|x)+c_2$, where c_1 and c_2 are chosen for convenience subject only to the restriction that c_1 be positive.] Then ML decoding of the trellis code can be accomplished as follows: use the received digits (i. e., the components of $\underline{y}$) to compute the metric for both paths from the root to each node at depth T; compare these paths and retain only the better—there now remain only two paths in the trellis to each node at depth T+1—again compare their metrics and retain only the better path and its metric; etc. Finally, when one has retained only the better path to the toor (i.e., to the unique node at depth L+T), one has the ML decoded path—and this method for finding it is precisely the VA.

We now relate the complexity of ML decoding by the VA to the parameters of the trellis code. We note first that the number of metric comparisons, N_c, is just the number of nodes in the trellis that have two entering branches. It follows, from parts (ii) and (iv) of the definition of an (L,T) trellis and from proposition 2, that

$$N_c = \sum_{i=0}^{T-1} 2^i + \sum_{i=T}^{L-1} 2^T = (L-T+1)2^T - 1. \qquad (1)$$

Next, we note that the number of digit metrics, N_m, that need to be found in order to compute the path metrics is just the total number of digits on all branches of the trellis. From part (ii) of the trellis definition, from proposition 2, and from the definition of an $(L,T;n_1,n_2,\ldots n_{L+T})$ trellis code, it follows that

$$N_m = \sum_{j=1}^{T} (n_j + n_{L+T+1-j}) 2^j + 2^{T+1} \sum_{j=T+1}^{L} n_j \tag{2}$$

where the second sum is 0 when L=T. Finally, we note that the number of accumulators, N_a, (i.e., the maximum number of metrics which must be stored at any time to permit the progression from one depth to the next in the course of applying the VA) is simply

$$N_a = 2^T. \tag{3}$$

Example 1: We have already noted that the three trellis codes in Fig.3 are different trellis code formulations of the same block code. When the VA is applied according to the (2,1;1,2,1) trellis code formulation of Fig.3(a), we find

$$N_c = 3,\ N_m = 12 \text{ and } N_a = 2.$$

The (2,2;1,3,0,0) formulation of Fig.3(b) gives

$$N_c = 3,\ N_m = 14 \text{ and } N_a = 4.$$

(We now see that the "empty branches" in Fig.3(b) are needed to give the correct count of the number of comparisons that will be required to find the largest of the four metrics that will be stored when the VA has reached the nodes at depth 2 in this trellis.) Finally, the (2,2;1,1,1,1) formulation in Fig.3(c) gives

$$N_c = 3,\ N_m = 12 \text{ and } N_a = 4.$$

It is an accident, not a general property of different trellis code formulations of the same code, that the VA requires 3 comparisons for all three of our trellis formulations. The (2,1;1,2,1) formulation of Fig.3(a) is clearly the "best", of the three formulations in Fig.3, to use for decoding by the VA as it requires the smallest number of accumulators and is tied with the formulation of Fig.3(b) for the smallest number of digit metrics required.

To the first-order, the quantities N_c,N_m and N_a determine the complexity of decoding a trellis code by the VA. Naturally, there are some other considerations such as the ease by which the digits on the branches of the trellis can be generated and the simplicity of the rules for determining which nodes are the successors at the next depth of a given node -- these are considerations of the complexity of the encoder for the given trellis code.

3.3 Encoding Implications

Suppose that we now write an arbitrary code word in an $(L,T;n_1,n_2,\ldots n_{L+T})$ trellis code as

$$\underline{v} = [v_1, v_2, \ldots v_{L+T}],$$

where v_j is a binary n_j-tuple. Suppose we write the information sequence as the binary (L+T)-tuple

$$\underline{u} = [u_1, u_2, \ldots u_{L+T}]$$

where each u_j is a binary digit and where we adopt the convention

$$u_j = 0 \text{ for } j = L+1, L+2, \ldots L+T$$

to account for the fact that there are only L true information bits. Suppose finally that we write the sequence of nodes through which the code word passes as $\sigma_1, \sigma_2, \ldots, \sigma_{L+T+1}$ where, by necessity, σ_1 is the root and σ_{L+T+1} is the toor of the trellis. Then, the encoding operation can be described by the equations

$$v_j = \lambda_j(\sigma_j, u_j), \quad 1 \le j \le L+T \tag{4a}$$

and

$$\sigma_{j+1} = \delta_j(\sigma_j, u_j), \quad 1 \le j \le L+T. \tag{4b}$$

In fact, we see that a given block code admits a formulation as an $(L,T;n_1,n_2,\ldots n_{L+T})$ trellis code if and only if it can be encoded by an automaton of the type specified in (4a) and (4b). In usual automata-theoretic language, λ_j and δ_j are the "output function" at "time" j and the "next-state function" at "time" j, respectively -- the subscript j reflects the fact that the automaton may be "time-varying". Here σ_j is the "state" at time j or, equivalently, the node at depth j-1 in the trellis; u_j is the "input" at time j; and v_j is the "output" at time j. Note that our formulation permits the sets in which u_j, v_j and σ_j take values to be sets which are time-dependent.

In the remainder of this paper, we consider the trellis formulation of various block codes by considering their encoding in the manner of (4a) and (4b). For simplicity as well as practicality, we restrict our attention to linear codes.

4. Trellis Code Formulation of Linear Codes

4.1 General Codes

Every block code (linear or nonlinear) of length n with 2^L codewords admits a trivial formulation as an $(L,T;n_1,n_2,\ldots n_{L+T})$ trellis code with T=L, $n_T=n$ and $n_j=0$ for $j \neq T$; namely the trellis code in which the entire code words are the labels on the branches stemming from the 2^L nodes at depth L-1. (Fig.3(b) illustrates precisely this formulation.) This "trivial" trellis code formulation corresponds to the "naive" way to perform ML decoding, namely by computing the total metric for each of the 2^L codewords (which requires the finding of $N_m=n2^L$ digit metrics) and then choosing the code word with the largest metric (which requires $N_c=2^L-1$ comparisons of the metrics stored in the $N_a=2^L$ accumulators.)

4.2 Systematic Block Codes

Consider an (n,k) linear code and suppose that the code is systematic, i.e., that it has a generator matrix G of the form $G = [I:P_0]$, where I is the k x k identity matrix, and where P_0 is some k x (n-k) matrix whose <u>rank</u> we denote as r. Next, define the positive integers c_i and the matrices P_i recursively for i = 1,2,...r by the rules that (1) c_i is the smallest number of leading columns which, when removed from P_{i-1}, reduce its rank to r-i, and (2) P_i is the matrix P_{i-1} after removal of its first c_i columns. We shall now prove:

<u>Theorem</u>: An (n,k) linear systematic code can be formulated as an $(L,T;n_1,n_2,\ldots n_{L+T})$ trellis code such that L=k, T=r, and

$$n_j = \begin{cases} 1, & j=1,2,\ldots k \\ c_{j-k}, & j=k+1,\ k+2,\ldots,\ k+r. \end{cases} \tag{5}$$

Proof: Let $\underline{u} = [u_1,u_2,\ldots u_k]$ be the information vector and $\underline{v} = [v_1,v_2,\ldots v_{k+r}]$ be the code word where the length n_j of v_j is specified by (5). Since $\underline{v} = \underline{u}G = \underline{u}[I : P_0]$, it follows that

$$v_j = u_j, \quad j=1,2,\ldots k \tag{6a}$$

and

$$[v_{k+i},\ldots,v_{k+r}] = [u_1,\ldots,u_k]P_{i-1}. \tag{6b}$$

From (6a) and (6b), we see that (4a) and (4b) are satisfied by the choices:

$$\sigma_i = [u_1,u_2,\ldots u_{i-1},0,\ldots 0]P_0, \quad i=1,2,\ldots k+1 \tag{7a}$$

since this gives

$$\sigma_{j+1} = \sigma_j + [0,\ldots,0,u_j,0,\ldots 0]P_0 \tag{8}$$

together with

$$v_j = u_j$$

for j=1,2,...k; and

$$\sigma_{k+i+1} = [u_1,u_2,\ldots u_k]P_i, \qquad i=1,2,\ldots r \tag{7b}$$

since this gives both

$$\sigma_{j+1} = (\sigma_j \text{ with first } c_{j-k} \text{ components deleted})$$

and

$$v_j = (\text{first } c_{j-k} \text{ components of } \sigma_j)$$

for j=k+1,k+2,...,k+r. To complete the proof it remains only to observe that, because the rank of P_i is r-i, (7b) implies that there are exactly 2^{r-i} distinct values of σ_{k+i+1}, which is precisely the number of nodes at depth k+i in a (k,r) binary trellis for i=1,2,...r, while (7a) implies there are at most 2^r distinct values of σ_i for i=1,2,...k+1.

In case the formality of our proof of Theorem 1 has obscured its basic simplicity, we remark that (7a) specifies the choice of state in the "dividing part" of the trellis as the vector of parity digits that would result if all future information bits were 0, and (7b) specifies the choice of state in the "tail part" of the trellis as the vector of remaining parity digits.

Example 2: For the (5,3) systematic code with

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix},$$

we find r=2=n-k so that $c_1=c_2=1$ and $n_i=1$ for i=1,2,3,4,5. The (3,2;1,1,1,1,1) trellis formulation for this code is shown in Fig.4, where we have explicitly shown the choice of state at every node in the trellis.

Note that decoding by the VA will require $N_a=4$ accumulators, $N_c=7$ comparisons, and the use of $N_m=20$ digit metrics. From (1), we can see that it is accidental that $N_c=2^L-1=2^k-1=7$, the same as would have been the case for the "trivial" trellis code formulation of Section 4.1; however the latter would require 8 accumulators and 40 digit metrics.

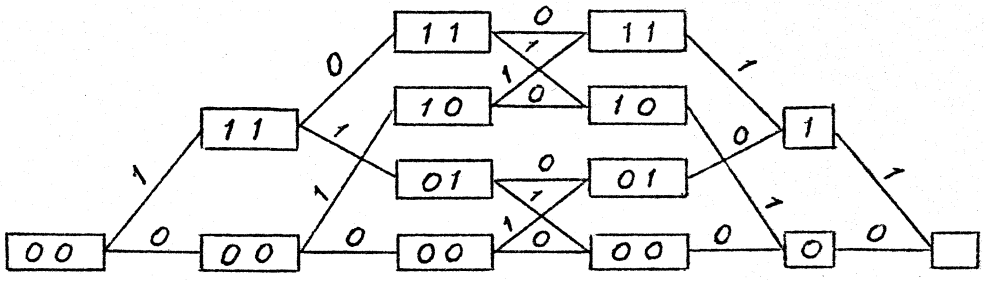

Fig. 4. The (3,2;1,1,1,1,1) trellis code of example 2.

In fact, whenever the columns of P_0 are linearly independent (or, equivalently, whenever $r=n-k$), which requires in particular that $n-k\leq k$, we see that the above procedure will lead to the formulation of the (n,k) systematic code as a (k,n-k;1,1,...1) trellis code. This special case is in fact precisely the "efficient maximum likelihood decoding of linear block codes using a trellis" proposed recently by Wolf [5] and the code in example 2 is the same as an example used by Wolf. Wolf's paper was the major motivation for our seeking a unified approach to "trellis" encoding and decoding.

As noted by Wolf, systematic codes which are also cyclic are particularly convenient for trellis decoding, since then the "next state" operation of (8) can equivalently be implemented by a simple shift-register circuit [5].

The more general formulation that we have developed here leads also to interesting results in the case where the columns of P_0 are linearly dependent.

Example 3: For the (5,2) systematic code with

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix} ,$$

we find $r=2$, and $c_1=2$ and $c_2=1$. The (2,2;1,1,2,1) trellis code formulation for this code is shown in Fig.5. Note that decoding by the VA will require $N_a=4$ accumulators, $N_c=3$ comparisons, and the use of $N_m=16$ digit metrics.

We remark that the number of accumulators, $N_a=2^r$, required for the trellis decoding of a systematic code by our procedure satisfies

$$r \leq \min\{k,n-k\}, \tag{9}$$

since the rank of the matrix P_0 cannot exceed the number of its rows, k, nor the number of its columns, n-k.

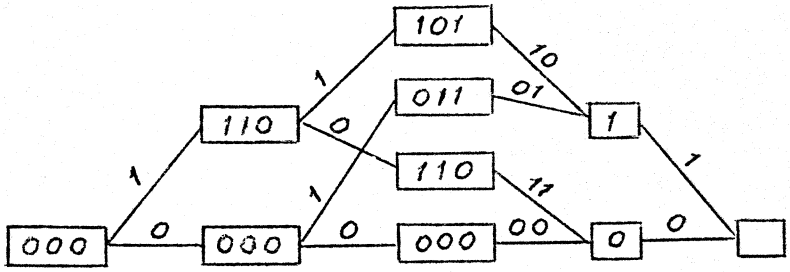

Fig. 5. The (2,2;1,1,2,1) trellis code of example 3.

4.3 Non-Systematic Block Codes

Rather that their "systematicity" being the reason that systematic block linear codes can be decoded "efficiently" by a trellis -- as one might erroneously infer from the preceding section -- we shall now show that "non-systematicity" is the real touchstone to "efficient" trellis decoding. We illustrate first with an example.

Example 4: Consider the "equivalent" non-systematic (5,3) code obtained from the systematic code of example 2 by interchanging positions 1 and 2 and also positions 3 and 4. This non-systematic code then has the generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} ,$$

where the "information bits" are now in positions 1, 2 and 4. This non-systematic code admits the (3,1;1,2,1,1) trellis code formulation shown in Fig. 6,

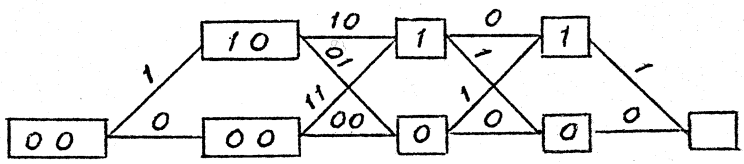

Fig. 6. The (3,1;1,2,1,1) trellis code of example 4.

as can be easily checked by comparing the 8 paths through the trellis to the eight code words. Note that decoding by the VA requires only N_a=2 accumulators, N_c=5 comparisons, and N_m=18 branch metrics -- whereas the equivalent systematic code required N_a=4, N_c=7 and N_m=20. This simplification is quite striking over what appeared to be a very "efficient" trellis decoding!

We now formalize the technique by which the simply decoded non-systematic code

of example 4 was obtained. Note that the code word in this example can be written

$$\underline{v} = [(u_1),(u_2,p_1),(u_3),(p_2)] \tag{10}$$

where u_1, u_2 and u_3 are the information bits, where p_1 and p_2 are the parity bits, and where the parentheses indicate the divisions for v_1, v_2, v_3 and v_4. The key is that we may take our choice of state, σ_j, to be the vector of parity bits in $v_{j+1}, v_{j+2},\ldots v_{L+T}$ as determined by the information bits in $v_1,v_2,\ldots v_j$. In general, we can write this choice of state as

$$\sigma_j = [u_1,u_2,\ldots u_j]M_j, \qquad j=1,2,\ldots k. \tag{11}$$

Then the maximum number of states at any depth in the trellis will be 2^T, where

$$T = \max_{1\le j\le k} \operatorname{rank}(M_j). \tag{12}$$

For instance, in example 4 we have

$$[p_1,p_2] = [u_1,u_2,u_3]\begin{bmatrix}1 & 0\\ 1 & 1\\ 0 & 1\end{bmatrix}$$

so that (10) and (11) specify

$$M_1 = [1\ 0], \quad M_2 = \begin{bmatrix}0\\1\end{bmatrix}, \quad M_3 = \begin{bmatrix}0\\1\\1\end{bmatrix}.$$

Since all of these matrices have rank 1, it follows from (12) that this is a trellis code with T=1.

The art of trellis decoding of a linear code can thus be seen to be that of re-arranging the order of digits in the code word to obtain a non-systematic code for which T, as determined by (12), is as small as possible. Minimizing T minimizes the number of accumulators $N_a=2^T$ required for decoding by the VA and, as can be seen from (1) and (2), also minimizes N_c and usually minimizes N_m as well.

4.4 Convolutional Codes

We end our study of trellis coding with the application where trellis coding was first considered, namely with convolutional codes.

A convolutional (b,1) code of length L and memory T is a special type of trellis code such that one may write

$$v_j = \sum_{i=0}^{T} u_{j-i}g_i, \qquad j=1,2,\ldots L+T \tag{13}$$

where each g_i is a binary b-tuple and where, in addition to our earlier convention that $u_j=0$ for $j>L$, we now impose the further convention that $u_j=0$ for $j<1$. Equation (13) suggests a very simple choice of state, namely

$$\sigma_j = [u_{j-1},u_{j-2},\ldots,u_{j-T}], \tag{14}$$

the vector of the past T information bits. The "next-state" rule thus requires

only a simple shift-register (without feedback) that also can be used in a simple computation of the branch digits according to (13). These encoding considerations make the implementation of decoding by the VA particularly attractive for convolutional codes as compared to the "less regular" trellis codes that we have treated in this paper. On the other hand, the results in Section 4.3 suggest that there well may be classes of non-systematic codes whose decoding by the VA could be as simple, or even simpler, than that for convolutional codes giving similar channel performance. It seems premature to conclude that convolutional codes are the most practical form of trellis code. Moreover, it seems exciting to speculate that "irregular" trellis codes might prove more amenable to algebraic constructions than have the algebra-resisting convolutional codes.

5. References

[1] A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", IEEE Trans. Info. Th., vol. IT-13, pp. 260-269, April 1967.

[2] G.D. Forney, Jr., "The Viterbi algorithm", Proc. IEEE, vol. 61, pp. 268-276, March 1973.

[3] A.J. Viterbi, "Convolutional codes and their performance in communication systems", IEEE Trans.Com.Tech., vol. COM-19, pp. 751-772, Oct. 1971.

[4] J.L. Massey, "Error bounds for tree codes, trellis codes and convolutional codes with encoding and decoding procedures", pp. 1-57 in Coding and Complexity, Ed. G. Longo, CISM Courses and Lectures no. 216. Vienna: Springer-Verlag, 1976.

[5] J.K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis", IEEE Trans.Info.Th., vol. IT-24, pp. 76-80, Jan. 1978.