A Fast Algorithm for Computing Distance Spectrum of Convolutional Codes

MATS CEDERVALL, MEMBER, IEEE, AND ROLF JOHANNESSON, MEMBER, IEEE

***Abstract* — A fast algorithm for searching a tree (FAST) is presented for computing the distance spectrum of convolutional codes. The distance profile of a code is used to limit substantially the error patterns which have to be searched. Our algorithm can easily be modified to determine the number of nonzero information bits of an incorrect path as well as the length of an error event. For testing systematic codes we give a faster version of the algorithm. FAST is much faster than the standard bidirectional search. On a μ-VAX-computer we verified $d_\infty = 27$ for a rate $R = 1/2$, memory $M = 25$ code in 37 s of CPU time. Extensive tables of rate $R = 1/2$ encoders are given. Several of the listed encoders have distance spectra superior to those of any previously known codes of the same rate and memory. Finally, Massey's old conjecture that a rate $R = 1/2$ systematic convolutional code of memory $2M$ will perform as well as a nonsystematic convolutional code of memory M is given striking support.**

I. Introduction

Over the past decade there has been a significant increase in using sequential decoding to achieve reliable communication. We can expect that the demand for communication with extremely low error probability will continue to grow. It is well-known [1], [2] that the distance spectrum is the main factor in determining the event error probability when maximum-likelihood decoding (or near maximum-likelihood) is used for a convolutional code. It has also been observed [2]–[4] that an optimum distance profile (ODP) is desirable for a good computational performance with sequential decoding. Thus it is important to find methods for constructing convolutional codes with both a good distance spectrum and a good distance profile.

So far there has been little success in finding very good convolutional codes by algebraic methods. Most codes used in practice were found by computer search. It is a simple task to determine the distance profile, but an evaluation of the spectrum needs a search among prohibitively many paths in the code tree and becomes practically impossible except for rather small code memories. Most algorithms for finding distances presented in the literature [5]–[9] are either too slow or use too much computer memory when applied to long, good codes. In the latter group are algorithms that invert the state transition matrix. The rank of the state transition matrix for the encoder grows exponentially with the code memory. An obvious way to save computer memory is to perform a sequential search in the code tree.

In this paper we show how the distance profile can be utilized to reduce the search dramatically to relatively small regions of error patterns. In Section II some introductory concepts are given. An illustrative example is given in Section III. In Section IV we give a brief description of the fast algorithm for searching trees (FAST) for determining the spectrum parameters [10]. Section V contains some modifications of the algorithm. In Section VI we discuss how to find good encoders. Finally, Section VII contains the results.

Manuscript received June 1, 1988; revised February 9, 1989. This work was supported in part by the Swedish National Board for Technical Development under Grant 85-3303. The material in this paper was partially presented at the International Workshop on Convolutional Codes, Multi-User Communications, Sochi, USSR, May 31–June 5, 1983, and at the International Symposium on Information Theory, St. Jovite, PQ, Canada, September 26–30, 1983.

The authors are with the Department of Information Theory, University of Lund, P.O. Box 118, S-221 00 Lund, Sweden.

IEEE Log Number 8931522.

II. Introductory Concepts

For simplicity, we limit our discussion to binary codes of rate $R = 1/2$. The extension to rate $R = 1/c$ is trivial and to rate $R = b/c$ straightforward.

In a rate $R = 1/2$ binary convolutional code, the information sequence $i_0, i_1, i_2, \cdots$ is encoded as the sequence

$$t_0^{(1)} t_0^{(2)}, t_1^{(1)}, t_1^{(2)}, t_2^{(1)} t_2^{(2)}, \cdots,$$

where

$$t_u^{(k)} = \sum_{j=0}^{M} i_{u-j} g_j^{(k)} \pmod 2, \tag{1}$$

$k = 1, 2$. The parameter M is the code *memory* and

$$G^{(k)} = \left[g_0^{(k)}, g_1^{(k)}, \cdots, g_M^{(k)}\right]$$

for $k = 1, 2$ are the code generators. The code is systematic when $G^{(1)} = [1, 0, \cdots, 0]$.

We shall find it convenient to write

$$\boldsymbol{t}_{[0,n]} = t_0^{(1)} t_0^{(2)}, t_1^{(1)} t_1^{(2)}, \cdots, t_n^{(1)} t_n^{(2)}$$

for the encoded path containing the first $n+1$ "branches" of the encoded sequence. The encoded path $\boldsymbol{t}_{[0,M]}$ is called the *first constraint length* of the code. The jth order *column distance* [11] d_j is the minimum Hamming distance between some $\boldsymbol{t}_{[0,j]}$ resulting from an information sequence with $i_0 = 1$ and some $\boldsymbol{t}_{[0,j]}$ with $i_0 = 0$. By linearity, d_j is also the minimum of the Hamming weights of the paths $\boldsymbol{t}_{[0,j]}$ resulting from information sequences with $i_0 = 1$. The quantities d_M and d_∞ are called the *minimum* distance and *free* distance, respectively, of the code. The

$(M+1)$-tuple

$$d = [d_0, d_1, \cdots, d_M]$$

is called the *distance profile* [3]. A code is said to have a distance profile d superior to a distance profile d' of another code of the same memory M, when there is some l such that

$$d_j = \begin{cases} = d_j', & j = 0, 1, \cdots, l-1 \\ > d_j', & j = l. \end{cases}$$

Let $n(d_\infty + i)$ denote the number of weight $d_\infty + i$ paths which depart from the all-zero path at the root node in the code tree and do not reach the zero state until their termini. We call $n(d_\infty + i)$ the $(i+1)$th *spectral component*. The sequence

$$n(d_\infty + i), \qquad i = 0, 1, 2, \cdots,$$

is called the *distance spectrum* of the code.

To compute the number of paths with a given distance d to the all-zero path we exploit the linearity of the code and count the number of weight d sequences stemming from the zero state and terminating for the first time at the zero state. Suppose we are in an arbitrary *node* in the tree and we have produced channel symbols whose total weight is W_t. Then, in each subtree stemming from this node we have to spend the weight d minus W_t. Hence, let us label each node with the *state* of the encoder and the remaining weight, i.e., $W = d - W_t$.

Let $S = [s_1, s_2, \cdots, s_M]$, where the *state variables* $s_n = i_{j-n}$ for $n = 1, 2, \cdots, M$, and $i_k = 0$ for $k < 0$, denote the state of the convolutional encoder. From each state we have two successor states, $S_0 = [0, i_{j-1}, \cdots, i_{j-M-1}]$ and $S_1 = [1, i_{j-1}, \cdots, i_{j-M-1}]$, corresponding to information symbol $i_j = 0$ and 1, respectively.

For given code generators we can of course use the state of a node to determine the weights, w_0 and w_1 of the branches stemming from that node. By using these branch weights together with the node weight W, we can determine the two new node weights $W_0 = W - w_0$ and $W_1 = W - w_1$ (see Fig. 1).

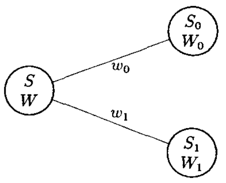

Fig. 1. Successor nodes at time j.

When searching for a path in the code tree with a given weight, we explore a subtree if and only if the new node weight W_{i_j} is nonnegative and if the state of the new node S_{i_j} differs from the zero state. Let us arbitrarily give *priority* to the zero branch whenever we have to select between two new possible nodes.

A straightforward algorithm for determining the number of paths of a given weight d can be formulated as follows. Start at state $S = [1, 0, \cdots, 0]$ with weight $W = d - d_0$ and move forward in the code tree. If $S = [0, 0, \cdots, 1]$ and $W_0 = 0$, then increase the path counter. If the new node weight is negative or if the new node is in its zero state, then we move backwards. Thus we have to remember all of the previous information symbols so that we can move backwards until we find a new "one"-branch with a nonnegative node weight. Then we move forward again. A *stop condition* appears when we reach the root.

This basic algorithm is of course very time-consuming. To measure the performance of the algorithm, we count the total number of nodes visited. Each visit to a node, regardless of whether we have been there before or not, is counted as a visit.

As an example, we use the basic algorithm to verify that the memory $M = 3$ code with generators $G^{(1)} = [1,1,1,1]$ and $G^{(2)} = [1,0,1,1]$, or since we prefer octal notation $G^{(1)} = 74$ and $G^{(2)} = 54$, has one path of weight $d_\infty = 6$. The code tree explored by the algorithm is shown in Fig. 2. As many as 121 nodes are visited.

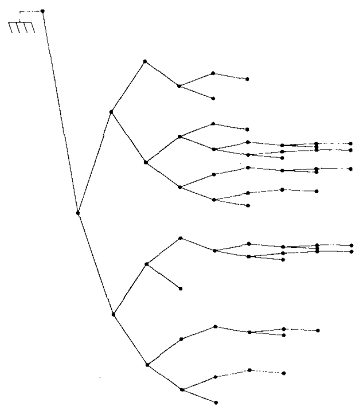

Fig. 2. Code tree explored by basic algorithm.

In the next section we shall use the same example to show how we can obtain a substantial reduction in the number of nodes we have to visit.

III. An Illustrative Example

Our $M = 3$ code with generator $G = [G^{(1)}, G^{(2)}] = [74, 54]$ has (optimum) distance profile $d = [2, 3, 3, 4]$. In Fig. 3 we show only the part of its trellis which contains the weight 6 $(= d_\infty)$ path. This path corresponds to the information

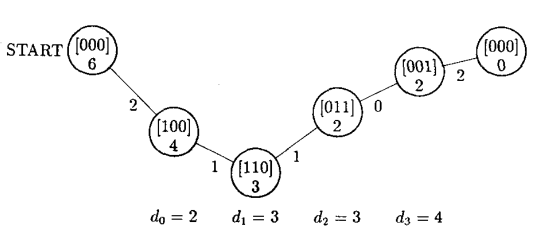

Fig. 3. Example of weight d_∞ path.

sequence 11000, i.e., the encoded sequence $t_{[0,4]} = 11,01,01,00,11$.

Since the column distance is the *minimum* of the Hamming weights of the paths with $i_0 = 1$, the distance profile can be used as a lower bound to the *decrease of the node weight* along the path. In steps 1, 2, and 4 in Fig. 3 this bound is tight. If we traverse this path in the opposite direction we will of course get the same total weight but different node weights.

In Fig. 4 we can use the distance profile as a lower bound to the node weights along the path. Notice that if a node has weight less than this bound, then every path leading backward to the all-zero state will give a negative node weight at the root node. For example, if the node weight in state [001] is less than $d_3 = d_M = 4$ we must not extend this node when we are traversing the trellis backwards. More generally, the weight of a backward path stemming from a node in state $S \neq [0,0,\cdots,0]$, starting with a one-branch, and eventually leading to the root node (zero state) is lower-bounded by d_M. In Fig. 4 we notice, e.g., that if the node weight in state [110] were less than $d_1 = 3$, then we should not extend this node.

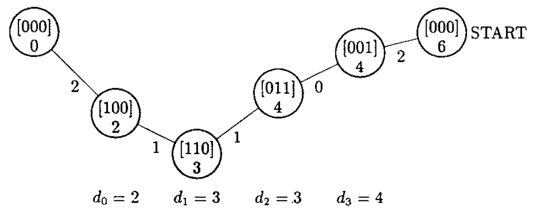

Fig. 4. Weight d_∞ path traversed backward.

The use of the distance profile as a lower bound works for every path from the end of to the root node. Moving backward from state S we can reach the states S_{-0} and S_{-1}, where

$$S = \Big[?,\cdots,?,?,1,\ \underbrace{0,\cdots,0,0}_{m-1 \text{ zeros}}\ \Big]$$

$$S_{-0} = \Big[?,\cdots,?,1,\underbrace{0,0,\cdots,0,0}_{m \text{ zeros}}\Big]$$

$$S_{-1} = \Big[?,\cdots,?,1,\ \underbrace{0,0,\cdots,0,}_{m-1 \text{ zeros}}1\Big].$$

The minimum weights of backward paths stemming from the states S_{-0} and S_{-1} are lower bounded by d_{M-m-1} and d_{M-1}, respectively. Instead of moving backward in the trellis we can, of course, *reverse* the code generators and move forward in the corresponding tree and use the distance profile (of the nonreversed code) to limit effectively the part of the tree that must be explored.

IV. The FAST Algorithm

We shall now describe a fast algorithm for searching a code tree to determine the distance spectrum for a convolutional code with generator $\boldsymbol{G} = [G^{(1)}, G^{(2)}]$ and distance profile $\boldsymbol{d}$. Let

$$\tilde{G}^{(k)} = \left[g_M^{(k)}, g_{M-1}^{(k)}, \cdots, g_0^{(k)}\right],$$

$k = 1,2$, denote the generators for the reversed code. The distance profile of the reversed code is denoted by $\vec{\boldsymbol{d}} = [\tilde{d}_0, \tilde{d}_1, \cdots, \tilde{d}_M]$. To calculate the ith spectral component we start at state $S = [1,0,\cdots,0]$ with weight $W = d_\infty + i - \tilde{d}_0$ in the code tree generated by the reversed encoder $\vec{\boldsymbol{G}} = [\tilde{G}^{(1)}, \tilde{G}^{(2)}]$. We then reduce this weight by the weights of the branches that we traverse when the code tree is searched for nodes with both node weight and state equal to zero. For the state of each explored node, we use the column distances d_{M-m-1} or d_{M-1} to lower-bound the weight of any path leading to a zero state. If the weight is less than this bound, we will always reach a weight that is zero or negative at a nonzero state. Hence it is only necessary to extend a node if the node weight is larger than or equal to this bound!

If both successor nodes are achievable, we follow the zero branch and save (push) the one-branch node (state S_1 and weight W_1) on a stack. Thus we can avoid calculating the node weight for the same node twice, and the algorithm will be twice as fast. (The basic algorithm should of course also be implemented with a stack.) The FAST is shown below. Notice that w_i is calculated using the reversed encoder $\vec{\boldsymbol{G}} = [\tilde{G}^{(1)}, \tilde{G}^{(2)}]$.

FAST: Given k code generators, $G^{(i)}$, $i = 1,\cdots,k$, and $d = d_\infty + j - 1$, this algorithm finds its j first spectral components.

```
F1  (Initialize.) Set $m \leftarrow 1$, $n(\cdot) \leftarrow 0$, $W \leftarrow d - \tilde{d}_0$, and $S \leftarrow [1,0,\cdots,0]$.
F2  (Next nodes.) Calculate $S_0$, $S_1$, $W_0$, and $W_1$. If $m < M$, go to F6.
F3  (Return to zero.) If $W_0 \geq 0$, set $n(d - W_0) \leftarrow n(d - W_0) + 1$.
F4  (Forward on one-branch?) If $W_1 < d_{M-1}$ or $W < d_M$, go to F5. Otherwise, select node$_1$ and set $m \leftarrow 1$. Go to F2.
F5  (From stack?) If stack is empty, the algorithm terminates with the result in $n(\cdot)$. Otherwise, select the node from the stack and set $m \leftarrow 1$. Go to F2.
F6  (Forward on zero-branch?) If $W_0 < d_{M-m-1}$, go to F4.
F7  (Save node$_1$?) If $W_1 \geq d_{M-1}$ and $W \geq d_M$, save node$_1$. In any case, select node$_0$ and set $m \leftarrow m + 1$. Go to F2.
```

If $\boldsymbol{d} \geq \vec{\boldsymbol{d}}$, then the reversed code generators, else the code generators, are used as input to FAST. In Fig. 5 we show the code tree explored by FAST to verify that $n(d_\infty) = 1$ for our encoder $\boldsymbol{G} = [74, 54]$. Only five nodes are visited!

Since we are interested in the spectral components for encoders with optimum (or good) distance profiles, it is interesting to notice that the better the distance profile is, the faster FAST runs! In Fig. 6 the efficiencies of FAST and the basic algorithm are compared when used for

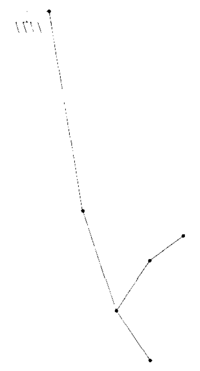
Fig. 5. Code tree explored by FAST.

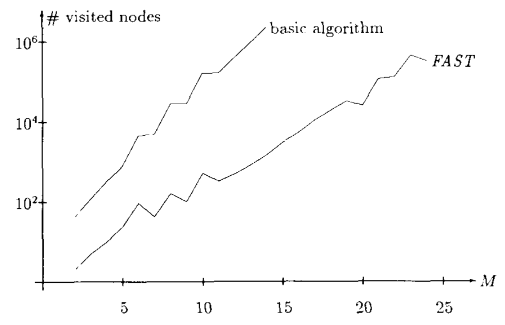

Fig. 6. Efficiencies of FAST and basic algorithm.

testing the nonsystematic ODP encoders given in Tables I and II (see Section VII).

The algorithm was programmed in VMS Fortran-77 and run on a μ-VAX-II computer. As an example we tested an $R = 1/2$, $M = 25$ convolutional encoder $G = [665041116, 516260772]$ with an optimum distance profile. It took only 37 s CPU time to calculate the number of paths (24) with weight $d_\infty = 27$! This code has a free distance larger than that of any previously known code of the same rate and memory. The first few spectral components are

i	0	1	2	3	4	5	6	7	8	9
$d_\infty + i$	27	28	29	30	31	32	33	34	35	36
$n(d_\infty + i)$	24	54	125	278	637	1599	3779	9073	21 831	52 929

An efficient bidirectional search algorithm for computing the free distance has been suggested by Bahl *et al.* [7] and modified by Larsen [8]. To test the efficiency of FAST, we programmed Larsen's modified algorithm and computed d_∞ for the memory $M = 25$ convolutional encoder mentioned before. It took almost 22 h of CPU time to verify that $d_\infty = 27$. Notice that, in addition, FAST calculated the number of weight d_∞ paths in less than 40 s of CPU time! The verification of $d_\infty = 18$ for a memory $M = 15$ encoder took 8 s (bidirectional) and 0.2 s (FAST), respectively. It seems fair to call FAST fast!

V. Modifications of FAST

In this section we shall describe several features that can easily be incorporated in FAST.

Free Distance Unknown

To compute the number of weight d_∞ paths, when d_∞ is unknown, we can use a sufficiently large distance d as input. If $d > d_\infty$ we will reach a zero state with a positive node weight W_{zero}, i.e., d_∞ is at most $d - W_{\text{zero}}$. Hence we can reduce the node weights in the stack by W_{zero} (adjust stack) and reset the path counter to 1. These modifications are shown below and will speed up the calculation of d_∞.

FAST—Free Distance Unknown: Use the previous algorithm with $d = \hat{d}_\infty$, where $\hat{d}_\infty \geq d_\infty$, and make the following modifications:

F3 (Return to zero.) If $W_0 = 0$, set $n(d) \leftarrow n(d) + 1$; else if $W > 0$, set $d \leftarrow d - W_0$, $W_1 \leftarrow W_1 - W_0$, $W \leftarrow W - W_0$, $n(d) \leftarrow 1$ and adjust stack.

Bit Error Probability

When Viterbi decoding is used for a specific code on a binary input channel, the distance spectrum can be used to upper-bound the first event error probability. This bound can be modified to provide a bound on the bit error probability, i.e., the expected number of erroneously decoded information bits per decoded information bit [1]. To calculate this latter bound, we need the total number of nonzero information bits, $n_b(d_\infty + i)$, on all weight $d_\infty + i$ paths, $i = 0, 1, 2, \cdots$. FAST can easily be modified to provide the sequence $n_b(d_\infty + i)$, $i = 0, 1, 2, \cdots$.

FAST—Bit Error Probability: Use the original algorithm and make the following modifications.

F1 (Initialize.) Set $m \leftarrow 1$, $b \leftarrow 1$, $n_b(\cdot) \leftarrow 0$, $W \leftarrow d - \tilde{d}_0$, and $S \leftarrow [1, 0, \cdots, 0]$.

F3 (Return to zero.) If $W_0 \geq 0$, set $n_b(d - W_0) \leftarrow n_b(d - W_0) + 1$.

F4 (Forward on one-branch?) If $W_1 < d_{M-1}$ or $W < d_M$, go to F5. Otherwise, select node$_1$ and set $b \leftarrow b + 1$ and $m \leftarrow 1$. Go to F2.

Note: The variable b is included in the node. Saving a node on the stack means that $b + 1$ will be stored.

Length of Error Event

By incorporating an additional variable L in FAST, we can determine the *length* of each path with weight $d_\infty + i$,

$i = 0, 1, 2, \cdots$. Each time we move forward in the algorithm, L is incremented. When saving a node on the stack, L must also be stored.

FAST—Length of Error Event: Use the original algorithm, and make the following modifications.

F1 (Initialize.) Set $m \leftarrow 1$, $L \leftarrow 0$, $n(\cdot,\cdot) \leftarrow 0$, $W \leftarrow d - \tilde{d}_0$, and $S \leftarrow [1,0,\cdots,0]$.

F2 (Next nodes.) Calculate S_0, S_1, W_0, W_1, and $L \leftarrow L+1$. If $m < M$, go to F6.

F3 (Return to zero.) If $W_0 \geq 0$, set $n(d - W_0, L) \leftarrow n(d - W_0, L) + 1$.

F7 (Save node$_1$?) If $W_1 \geq d_{M-1}$ and $W \geq d_M$ and $W_1 > W_S + 1$, save node$_1$ (saves $W_S + 1$). In any case, select node$_0$ and set $m \leftarrow m+1$. Go to F2.

Note: The variable L is included in the node.

As an example our $R = 1/2$, $M = 25$, ODP convolutional encoder with $d_\infty = 27$ has the following distribution, $n(L)$, of the lengths L of its 24 weight d_∞ paths:

L	25	31	32	33	34	35	36	37	38	40	41	42	43	44	45	47	50	51
$n(L)$	1	1	2	1	2	1	1	2	1	2	1	2	1	2	1	1	1	1

Note: The variable W_S is included in the node.

We tested a long $R = 1/2$, $M = 68$ systematic ODP convolutional code with generator $G^2 = 67114545755646670367015$. It took about 27 h CPU time to calculate the first spectral component. This code has only one path with weight $d_\infty = 31$ (28 paths with weight $d_M = 22$).

Systematic Codes

If we calculate the distance spectrum for systematic encoders, then it is possible to make an additional improvement of FAST. Since $\tilde{G}^{(1)}(D) = D^M$, we have at depth u an accumulated weight

$$W_S = \sum_{j=1}^{M-1} i_{u-j}$$

in the encoder. These information symbols will eventually appear on the path leading to the all-zero state. Hence, at a certain node the node weight must exceed the corresponding W_S; otherwise, every path leading to the all-zero state will result in a negative node weight at this latter state. This modification is shown below, and its effect is shown by calculating the spectrum for the systematic $R = 1/2$, $M = 31$ convolutional encoder $G^{(2)} = [67114543066]$. The first few spectral components are

i	0	1	2	3	4	5	6	7	8	9
$d_\infty + i$	18	19	20	21	22	23	24	25	26	27
$n(d_\infty + i)$	11	0	53	0	307	0	1742	0	10 218	0

When we introduced W_S, the total number of visited nodes was reduced from 2 743 313 394 to 492 096 324.

FAST—Systematic Codes: Use the original algorithm, and make the following modifications.

F1 (Initialize.) Set $m \leftarrow 1$, $n(\cdot) \leftarrow 0$, $W \leftarrow d - \tilde{d}_0$, $W_S \leftarrow 1$, and $S \leftarrow [1,0,\cdots,0]$.

F2 (Next nodes.) Calculate S_0, S_1, W_0, W_1, and $W_S \leftarrow W_S - i^{(1)}$. If $m < M$, go to F6.

F4 (Forward on one-branch?) If $W_1 < d_{M-1}$ or $W < d_M$ or $W_1 \leq W_S + 1$, go to F5. Otherwise, select node$_1$ and set $m \leftarrow 1$ and $W_S \leftarrow W_S + 1$. Go to F2.

VI. Searching for Good Encoders

An exhaustive search for encoders with large d_∞ is practically impossible even for relatively short memories. Therefore, we need efficient rejecting rules that limit the computation of d_∞ to a small fraction of the complete ensemble of encoders.

The jth-order *row distance* r_j is the minimum Hamming weight of a path of length $M + j + 1$ branches which diverges at some point from the zero state and terminates on the zero state [12]. The row distance r_j is nonincreasing with j. Moreover, we have the inequalities

$$d_0 \leq d_1 \leq \cdots \leq d_j \leq \cdots \leq d_\infty$$
$$\leq r_\infty \leq \cdots \leq r_j \leq \cdots \leq r_1 \leq r_0.$$

For a noncatastrophic encoder, $d_\infty = r_\infty$. From the inequalities above it is clear that the row distances can be used as rejecting rules. Their efficiency is shown by the following example.

The total number of rate $R = 1/2$, memory $M = 16$ ($g_M^{(1)} = 1$ or $g_M^{(2)} = 1$) encoders with $g_0^{(1)} = g_0^{(2)} = 1$ is $3 \cdot 2^{2M-2} = 3\ 221\ 225\ 472$. A simple way to generate all the encoders $\boldsymbol{G} = (G^{(1)}, G^{(2)})$ and eliminate the encoders $\boldsymbol{G}' = (G^{(2)}, G^{(1)})$ is, for each $G^{(1)}$, to test only those $G^{(2)}$ for which $\tilde{G}^{(2)} < \tilde{G}^{(1)}$ (in obvious binary notion). The number of encoders is reduced to $3 \cdot 2^{2M-3} - 2^{M-2}$, and thus we have 1 610 596 352 encoders left to test. Hoping to find an encoder with $d_\infty = 20$ we reject successively all encoders with $r_j < 20$, $j = 0, 1, \cdots, 15$, where 15 is arbitrarily chosen. After having used the row distance r_{15}, as a rejecting rule only 1034 candidates are left:

j	Number of Codes with $r_j \geq 20$
0	543 537 361
1	267 253 166
2	84 145 636
3	19 788 663
4	4 764 506
5	1 138 502
6	309 889
7	96 872
8	35 853
9	14 974
10	7167
11	3954
12	2488
13	1650
14	1233
15	1034

Another 123 of these can be rejected since they suffer from *catastrophic error-propagation*, i.e., there exists certain infinite weight input sequences that produce finite weight code sequences [13]. These encoders must be avoided. For the remaining 911 encoders we calculated d_∞ and found 200 encoders with $d_\infty = 20$. The best one is given in Table III and IV.

One might suspect that a memory $M = 17$, $R = 1/2$ encoder exists with $d_\infty = 21$. We tested this hypothesis and found that all candidates have row distance $r_{10} < 21$. The efficiency of using the row distances as rejecting rules in this case is shown as follows:

j	Number of Codes with $r_j \geq 21$
0	2 204 679 293
1	791 375 586
2	160 725 370
3	16 854 476
4	1 471 120
5	101 684
6	5098
7	236
8	16
9	2
10	0

By this method we also verified that the memory $M = 19$, $R = 1/2$ encoder, which is given in Tables I and II has optimum d_∞ (optimum free distance or OFD)!

VII. RESULTS

In this section we report the results of using FAST to find good binary rate $R = 1/2$ convolutional codes. A code is said to be an *optimum distance profile* code when its distance profile is superior to or equal to that of any code with the same rate and memory. In Tables I and II we give extensive lists of nonsystematic rate $R = 1/2$ ODP encoders. Some of the reported encoders have a d_∞ superior to that of any previously known code with the same rate and memory. Some of the encoders have been reported before [3], [14] but without specifying the distance spec-

TABLE I
$n(d_\infty + i)$, $i = 0, \cdots, 9$ FOR NONSYSTEMATIC ODP ENCODERS

				Value of i									
M	$G^{(1)}$	$G^{(2)}$	d_∞	0	1	2	3	4	5	6	7	8	9
2	7	5	5	1	2	4	8	16	32	64	128	256	512
3	74	54	6	1	3	5	11	25	55	121	267	589	1299
4	62	56	7	2	3	4	16	37	68	176	432	925	2156
5	75	55	8	2	7	10	18	49	124	292	678	1576	3694
6	634	564	10	12	0	53	0	234	0	1517	0	8862	0
7	626	572	10	1	6	13	20	64	123	321	764	1858	4442
8	751	557	12	10	9	30	51	156	340	875	1951	5127	11589
9	7664	5714	12	1	8	8	31	73	150	441	940	2214	5531
10	7512	5562	14	19	0	80	0	450	0	2615	0	15276	0
11	6643	5175	14	1	10	25	46	105	258	616	1531	3611	8675
12	63374	47244	15	2	10	29	55	138	301	692	1720	4199	10245
13	45332	77136	16	5	15	21	56	161	381	879	2095	5085	12207
14	65231	43677	17	3	16	44	62	172	455	1025	2395	5853	14487
15	517604	664134	18	10	0	86	0	417	0	2461	0	14251	0
16	717066	522702	19	9	16	48	112	259	596	1457	3460	8257	20562
17	506477	673711	20	12	37	47	140	358	855	2013	4827	11694	28213
18	5653664	7746714	21	13	38	63	142	363	934	2205	5146	12657	30579
19	5122642	7315626	22	26	0	160	0	916	0	5154	0	29386	0
20	6567413	5322305	22	2	28	51	86	222	554	1321	3316	8007	19074
21	67520654	50371444	24	40	0	251	0	1379	0	7812	0	45858	0
22	67132702	50516146	24	25	0	163	0	844	0	5183	0	29380	0
23	55076157	75501351	26	65	0	331	0	2014	0	11359	0	65585	0
24	673275374	506302644	26	26	0	182	0	940	0	5604	0	32677	0
25	665041116	516260772	27	24	54	125	278	637	1599	3779	9073	21831	52929

TABLE II
$n_b(d_\infty + i)$, $i = 0,\cdots,9$ FOR NONSYSTEMATIC ODP ENCODERS

				Value of i									
M	$G^{(1)}$	$G^{(2)}$	d_∞	0	1	2	3	4	5	6	7	8	9
2	7	5	5	1	4	12	32	80	192	448	1024	2304	5120
3	74	54	6	2	7	18	49	130	333	836	2069	5060	12255
4	62	56	7	4	12	20	72	225	500	1324	3680	8967	22270
5	75	55	8	6	23	44	104	302	832	2180	5596	14254	36240
6	634	564	10	46	0	332	0	1911	0	14149	0	97518	0
7	626	572	10	6	22	56	136	464	981	2914	7362	19800	50322
8	751	557	12	40	33	196	357	1236	2884	8522	19967	57254	138811
9	7664	5714	12	2	36	34	187	490	1212	3864	9064	23494	62311
10	7512	5562	14	82	0	530	0	3678	0	25955	0	177972	0
11	6643	5175	14	4	46	156	366	930	2514	6228	17379	44124	112437
12	63374	47244	15	6	46	177	386	1070	2668	6780	18136	47755	125068
13	45332	77136	16	26	91	128	416	1350	3503	9154	23227	60938	156079
14	65231	43677	17	17	78	276	484	1464	4152	10513	26910	69939	185214
15	517604	664134	18	49	0	615	0	3710	0	26416	0	176688	0
16	717066	522702	19	55	80	344	914	2317	5936	16043	40292	103109	275218
17	506477	673711	20	90	275	404	1350	3596	9671	24386	62711	161662	414905
18	5653664	7746714	21	73	272	459	1360	3701	10452	26975	66418	174421	446654
19	5122642	7315626	22	130	0	1392	0	9539	0	62822	0	407587	0
20	6567413	5322305	22	14	202	398	810	2210	6030	15520	42046	107978	273086
21	67520654	50371444	24	260	0	2351	0	15425	0	100971	0	669522	0
22	67132702	50516146	24	190	0	1585	0	9741	0	67999	0	438897	0
23	55076157	75501351	26	498	0	3339	0	23971	0	154099	0	1001291	0
24	673275374	506302644	26	217	0	1668	0	10679	0	73589	0	482524	0
25	665041116	516260772	27	214	526	1301	3212	7861	21034	52895	134474	341063	876004

trum. Furthermore, distance spectra for the short (memory $M \leq 4$) encoders have been reported by Odenwalder [6] and by Conan [15]. They are included in the tables for the sake of completeness.

In Tables III and IV we list a few encoders that have a d_∞ superior to that of an ODP encoder of the same memory given in Tables I and II. The encoders with memories $M = 11$ and $M = 12$ have fewer weight d_∞ paths than the corresponding encoders in [15]!

Twenty years ago Bahl and Jelinek [16] developed a class of complementary rate $R = 1/2$ encoders with surprisingly large d_∞. They are defined by

$$G^{(1)} = [1, g_1, g_2, \cdots, g_{M-1}, 1]$$

and

$$G^{(2)} = [1, g_1', g_2', \cdots, g_{M-1}', 1]$$

where g_j' denotes the complement of g_j. We have extended their table to memory $M = 31$ and list the encoders in Tables V and VI together with the first ten spectral components. It is worth noticing that the memory $M = 15$

TABLE III
$n(d_\infty + i)$, $i = 0,\cdots,9$ FOR NONSYSTEMATIC OFD ENCODERS

				Value of i									
M	$G^{(1)}$	$G^{(2)}$	d_∞	0	1	2	3	4	5	6	7	8	9
11	7173	5261	15	14	21	34	101	249	597	1373	3317	8014	19559
12	53734	72304	16	14	38	35	108	342	724	1604	4020	9825	23899
14	63057	44735	18	26	0	165	0	845	0	4844	0	28513	0
15	533514	653444	19	30	67	54	167	632	1402	2812	7041	18178	43631
16	626656	463642	20	43	0	265	0	1341	0	7613	0	44817	0
18	4551474	6354344	22	65	0	349	0	1903	0	10947	0	63130	0

TABLE IV
$n_b(d_\infty + i)$, $i = 0,\cdots,9$ FOR NONSYSTEMATIC OFD ENCODERS

				Value of i									
	$G^{(1)}$	$G^{(2)}$	d_∞	0	1	2	3	4	5	6	7	8	9
11	7173	5261	15	66	98	220	788	2083	5424	13771	35966	93970	246720
12	53734	72304	16	60	188	288	952	2754	6628	16606	44640	116712	304987
14	63057	44735	18	133	0	1321	0	7901	0	54864	0	370057	0
15	533514	653444	19	174	420	534	1712	5838	14210	32898	87786	237228	609868
16	626656	463642	20	255	0	2382	0	14089	0	92985	0	624583	0
18	4551474	6354344	22	418	0	3219	0	20753	0	138503	0	904981	0

TABLE V
$n(d_\infty + i)$, $i = 0,\cdots,9$ FOR BAHL-JELINEK ENCODERS

			Value of i									
M	$G^{(1)}$	d_∞	0	1	2	3	4	5	6	7	8	9
2	7	5	1	2	4	8	16	32	64	128	256	512
3	64	6	2	0	10	0	49	0	241	0	1185	0
4	72	7	2	3	4	16	37	68	176	432	925	2156
5	73	8	2	0	20	0	68	0	469	0	2560	0
6	754	9	2	3	10	28	43	101	287	655	1554	3804
7	712	10	3	0	18	0	106	0	572	0	3347	0
8	745	11	2	5	16	31	75	175	366	940	2311	5567
9	6654	12	4	0	33	0	139	0	902	0	5051	0
10	7026	13	2	10	20	40	92	216	539	1270	3131	7597
11	7363	14	5	0	44	0	221	0	1334	0	7629	0
12	73214	15	3	13	27	74	130	345	851	2039	4816	11816
13	67432	16	7	0	60	0	323	0	1797	0	10539	0
14	66475	17	5	22	32	81	164	489	986	2628	6035	15072
15	673514	18	12	0	85	0	483	0	2635	0	15740	0
16	644246	18	2	4	30	36	153	237	742	1490	4142	9072
17	735505	20	23	0	114	0	700	0	4023	0	23311	0
18	6756224	20	3	9	52	41	237	299	1207	2038	6341	12342
19	6763426	20	1	0	33	0	191	0	1030	0	5769	0
20	6243735	22	9	5	82	61	319	384	1648	2571	9271	16000
21	66452114	22	3	0	49	0	262	0	1320	0	8096	0
22	66237046	24	23	5	113	60	526	562	2778	3505	14621	22323
23	66557413	24	7	0	73	0	388	0	1962	0	11506	0
24	661563014	26	49	7	188	70	778	668	4236	4444	22374	29329
25	620113072	26	15	0	114	0	561	0	2941	0	17396	0
26	670735375	26	3	0	73	8	292	88	1239	844	6402	5877
27	6243245414	28	38	0	159	0	829	0	4425	0	25435	0
28	7133254172	28	13	0	95	11	414	95	2065	953	10393	7811
29	6166044143	28	2	0	63	0	246	0	1140	0	6444	0
30	67526462014	30	24	0	176	11	637	116	3099	1195	15540	9413
31	62162354046	30	9	0	106	0	423	0	2130	0	10583	0

TABLE VI
$n_b(d_\infty + i)$, $i = 0,\cdots,9$ FOR BAHL-JELINEK ENCODERS

			Value of i									
M	$G^{(1)}$	d_∞	0	1	2	3	4	5	6	7	8	9
2	7	5	1	4	12	32	80	192	448	1024	2304	5120
3	64	6	4	0	38	0	277	0	1806	0	11063	0
4	72	7	4	12	20	72	225	500	1324	3680	8967	22270
5	73	8	5	0	98	0	446	0	3872	0	25644	0
6	754	9	4	10	50	166	303	774	2371	6042	15826	41796
7	712	10	10	0	82	0	736	0	4841	0	34216	0
8	745	11	4	22	100	192	555	1488	3346	9444	25307	64860
9	6654	12	10	0	199	0	1040	0	8384	0	55137	0
10	7026	13	8	50	104	292	766	1980	5195	13084	35343	92740
11	7363	14	17	0	292	0	1829	0	13435	0	90432	0
12	73214	15	11	66	173	564	1102	3412	8881	22870	57952	153336
13	67432	16	28	0	433	0	2962	0	19758	0	132399	0
14	66475	17	25	146	258	672	1484	5154	10898	31620	76925	206402
15	673514	18	59	0	640	0	4694	0	29937	0	204931	0
16	644246	18	8	16	206	302	1410	2249	8316	17306	52576	122012
17	735505	20	143	0	1005	0	7102	0	48029	0	320493	0
18	6756224	20	20	45	430	399	2422	3133	14224	24768	86172	173424
19	6763426	20	8	0	237	0	1678	0	11297	0	73363	0
20	6243735	22	56	27	746	595	3364	4248	20830	32619	130050	232944
21	64452114	22	24	0	395	0	2688	0	15372	0	108280	0
22	66237046	24	172	29	1038	620	6062	6400	36374	46075	214750	336429
23	66557413	24	42	0	631	0	4233	0	24010	0	159902	0
24	661563014	26	414	45	1932	788	9260	8190	58036	60804	342974	456373
25	620113072	26	122	0	1088	0	6518	0	38745	0	257952	0
26	670735375	26	22	0	656	66	3150	1008	15562	10884	91682	85017
27	6243245414	28	330	0	1636	0	10164	0	61360	0	394667	0
28	7133254172	28	92	0	894	85	4854	1111	27514	12869	157388	118051
29	6166044143	28	14	0	572	0	2679	0	14602	0	94100	0
30	67526462014	30	212	0	1824	109	7734	1504	42388	17263	241152	148269
31	62162354046	30	76	0	1134	0	5106	0	29353	0	163336	0

encoder in [16] is catastrophic. This was observed by Vinck [17].

Massey and Costello [2] introduced a class of nonsystematic convolutional codes called quick-look-in (QLI) codes in which the two generators differ only in the second position. The main feature of QLI encoders is that they have a feedforward inverse, which can be implemented by a simple modulo 2 adder. This makes it easy to extract the information digits from the hard-decisioned received sequences. Furthermore, since the feedforward inverse has "weight" two, the error amplification factor $A = 2$ is the smallest possible for nonsystematic codes. In Tables VII and VIII we list our QLI encoders.

In [18] and [19] Helgert introduced a slight generalization of the QLI encoders by letting only the Lth position of the two generators differ. In Tables IX and X we present our extension of Helgert's results. In Tables XI and XII we list ODP QLI encoders.

Using the modification of FAST described in Section V, we compiled an extensive list of systematic ODP encoders (Tables XIII and XIV). For the sake of completeness we also give the minimum distances d_M for an extensive list,

TABLE VII
$n(d_\infty + i)$, $i = 0, \cdots, 9$ FOR QLI ENCODERS (NON ODP)

			Value of i									
M	$G^{(1)}$	d_∞	0	1	2	3	4	5	6	7	8	9
2	5	5	1	2	4	8	16	32	64	128	256	512
3	54	6	1	3	5	11	25	55	121	267	589	1299
4	46	7	2	4	6	15	37	83	191	442	1015	2334
5	55	8	2	7	10	18	49	124	292	678	1576	3694
6	454	9	4	8	11	25	70	181	405	945	2279	5414
7	542	9	1	4	13	25	51	115	270	686	1663	3955
8	551	10	1	9	18	30	73	172	379	992	2495	5735
9	5664	11	3	6	19	37	83	207	450	1146	2719	6631
10	5506	12	3	11	23	47	99	234	587	1474	3535	8363
11	5503	13	8	16	26	67	146	361	870	2128	5205	12510
12	56414	14	10	21	47	90	210	520	1311	3096	7458	17856
13	46716	14	3	12	32	71	141	335	877	1991	4852	11775
14	51503	15	6	14	36	68	176	469	1006	2390	5924	14285
15	510474	16	11	29	50	122	269	688	1637	3955	9574	22960
16	522416	16	2	21	35	51	155	376	898	2164	5337	12891
17	454643	17	5	23	38	90	230	499	1227	2994	7233	17526
18	5522214	18	6	26	62	139	326	727	1614	4070	10189	24338
19	4517006	18	2	16	42	78	173	445	1120	2610	6158	14933
20	5036543	19	4	24	50	97	253	586	1441	3525	8526	20482
21	47653514	20	7	26	69	138	349	867	1965	4803	11405	27759
22	51102726	20	1	17	42	93	215	476	1096	2733	6640	16127
23	53171663	21	3	31	70	136	327	754	1866	4531	10676	26209
24	510676714	22	7	38	77	171	423	948	2231	5469	13466	32186

TABLE VIII
$n_b(d_\infty + i)$, $i = 0, \cdots, 9$ FOR QLI ENCODERS (NON-ODP)

			Value of i									
M	$G^{(1)}$	d_∞	0	1	2	3	4	5	6	7	8	9
2	5	5	1	4	12	32	80	192	448	1024	2304	5120
3	54	6	2	7	18	49	130	333	836	2069	5060	12255
4	46	7	4	12	26	74	205	530	1369	3504	8849	22180
5	55	8	6	23	44	104	302	832	2180	5596	14254	36240
6	454	9	14	34	55	148	472	1310	3249	8338	21707	55610
7	542	9	1	16	63	132	307	800	2088	5720	15065	38732
8	551	10	2	35	96	184	476	1280	3120	8700	23636	58923
9	5664	11	11	26	113	236	619	1646	3848	10726	27371	71538
10	5506	12	18	55	140	345	772	1940	5406	14778	37604	94779
11	5503	13	44	100	172	492	1168	3206	8368	21972	57605	146902
12	56414	14	64	125	342	688	1784	4930	13330	33422	86052	218198
13	46716	14	20	62	214	553	1156	3139	8720	20991	54638	141197
14	51503	15	26	98	284	528	1550	4514	10428	26168	69266	178100
15	510474	16	74	215	378	1080	2622	7140	17886	46291	119356	302454
16	522416	16	10	149	294	455	1488	3772	9676	25092	65646	168003
17	454643	17	29	168	300	804	2176	5098	13807	35484	90999	232870
18	5522214	18	44	210	548	1369	3448	8077	19216	51812	135956	342772
19	4517006	18	12	122	338	752	1704	4497	12418	30712	77344	197479
20	5036543	19	28	210	452	958	2629	6686	17295	45238	115760	291672
21	47653514	20	54	220	642	1398	3906	10319	24774	63613	159064	406201
22	51102726	20	6	155	432	939	2384	5624	13846	36425	92838	237537
23	53171663	21	21	308	696	1508	3761	9306	24806	62782	155366	400384
24	510676714	22	72	368	858	1941	4958	12198	30032	77495	200126	501950

TABLE IX
$n(d_\infty + i)$, $i = 0,\cdots,9$ FOR HELGERT ENCODERS $G = (G^{(1)}, G^{(1)} + D^L)$

				Value of i									
M	L	$G^{(1)}$	d_∞	0	1	2	3	4	5	6	7	8	9
7	2	656	9	1	4	11	19	39	101	252	597	1409	3411
8	2	623	10	1	5	10	23	51	146	326	676	1754	4275
9	4	6134	11	2	7	17	38	80	179	450	1075	2595	6351
10	2	6626	12	2	14	29	38	105	260	580	1484	3693	8867
11	2	6713	13	6	16	33	71	147	340	875	2184	5249	12606
13	2	66206	14	2	12	24	55	120	256	686	1668	3903	9409
14	6	71073	15	3	16	41	67	184	392	929	2424	5609	13693
15	6	602364	16	6	29	48	115	257	589	1532	3567	8663	20852
16	7	651426	16	1	15	29	73	165	364	951	2174	5384	12745
17	4	601067	17	3	15	42	100	192	500	1151	2905	6536	16331
18	7	6504664	18	5	20	63	113	296	674	1594	3837	9263	22629
19	3	6047646	19	11	32	61	167	358	818	2124	4999	12120	29128
20	4	6144363	19	2	20	42	90	247	543	1261	3063	7512	17804
21	8	60405634	20	4	27	59	134	332	769	1779	4331	10520	25166
22	6	61310166	21	12	39	71	200	427	1036	2493	5980	14560	35001
23	9	70311243	21	2	19	59	101	248	602	1420	3427	8367	20297
24	4	603757254	22	3	49	73	151	378	881	2169	5096	12507	30573

TABLE X
$n_b(d_\infty + i)$, $i = 0,\cdots,9$ FOR HELGERT ENCODERS $G = (G^{(1)}, G^{(1)} + D^L)$

				Value of i									
M	L	$G^{(1)}$	d_∞	0	1	2	3	4	5	6	7	8	9
7	2	656	9	3	18	55	108	257	736	2006	5168	13227	34518
8	2	623	10	4	21	54	145	328	1144	2744	6156	17312	44905
9	4	6134	11	8	38	101	258	580	1446	4002	10290	26763	69850
10	2	6626	12	10	76	188	268	868	2282	5494	15030	39846	102615
11	2	6713	13	36	84	235	560	1219	3172	8549	23086	59333	150564
13	2	66206	14	6	70	168	401	1046	2398	6914	17682	43948	112871
14	6	71073	15	29	100	295	584	1696	3976	10023	27610	68347	175998
15	6	602364	16	32	199	356	991	2462	5973	16478	40735	105526	268914
16	7	651426	16	8	115	210	675	1624	3786	10714	25630	68078	170013
17	4	601067	17	17	112	364	932	1858	5370	12849	34890	82240	219482
18	7	6504664	18	46	160	556	1067	3020	7402	18728	47619	121946	314189
19	3	6047646	19	83	238	575	1734	3842	9512	26218	65344	166850	421064
20	4	6144363	19	12	170	384	894	2551	6264	15469	39584	102048	254166
21	8	60405634	20	40	227	556	1278	3506	8935	21780	55887	143438	361328
22	6	61310166	21	96	350	713	2180	4963	12812	32535	82774	212392	533072
23	9	70311243	21	10	166	601	1106	2840	7350	18460	46606	119523	305544
24	4	603757254	22	20	473	780	1753	4550	11439	29532	72466	187234	478577

$2 \le M \le 96$, of systematic ODP encoders (Table XV). These results are an extension of the tables reported in [20], [3], [21].

Fifteen years ago Massey [22] conjectured, in contrast to the presumed superiority of nonsystematic codes to systematic codes, that a sequential decoder will perform about as well with a systematic $R = 1/2$ code of memory $2M$ as with a nonsystematic code of memory M. Since the longer code is systematic, every other channel symbol in the tail, which is used to terminate an encoded information sequence, is a zero which is known beforehand and can be omitted before transmission. Hence the two codes require the same alloted space for transmission of their corresponding tails, which is the practical consequence of Massey's conjecture. To test the conjecture, we have compared d_∞ for nonsystematic ODP codes of memory M with d_∞ for systematic ODP codes of memory $2M$. The result is in Fig. 7, which gives striking support to Massey's conjecture.

The import of Massey's conjecture, to which our comparison lends credence, is that a systematic $R = 1/2$ convolutional code can be used instead of a nonsystematic one without any sacrifice in the effective transmission rate, error probability or computational performance of a sequential decoder, provided that the memory of the systematic code is chosen as twice the allotted tail length in information symbols. Thus the long systematic convolutional encoders in Tables XIII and XIV appear attractive for use with sequential decoders.

In Fig. 8 we show the free distance for several classes of rate $R = 1/2$ convolutional codes. For comparison, we give Heller's upper bound [23]:

$$d_\infty \le \min_{1 \le k} \left\lfloor \frac{2^k}{2^k - 1}(M + k) \right\rfloor, \qquad R = 1/2$$

where $\lfloor \cdot \rfloor$ denotes the floor function. This bound has been improved by one for some memories and by two for

TABLE XI
$n(d_\infty + i)$, $i = 0,\cdots,9$ FOR QLI ENCODERS WITH ODP

			Value of i									
M	$G^{(1)}$	d_∞	0	1	2	3	4	5	6	7	8	9
2	7	5	1	2	4	8	16	32	64	128	256	512
3	74	6	1	3	5	11	25	55	121	267	589	1299
4	76	6	1	1	3	7	18	40	87	209	476	1096
5	75	8	2	7	10	18	49	124	292	678	1576	3694
6	714	8	1	2	5	14	27	68	157	366	914	2161
7	742	9	1	4	13	25	51	115	270	686	1663	3955
8	743	9	1	1	5	12	21	51	127	316	780	1886
9	7434	10	2	1	6	14	31	112	219	492	1205	2846
10	7422	11	2	5	14	26	57	146	345	841	2070	4956
11	7435	12	5	3	10	45	81	183	427	1020	2593	6186
12	74044	11	1	1	5	18	33	62	162	377	930	2352
13	74046	13	2	6	7	19	48	115	278	676	1726	4070
14	74047	14	2	8	12	32	71	184	402	981	2391	5589
15	740464	14	2	1	6	18	61	89	260	633	1466	3560
16	740462	15	3	5	11	33	67	168	404	992	2470	5903
17	740463	16	2	9	15	46	114	231	585	1344	3179	7850
18	7404634	16	1	2	13	24	43	139	283	741	1717	4040
19	7404242	15	1	0	2	9	19	48	143	315	725	1825
20	7404155	18	2	12	15	45	126	226	552	1412	3329	8109
21	74041544	18	2	4	6	36	78	183	439	1026	2419	6049
22	74042436	19	2	9	13	28	96	225	539	1283	3131	7534
23	74041567	19	1	2	8	26	50	105	302	722	1702	4064
24	740415664	20	1	8	11	29	67	170	427	939	2325	5702
25	740424366	20	1	3	6	19	54	117	242	567	1447	3525
26	740424175	22	8	7	38	52	164	311	806	1996	4828	12103
27	7404155634	22	2	6	14	31	93	186	467	1141	2658	6545
28	7404241726	23	2	7	24	38	105	270	685	1589	3936	9611
29	7404154035	24	6	24	32	84	202	473	1195	2653	6687	16203
30	74041567514	23	1	1	6	10	34	88	208	559	1293	3051
31	74041567512	25	5	11	15	54	134	332	841	2072	4878	11683

TABLE XII
$n_b(d_\infty + i)$, $i = 0,\cdots,9$ FOR QLI ENCODERS WITH ODP

			Value of i									
M	$G^{(1)}$	d_∞	0	1	2	3	4	5	6	7	8	9
2	7	5	1	4	12	32	80	192	448	1024	2304	5120
3	74	6	2	7	18	49	130	333	836	2069	5060	12255
4	76	6	2	3	10	27	88	220	568	1497	3794	9606
5	75	8	6	23	44	104	302	832	2180	5596	14254	36240
6	714	8	4	6	22	70	160	450	1148	2972	8058	20657
7	742	9	1	16	63	132	307	800	2088	5720	15065	38732
8	743	9	5	2	21	66	127	352	977	2578	7000	18368
9	7434	10	6	5	32	74	226	838	1796	4466	11636	29568
10	7422	11	6	24	72	158	397	1092	2909	7670	20198	51668
11	7435	12	26	15	64	313	614	1563	3936	10036	27462	69868
12	74044	11	1	4	21	120	233	434	1308	3262	8740	23500
13	74046	13	6	42	47	134	384	974	2576	6896	18550	46676
14	74047	14	6	46	86	258	634	1690	3936	10439	26972	67351
15	740464	14	12	1	48	124	496	777	2492	6467	16250	41734
16	740462	15	13	30	69	248	577	1564	3958	10372	27884	70560
17	740463	16	14	71	112	382	1090	2309	6226	15342	38726	101108
18	7404634	16	8	12	98	196	396	1343	2884	8395	20466	51330
19	7404242	15	1	0	12	58	161	434	1283	3098	7639	20114
20	7404155	18	12	88	144	403	1238	2410	6194	16954	42076	108179
21	74041544	18	18	22	52	292	720	1871	4828	12178	29772	78919
22	74042436	19	12	62	107	252	950	2440	6167	15794	41077	103602
23	74041567	19	11	18	72	232	532	1116	3514	8806	21986	55304
24	740415664	20	6	66	106	285	692	1842	5096	11959	31040	80010
25	740424366	20	6	25	64	175	562	1319	2908	7159	19038	49437
26	740424175	22	58	85	376	552	1838	3879	10374	27482	68686	182757
27	7404155634	22	14	60	136	329	1030	2276	5874	15205	37782	97169
28	7404241726	23	14	86	248	392	1209	3340	8855	21772	57292	146058
29	7404154035	24	64	238	370	1012	2530	6267	16618	38911	102596	259647
30	74041567514	23	15	8	54	108	390	1050	2766	7616	18733	46492
31	74041567512	25	51	112	165	614	1652	4424	11779	30322	74916	187794

TABLE XIII
$n(d_\infty + i)$, $i = 0, \cdots, 9$ FOR SYSTEMATIC ODP ENCODERS

			Value of i									
M	$G^{(2)}$	d_∞	0	1	2	3	4	5	6	7	8	9
1	6	3	1	1	1	1	1	1	1	1	1	1
2	7	4	2	0	5	0	13	0	34	0	89	0
3	64	4	1	0	6	0	16	0	69	0	232	0
4	72	5	2	2	2	12	19	28	88	174	300	718
5	73	6	3	0	13	0	55	0	298	0	1401	0
6	654	6	2	0	9	0	40	0	251	0	1178	0
7	654	6	2	0	9	0	40	0	251	0	1178	0
8	715	7	2	2	5	18	26	66	169	383	980	2160
9	6714	8	5	0	15	0	101	0	571	0	3057	0
10	7152	8	3	0	16	0	79	0	457	0	2618	0
11	7155	9	5	6	7	31	66	166	379	882	2271	5245
12	67114	9	1	4	10	15	46	104	224	576	1368	3322
13	67116	10	5	0	27	0	124	0	777	0	4529	0
14	67115	10	4	0	15	0	121	0	594	0	3550	0
15	671144	10	3	0	19	0	90	0	556	0	3134	0
16	671166	12	13	0	46	0	263	0	1486	0	9019	0
17	714447	11	4	3	0	14	50	101	219	589	1358	3277
18	7144474	12	5	0	29	0	131	0	842	0	4856	0
19	7144616	12	3	0	23	0	92	0	556	0	3472	0
20	6711455	12	2	4	1	19	43	88	223	543	1306	3104
21	67115144	12	2	2	9	14	30	94	181	487	1155	2656
22	67114552	14	12	0	43	0	225	0	1388	0	8057	0
23	71446165	14	10	0	31	0	161	0	1085	0	5834	0
24	671145434	15	7	9	18	56	114	310	698	1661	4097	10025
25	671145452	15	6	10	17	55	112	263	663	1547	3761	9085
26	714476125	16	16	0	92	0	488	0	2843	0	16243	0
27	6711455364	16	10	0	35	0	225	0	1322	0	7732	0
28	7144761242	16	11	0	60	0	273	0	1850	0	10372	0
29	7144760535	18	22	0	118	0	695	0	3926	0	22788	0
30	67114545644	16	3	0	21	0	106	0	554	0	3400	0
31	67114543066	18	11	0	53	0	307	0	1742	0	10218	0

TABLE XIV
$n_b(d_\infty + i)$, $i = 0, \cdots, 9$ FOR SYSTEMATIC ODP ENCODERS

			Value of i									
M	$G^{(2)}$	d_∞	0	1	2	3	4	5	6	7	8	9
1	6	3	1	2	3	4	5	6	7	8	9	10
2	7	4	3	0	15	0	58	0	201	0	655	0
3	64	4	1	0	16	0	62	0	360	0	1502	0
4	72	5	4	4	6	46	79	138	488	1044	2016	5292
5	73	6	6	0	44	0	245	0	1661	0	9508	0
6	654	6	4	0	28	0	158	0	1311	0	7433	0
7	654	6	4	0	28	0	158	0	1311	0	7433	0
8	715	7	4	6	17	68	110	318	917	2256	6276	15124
9	6714	8	13	0	52	0	477	0	3226	0	20650	0
10	7152	8	8	0	55	0	365	0	2568	0	17502	0
11	7155	9	13	22	27	126	338	910	2277	5688	15763	39552
12	67114	9	1	10	40	64	214	538	1258	3530	9132	23826
13	67116	10	13	0	107	0	647	0	4713	0	32181	0
14	67115	10	10	0	59	0	577	0	3553	0	24620	0
15	671144	10	5	0	74	0	427	0	3169	0	21154	0
16	671166	12	44	0	225	0	1532	0	9786	0	69341	0
17	714447	11	12	10	0	56	254	570	1347	3812	9456	24778
18	7144474	12	14	0	131	0	738	0	5383	0	36069	0
19	7144616	12	7	0	104	0	478	0	3462	0	25312	0
20	6711455	12	6	12	4	93	210	488	1356	3511	9124	23462
21	67115144	12	8	4	40	64	152	526	1078	3139	7918	19810
22	67114552	14	47	0	204	0	1303	0	9453	0	62741	0
23	71446165	14	41	0	150	0	919	0	7354	0	44615	0
24	671145434	15	27	36	88	318	656	1960	4890	12504	32553	85136
25	671145452	15	22	42	91	308	644	1678	4555	11446	29925	76436
26	714476125	16	66	0	475	0	3118	0	20855	0	135670	0
27	6711455364	16	44	0	169	0	1387	0	9325	0	62403	0
28	7144761242	16	44	0	313	0	1614	0	12801	0	82777	0
29	7144760535	18	89	0	681	0	4708	0	30544	0	199096	0
30	67114545644	16	9	0	99	0	637	0	3575	0	26056	0
31	67114543066	18	50	0	268	0	2064	0	12945	0	86741	0

TABLE XV
MINIMUM DISTANCES d_M AND $n(d_M)$ FOR SYSTEMATIC ODP ENCODERS

M	d_M	$n(d_M)$	$G^{(2)}$	M	d_M	$n(d_M)$	$G^{(2)}$
1	3	1	6	49	17	1	67114545755646676
2	3	1	6	50	18	38	67114545755646676
3	4	3	64	51	18	16	671145457556466760
4	4	1	64	52	18	7	671145457556466760
5	5	5	65	53	18	2	671145457550027077
6	5	2	650	54	19	43	6711454575571301174
7	6	11	670	55	19	20	6711454575571301176
8	6	5	670	56	19	7	6711454575571301176
9	6	1	6710	57	19	2	67114545755713011760
10	7	12	6710	58	20	60	67114545755713011760
11	7	5	6711	59	20	25	67114545755646670367
12	8	29	67114	60	20	10	671145457556466703670
13	8	12	67114	61	20	2	671145457557130117610
14	8	6	67115	62	20	1	671145457557130117611
15	8	1	671150	63	21	25	6711454575571301176114
16	9	18	671144	64	21	10	6711454575571301176114
17	9	7	671151	65	21	2	6711454575571301176114
18	9	3	6711514	66	22	71	67114545755713011761144
19	10	31	6711454	67	22	29	67114545755646670367016
20	10	13	6711454	68	22	9	67114545755646670367017
21	10	4	67114544	69	22	4	671145457556466703670170
22	10	1	67115142	70	22	1	671145457556466703670170
23	11	27	67114543	71	23	46	671145457557130117611463
24	11	11	671145430	72	23	16	6711454575564667036701444
25	11	5	671151572	73	23	5	6711454575564667036701446
26	11	1	671151505	74	23	2	6711454575564667036701447
27	12	21	6711454574	75	24	56	67114545755713011761146370
28	12	8	6711454306	76	24	20	67114545755713011761146342
29	12	2	6711454311	77	24	8	67114545755713011761146373
30	13	43	67114545754	78	24	3	671145457557130117611463424
31	13	15	67114545754	79	25	74	671145457557130117611463432
32	13	4	67114545755	80	25	33	671145457557130117611463433
33	13	1	671145457554	81	25	16	6711454575571301176114634334
34	14	34	671145457556	82	25	4	6711454575564667036701447272
35	14	14	671145454470	83	25	1	6711454575564667036701447277
36	14	5	6711454544704	84	26	41	67114545755646670367014472730
37	14	2	6711454544676	85	26	20	67114545755713011761146343362
38	15	31	6711454575564	86	26	6	67114545755713011761146343363
39	15	12	67114545755644	87	26	2	671145457557130117611463433634
40	15	3	67114545755712	88	27	62	671145457556466703670144727304
41	15	1	67114545755713	89	27	28	671145457556466703670144727305
42	16	31	671145457556464	90	27	11	6711454575564667036701447273054
43	16	14	671145457556464	91	27	5	6711454575564667036701447273056
44	16	5	671145457556153	92	27	1	6711454575564667036701447273357
45	16	1	6711454575561314	93	28	42	67114545755646670367014472730510
46	17	39	6711454575564666	94	28	20	67114545755646670367014472730512
47	17	13	6711454575565667	95	28	5	67114545755646670367014472730511
48	17	4	67114545755646674	96	28	1	671145457556466703670144727305110

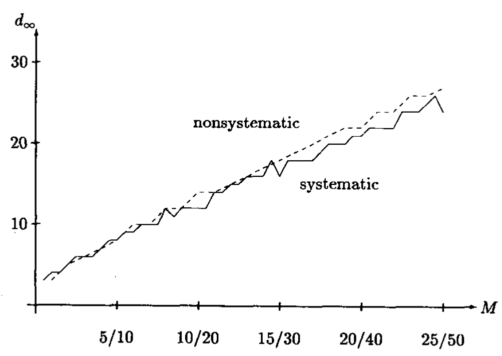

Fig. 7. Nonsystematic memory M versus systematic memory $2M$ ODP codes.

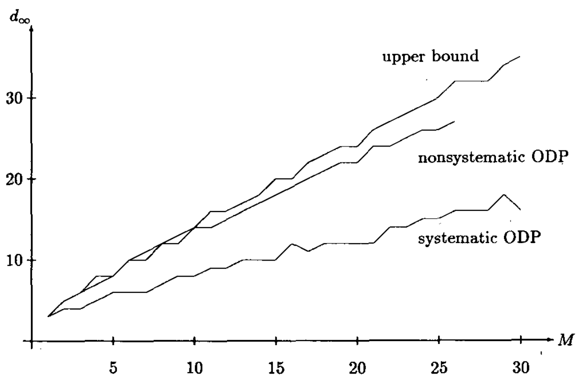

Fig. 8. Free distance for some classes of convolutional codes.

$M = 28$ using the Griesmer bound for block codes [24], [25]:

$$\sum_{i=0}^{k-1} \left\lceil \frac{d_\infty}{2^i} \right\rceil \le 2(M+k), \qquad k \ge 1,\ R = 1/2$$

where $\lceil \cdot \rceil$ denotes the ceiling function.

VIII. Conclusion

In this paper we have shown that the distance profile can be exploited in a very efficient way to prune the code tree when calculating the distance spectrum of rate $R = 1/2$ convolutional codes. FAST can easily be extended to other rates. We have also reported extensive lists of good rate $R = 1/2$ convolutional encoders. These tables provide encoders that can be used in practice.

References

[1] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 751–772, Oct. 1971.

[2] J. L. Massey and D. J. Costello, Jr., "Nonsystematic convolutional codes for sequential decoding in space applications," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 806–813, Oct. 1971.

[3] R. Johannesson, "Robustly optimal rate one-half binary convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 464–468, July 1975.

[4] P. R. Chevillat and D. J. Costello, Jr., "Distance and computing in sequential decoding," *IEEE Trans. Commun.*, vol. COM-24, pp. 440–447, Apr. 1976.

[5] G. D. Forney, Jr., "Use of a sequential decoder to analyze a convolutional code structure," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 793–795, Nov. 1970.

[6] J. P. Odenwalder, "Optimal decoding of convolutional codes," Ph.D. dissertation, Dep. Syst. Sci., Sch. Eng. Appl. Sci., Univ. of California, Los Angeles, 1970.

[7] L. R. Bahl, C. D. Cullum, W. D. Frazer, and F. Jelinek, "An efficient algorithm for computing the free distance," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 437–439, Nov. 1972.

[8] K. J. Larsen, "Comments on 'An efficient algorithm for computing free distance' by Bahl *et al.*," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 577–579, July, 1973.

[9] D. Divsalar, "Performance of mismatched receivers on bandlimited channels," Ph.D. dissertation, Dep. Syst. Sci., Sch. Eng. Appl. Sci., Univ. of California, Los Angeles, 1978.

[10] M. Cedervall, "Contribution to the decoding and structure of convolutional codes," Ph.D. dissertation, Dep. of Comput. Eng., Lund Univ., Lund, Sweden, 1983.

[11] D. J. Costello, Jr., "A construction technique for random-error-correcting convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 631–636, Sept. 1969.

[12] J. L. Massey, "Error bounds for tree codes, trellis codes, and convolutional codes with encoding and decoding procedures," in *Coding and Complexity, CISM Courses and Lectures no. 216*, G. Longo, Ed. Vienna, Austria: Springer-Verlag, 1975.

[13] J. L. Massey and M. K. Sain, "Inverses of linear sequential circuits," *IEEE Trans. Comput.*, vol. C-17, pp. 330–337, Apr. 1968.

[14] R. Johannesson and E. Paaske, "Further results on binary convolutional codes with an optimum distance profile," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 264–268, Mar. 1978.

[15] J. Conan, "The weight spectra of some short low-rate convolutional codes," *IEEE Trans. Commun.*, vol. COM-32, pp. 1050–1053, Sept. 1984.

[16] L. R. Bahl and F. Jelinek, "Rate 1/2 convolutional codes with complementary generators," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 718–727, Nov. 1971.

[17] A. J. Vinck, private communication, 1982.

[18] H. J. Helgert, "Short constraint length rate 1/2 'quick-look' codes," *IEEE Trans. Commun.*, vol. COM-23, pp. 768–773, July 1975.

[19] ——, "Correction to 'Short constraint length rate 1/2 'quick-look' codes'," *IEEE Trans. Commun.*, vol. COM-24, p. 286, Feb. 1976.

[20] J. J. Bussgang, "Some properties of binary convolutional code generators," *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 90–100, Jan. 1965.

[21] R. Johannesson, "Some long rate one-half binary convolutional codes with an optimum distance profile," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 629–631, Sept. 1976.

[22] J. L. Massey, private communication, 1974.

[23] J. A. Heller, "Sequential decoding: Short constraint length convolutional codes." Jet Propulsion Lab., California Inst. Technol., Pasadena, Space Program Summary 37-54, vol. 3, Dec. 1968, pp. 171–174.

[24] J. Layland and R. McEliece, "An upper bound on the free distance of a tree code," Jet Propulsion Lab., California Inst. Technol., Pasadena, Space Program Summary 37-62, vol. 3, Apr. 1970, pp. 63–64.

[25] J. H. Griesmer, "A bound for error-correcting codes," *IBM J. Res. Develop.*, vol. 4, no. 5, 1960.