On a Technique to Calculate the Exact Performance of a Convolutional Code

Mark R. Best, Marat V. Burnashev, Yannick Lévy, Asya Rabinovich, Peter C. Fishburn, A. R. Calderbank, Member, IEEE, and Daniel J. Costello, Jr., Fellow, IEEE

Abstract—A Markovian technique is described to calculate the exact performance of the Viterbi algorithm used as either a channel decoder or a source encoder for a convolutional code. The probability of information bit error and the expected Hamming distortion are computed for codes of various rates and constraint lengths. The concept of tie-breaking rules is introduced and its influence on decoder performance is examined. Computer simulation is used to verify the accuracy of the results. Finally, we discuss the issue of when a coded system outperforms an uncoded system in light of the new results.

Index Terms—Convolutional codes, bit error probability, Viterbi algorithm.

Authors’ Note

This paper is dedicated to the memory of the first author, Marc R. Best, whose untimely death in 1987 cut short a promising career in research. The impetus for this paper came from recent work by Calderbank, Fishburn, and Rabinovich [2], [7] on calculating the performance of the Viterbi algorithm used as a source encoder. Costello and Levy pointed out that their method was analogous to one used by Morrissey [3], [4] to calculate the performance of the Viterbi algorithm used as a channel decoder. Subsequently, Herro1 reminded the authors of an earlier unpublished manuscript by Best and related work by Burnashev which contained similar results.

I. INTRODUCTION

A CONVOLUTIONAL code is a set of code sequences generated by a finite-state machine whose states define a trellis and allow efficient maximum-likelihood decoding techniques such as the Viterbi algorithm. The Viterbi algorithm was first used to decode channel sequences using a convolutional code [5]. More recently, the Viterbi algorithm has been used to encode source sequences using a convolutional code [1]. The algorithm finds a maximum-likelihood trellis path to decode a channel sequence or to encode a source sequence.

The performance of a convolutional code can be evaluated by computing either 1) the expected number of information bits that differ between the transmitted sequence and the decoded sequence or 2) the expected number of source bits that differ between a source sequence and the encoded sequence. The performance is usually expressed as the information bit error probability in case 1) and the expected Hamming distortion in case 2). Because exact calculation is difficult in both cases, simulations or upper bounds are often used to estimate these quantities.

The exact calculation of decoder error probability for binary convolutional codes was first investigated by Morrissey [3] using a suboptimum feedback decoding technique. This was then extended to Viterbi decoding for the single case of a rate 1/2, 2-state code [4]. Later, Schalkwijk, Post, and Aarts [6] developed a method for calculating error probability using another maximum likelihood decoding technique called “syndrome decoding.” Each of these approaches used a Markov chain to describe the decoding procedure. More recently, Calderbank, Fishburn, and Rabinovich [2] used a similar Markov chain approach to evaluate the source encoding performance of binary convolutional codes.

The present paper utilizes the approach in [2] to evaluate the exact performance of convolutional codes with Viterbi decoding, thereby extending the results of [4] to codes of different rates and constraint lengths. The probability of information bit error when using a convolutional code as a channel code, and the expected Hamming distortion when using a convolutional code as a source code, can both be calculated using this approach. We discuss the Viterbi algorithm, describe its associated Markov chain, and formulate expressions for the expected Hamming distortion and the probability of information bit error. We then use these expressions to compute the probability of information bit error for codes of various rates and constraint lengths. To avoid confusion between trellis states and Markov chain states, we will call the states of a trellis vertices and the states of a Markov chain states.

II. THE VITERBI ALGORITHM AND TIE-BREAKING RULES

The Viterbi algorithm is a dynamic maximum-likelihood procedure that updates a set of trellis vertices at every time unit. For a rate k/n convolutional code with 2^r vertices, the
Viterbi algorithm computes a metric for the $2^n$ paths entering each vertex at a new time unit from the metrics at the preceding time unit and the received sequence. It then selects a surviving path corresponding to each vertex with minimum metric. The smallest surviving path metric is then subtracted from all $2^n$ metrics to yield a metric vector over the $2^n$ vertices. It therefore keeps track of a metric vector and $2^n$ surviving paths. When the received sequence terminates, the Viterbi algorithm chooses a final surviving path that ends in a zero vertex of the final metric vector, i.e., a path with minimum metric.

The associated Markov chain consists of the one-step trajectories of the Viterbi algorithm. Its states are the possible metric vectors. Transitions from one state to another depend on the received sequence if the Viterbi algorithm is used as a channel decoder, or on the source sequence if it is used as a source encoder. When used as a channel decoder, linearity allows us (without loss of generality) to assume that the all-zero sequence has been sent. If the channel is the binary symmetric channel, as shown in Fig. 1, an error occurs with probability $p$ and no error occurs with probability $1-p$. Thus $p$ is the probability that a transmitted 0 is received as a 1 or a transmitted 1 is received as a 0. When used as a source code, we assume that the source produces 1 with probability $p$ and 0 with probability $1-p$. Thus the same Markov chain results in both cases [2], [4].

A potential problem is encountered by the Viterbi algorithm when a tie occurs at a given vertex, that is, when two or more paths produce the same metric at a given time unit. This happens when two or more maximum-likelihood paths come into a given vertex of the trellis; the algorithm can choose any of those paths and still remain maximum-likelihood. This is not a problem when computing expected Hamming distortion, which is the same regardless of the path selected by the decoder. However, in the case of channel decoding, only one path can be correct and the selection may determine whether an information bit error occurs. Since we assume the all-zero sequence was transmitted, the way the decoder breaks ties should not favor the decoding of the all-zero path since this would bias the result. We therefore define a fair tie-breaking rule as one which does not favor any particular sequence when a tie occurs.

We consider three types of fair tie-breaking rules. The first, the lexicographic tie-breaker, selects the path which, looking backward in the trellis, first had the smallest metric among all tied paths, i.e., the path for which more errors occurred recently. The opposite of this rule, the anti-lexicographic tie-breaker, selects the path which most recently had the largest metric. The third rule, the coin-flip tie-breaker, randomly selects a maximum-likelihood path. An example of an unfair tie-breaking rule is to always select the path coming from the highest vertex in the trellis diagram. This would favor the decoding of the all-zero path when the trellis is drawn with the all-zero path on top.

Another possibility is to use a Markovian tie-breaking rule that only considers the preceding vertex. The lexicographic and anti-lexicographic rules are usually non-Markovian since they may have to look back further than the preceding vertex. On the other hand, the coin-flip tie-breaker is both fair and Markovian, but can be more difficult to analyze, as we will see in Section V.

In some of our calculations, we use a Markovian but unfair deterministic tie-breaking rule for simplicity. We define the 1-step lexicographic tie-breaker as follows. If one of the maximum-likelihood paths into a vertex comes from a vertex whose metric is smaller than the others, we choose that path. Otherwise, the tie is broken by looking at the branch labels from right to left, finding the first bit in which the two paths disagree, and then picking the path that agrees with the received bit in that position. However, this rule is unfair since certain paths, depending on the transmitted code sequence, will be favored over others. We can try to correct for this by reversing the rule, i.e., by picking the path that first disagrees with the received sequence. Another possibility is to choose a branch at random when the 1-step lexicographic tie-breaker does not resolve the issue. The result for this procedure would fall between the other two results.

The 1-step anti-lexicographic tie-breaker is similar to the 1-step lexicographic tie-breaker, except that we choose the maximum-likelihood path which comes from a vertex whose metric is greater than the others. If this fails, we break the tie as in the 1-step lexicographic tie-breaker.

The binary symmetric channel results in a quantized channel output with two values and is equivalent to the source coding problem with binary source outputs. Other important channels, such as the binary symmetric erasure channel (see Fig. 1), which quantizes the output into three values (0.1, and $\epsilon$), can also be studied. This channel is equivalent to the source coding problem with ternary source outputs. Ultimately, soft decision decoding uses unquantized received values in the Viterbi algorithm, which corresponds to source coding with a continuous source. Although the last problem was studied for source coding by Calderbank and Fishburn [7], it is much more complex since the number of states in the Markov chain is infinite. Therefore, in the remainder of this paper, we only consider channels or sources with a finite number of outputs.
III. THE VITERBI ALGORITHM'S MARKOV CHAIN

In this section we use the Viterbi algorithm for its original purpose of channel decoding. We restrict ourselves to the binary symmetric channel and consider the Hamming distance $d_H$ between two sequences, defined as the number of bits in which they differ, as the metric. The variables introduced for the Markov chain are the same as for the source encoding problem since the Markov chain is the same. Other channels and metrics can be used in a similar fashion with suitably defined Markov chains.

Let $R^N$ be a received sequence of length $N$ time units. At each vertex $\gamma$ of the trellis, a maximum-likelihood path is chosen from among the paths leading to that vertex and a metric $D^N_\gamma$ is computed from the metrics at time unit $N-1$. Let $A^N_\gamma$ be the maximum-likelihood path. Then

$$D^N_\gamma = d_H(A^N_\gamma, R^N).$$

The relative metric $\bar{D}^N_\gamma$ at vertex $\gamma$ is obtained by subtracting the minimum metric among all the vertices from $D^N_\gamma$

$$\bar{D}^N_\gamma = D^N_\gamma - \min_{\delta} (D^N_\delta).$$

The metric vector at time unit $N$ is

$$\bar{D}^N = (\bar{D}^N_0, \bar{D}^N_1, \ldots, \bar{D}^N_{2^N-1}).$$

These internal states of the Viterbi decoder form the Markov chain, with the received symbol $r$ in the sequence $R^N$ at time $N$ determining the transitions from one state to another. We let $M$ denote the number of recurrent states that adhere to (3).

Fig. 2 shows the trellis diagram of the rate 1/2, 2-state convolutional code with generator matrix $[1, 1 + D]$. Time evolves from left to right following the arrows. For this code, the Hamming distance between a branch label and a channel output is at most 2, so that the possible metric vectors or states of the Markov chain are $(2, 0), (1, 0), (0, 0), (0, 1), (0, 2)$.

The transition probability matrix $T$ for the resulting 5-state Markov chain can be easily computed by checking which received signals determine a transition from one metric vector to the next. The conditional probability of this received signal defines the transition probability. Fig. 3 shows the Markov chain for the rate 1/2, 2-state code with

$$T = \begin{pmatrix}
(1 - p)^2 & 0 & 2p(1 - p) & 0 & p^2 \\
(1 - p)^2 & 0 & 2p(1 - p) & 0 & p^2 \\
0 & (1 - p) & p^2 & (1 - p)^2 & 0 \\
p(1 - p) & 0 & p^2 + (1 - p)^2 & 0 & p(1 - p) \\
p(1 - p) & 0 & p^2 + (1 - p)^2 & 0 & p(1 - p)
\end{pmatrix}. \tag{4}
$$

We compute the probability of information bit error in channel decoding, or the expected Hamming distortion in source encoding, from the steady-state behavior of the Viterbi algorithm. Let $\pi = (\pi_0, \pi_1, \ldots, \pi_{M-1})'$ be the vector of steady-state probabilities of being in states 0 to $M - 1$ of the Markov chain. Then $\pi$ is given by

$$\pi = T^\ast \pi$$

and $\pi_0 + \pi_1 + \ldots + \pi_{M-1} = 1$. The steady-state probability vector for the rate 1/2, 2-state convolutional code is

$$\pi = \frac{1}{1 + 3p^2 - 2p^3} \begin{pmatrix}
1 - 4p + 8p^2 - 7p^3 + 2p^4 \\
2p - 5p^2 + 5p^3 - 2p^4 \\
2p - 3p^2 + 2p^3 \\
2p^2 - 3p^3 + 2p^4 \\
p^2 + p^3 - 2p^4
\end{pmatrix}. \tag{5}
$$

IV. EXACT CALCULATION OF EXPECTED HAMMING DISTORTION

The expected Hamming distortion corresponds to the expected number of bits that must be changed in a source sequence to obtain the closest code sequence, i.e., the closest path through the trellis. For a state $\bar{D}$ of the decoder, suppose that a source symbol $r$ causes a transition from $\bar{D}$ to $\bar{D}'$. The one-step aggregate distortion $g(r, \bar{D})$ is defined as the total number of bits that differ from the source symbol $r$ along all surviving paths

$$g(r, \bar{D}) = |D_0' - D_0| + |D_1' - D_1| + \ldots + D_{2^N-1} - D_{2^N-1}. \tag{6}
$$

Calderbank and Fishburn [7] have shown that the expected Hamming distortion $\mu$ per dimension for a rate $k/n$ convolu-


where \( q_r \) is the probability of source symbol \( r \) and \( \pi_\overline{D} \) is the steady-state probability of state \( \overline{D} \). For the rate 1/2, 2-state code, the one-step aggregate distortions for the five states of the Markov chain are \( g(r, (0, 2)) = g(r, (2, 0)) = 0 \) and \( g(r, (0, 1)) = g(r, (0, 0)) = g(r, (1, 0)) = 1 \) for all possible source symbols \( r \). Thus

\[
\mu = \frac{1}{4} (\pi_1 + \pi_2 + \pi_3) = \frac{2p - 3p^2 + 2p^3}{2(1 + 3p^2 - 2p^3)}.
\]

This calculation is straightforward because it is computed as the average of one-step aggregate distortions. This is possible since the minimum distortion among all 2
\( \nu \) paths equals the average distortion over all 2
\( \nu \) paths in the limit as the length of the encoded source sequence increases. Computations for other codes are described in [7].

In the case of channel decoding, the probability of codeword bit error is the expected fraction of bits that one has to change in a received sequence to obtain the closest code sequence, i.e., the closest path through the trellis. Because of the analogy between source encoding and channel decoding, exactly the same procedure as described above for calculating the expected Hamming distortion of a source encoder can be used to calculate the probability of codeword bit error of a channel decoder. However, because we must consider explicitly the path decoded by the Viterbi algorithm to determine information bit errors, the calculation of the probability of information bit error described in the next section is more difficult.

V. Exact Calculation of Information Bit Error Probability

In this section we compute the probability of information bit error by examining the path decoded by the algorithm. At a given state of the Markov chain, we want to compute the exact error probability per bit for the current \( k \) information bits. To do this, we must consider all received sequences that stem from a particular decoded branch. Given metric state \( \overline{D} \) and a received sequence \( r^l \) of \( l \) branches, let \( P(r^l, \overline{D}) = i/k \) if \( i \) information bits are decoded incorrectly, \( i = 0, 1, \ldots, k \). Averaging over all metric states and received sequences, the probability of error can then be expressed as

\[
P_e = \sum_{\overline{D}, r^l} \pi_{\overline{D}} q_r P(r^l, \overline{D}).
\]

where \( q_r \) is the probability of receiving sequence \( r^l \). It is understood that the received sequences in (10) are mutually disjoint and exhaust all possibilities that cause errors in the current time unit.

The main problem encountered in calculating (10) is cataloging all possible future received sequences that cause information bits in a given time unit to be decoded incorrectly. If we use the lexicographic tie-breaker for the rate 1/2, 2-state code, only length-1 sequences need to be examined. In this case, a decoding error occurs if and only if i) the correct node has a nonzero relative metric, or ii) if both nodes have zero relative metrics and channel errors of type (0, 1) or (1, 0) occur. Thus \( l = 1 \), and \( P(r, \overline{D}) \in \{0, 1\} \) since \( k = 1 \).

For the rate 1/2, 2-state code, \( P(r, \overline{D}) = 0 \) for \( \overline{D} = (0, 2) \) or \( (0, 1) \) (no error regardless of the future received sequence) and \( P(r, \overline{D}) = 1 \) for \( \overline{D} = (2, 0) \) or \( (1, 0) \) (an error always occurs). In addition, \( P(r, (0, 0)) = 1 \) if \( r = (0, 1) \) or \( r = (1, 0) \), and \( P(r, (0, 0)) = 0 \) if \( r = (0, 0) \) or \( r = (1, 1) \). Thus for this code

\[
P_e = 2p(1-p)\pi_0 + \pi_3 + \pi_4 = \frac{7p^2 - 12p^3 + 10p^4 - 4p^5}{1 + 3p^2 - 2p^3}.
\]

The anti-lexicographic tie-breaker requires consideration of sequences of length 2, so we examine all \( l = 2 \) sequences as follows:

<table>
<thead>
<tr>
<th>State ( \overline{D} )</th>
<th>Received Sequence ( r^2 )</th>
<th>( P(r^2, \overline{D}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 : (0, 2)</td>
<td>((\cdot, \cdot, \cdot))</td>
<td>0</td>
</tr>
<tr>
<td>1 : (0, 1)</td>
<td>(0, 0, (\cdot)) or (1, 1, (\cdot))</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(0, 1, 0) or (1, 0, 1)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(0, 1, 1) or (1, 0, 1)</td>
<td>1</td>
</tr>
<tr>
<td>2 : (0, 0)</td>
<td>(0, 0, (\cdot)) or (1, 1, (\cdot))</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(0, 1, (\cdot)) or (1, 0, (\cdot))</td>
<td>1</td>
</tr>
<tr>
<td>3 : (1, 0)</td>
<td>(0, 1, (\cdot)) or (1, 0, (\cdot))</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(0, 0, 0) or (1, 1, 0)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(0, 0, 1) or (1, 1, 0)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(0, 0, 1) or (1, 1, 1)</td>
<td>1</td>
</tr>
<tr>
<td>4 : (2, 0)</td>
<td>((\cdot, \cdot, \cdot))</td>
<td>1</td>
</tr>
</tbody>
</table>

Then

\[
P_e = p(1-p)\pi_1 + 2p(1-p)\pi_2 + (4p^3 - 7p^2 + 4p)\pi_3 + \pi_4
= \frac{p^2(7 - 8p - 8p^2 + 26p^3 - 24p^4 + 8p^5)}{1 + 3p^2 - 2p^3}.
\]

For the coin-flip tie-breaker, the length \( l \) of sequences that one must look at is too large, and we need to use a recursive technique. We compute the bit error probability for the coin-flip tie-breaker by solving for the probability

\[
P_e = \sum_{r^l} q_i P(r^l, \overline{D})
\]

of an error at a vertex in the trellis conditioned on being in state \( \overline{D} \) at that step. Then

\[
P_e = \sum_{\overline{D}} \pi_{\overline{D}} P(\overline{D}).
\]

As before, \( P(0, 0) = 0 \) and \( P(4) = 1 \). To compute \( P(1), P(2), \) and \( P(3) \), a forward recursion can be used to account for all \( r^l \) possibilities by looking at one step at a time. Specifically, given a starting state \( \overline{D} \), \( P(\overline{D}) \) can be computed by the recursive formula

\[
P(\overline{D}) = \sum_r q_r P(\overline{D}_r).
\]
where \( r \) can be any received symbol and \( D \) is the resulting Markov chain state. Using this forward recursion for states 1, 2, and 3 under the coin-flip tie-breaker gives

\[
P(1) = \frac{p(1-p)}{2} \left[ 1 - P(2) \right] + \frac{p(1-p)}{2} \left[ P(2) \right]
\]

\[
P(2) = p(1-p)\left[ P(1) \right] + p^2 \left[ 1 - P(3) \right] + p(1-p)\left[ 1 - P(1) \right] + p(1-p)\left[ P(3) \right]
\]

\[
P(3) = \frac{1}{2} + \frac{(1-p)^2}{2} \left[ P(2) \right] + \frac{p^2}{2} \left[ 1 - P(2) \right] + p(1-p).
\]

The solutions for the latter two \( P(D) \) are

\[
P(2) = \frac{4p(1-p)}{2 - p + 4p^2 - 4p^3}
\]

(15)

\[
P(3) = \frac{2 + 7p - 12p^2 + 13p^3 - 12p^4 + 4p^5}{2(2 - p + 4p^2 - 4p^3)}.
\]

Then

\[
P_e = P(1)\pi_1 + P(2)\pi_2 + P(3)\pi_3 + \pi_4
\]

(16)

\[
= p^2 \left( 14 - 23p + 16p^2 + 2p^3 - 16p^4 + 8p^5 \right)
\]

\[
\left( 1 + 3p^3 - 2p^4 \right) \left( 2 - p + 4p^2 - 4p^3 \right)
\]

for the coin-flip tie-breaker.

The preceding error probabilities are plotted as functions of \( p \) in Fig. 4. We note for this particular code that the lexicographic tie-breaker provides a fair and better tie-breaking rule than the others. However, the anti-lexicographic rule might be better for other codes. In any case, we note that the probability of error associated with the coin-flip tie-breaker is always between the other two. Hence, for a given code, either the anti-lexicographic or the lexicographic tie-breaking rule is more advantageous than the simple coin-flip tie-breaking rule that is used in practice.

VI. CALCULATION OF INFORMATION BIT ERROR

PROBABILITY FOR OTHER RATES AND CONSTRAINT LENGTHS

New problems arise when the preceding approach is used to calculate the probability of information bit error for more complicated rates and larger constraint lengths. One problem is a rapid increase in the number of Markov chain states as the constraint length increases. For example, while the 2-state, rate 1/2 code has a Markov chain with 5 states corresponding to all possible metric vectors, the 4-state, rate 1/2 code with generators \([101, 111] \) has 30 Markov chain states, and a typical 8-state code has several hundred states. This prevents us from computing the exact probability of information bit error or the expected Hamming distortion for larger constraint lengths.

Another problem is the length of the received sequences \( r^t \) that must be examined to compute \( P(r^t, D) \). As the constraint length increases, the length of the sequence can become large and create a very large number of terms when calculating the probability of error. A program was written to determine all the received sequences that must be examined. However, due to the large number of terms for codes other than the rate 1/2, 2-state code, we decided to directly introduce the value of the probability of occurrence of each sequence for a particular value of \( p \), instead of keeping it as a function of \( p \). Thus instead of a formula as a function of \( p \), we can compute the probability of information bit error for any given crossover probability \( p \).

Finally, for rate \( k/n \) codes with \( k > 1 \), up to \( k \) information digits can be decoded incorrectly at each step. In this case, each vertex of the trellis can be labeled with the number of errors caused by passing through it, and that information can then be included in the state of the Markov chain. Another technique, simpler to implement, is to consider each information bit separately and average the probability of error obtained for each bit.
We calculated the information bit error probability for the 2-, 4-, and 8-state rate 1/2 codes, the 2-state rate 1/3 and 1/4 codes, and the 4- and 8-state rate 2/3 codes in the list of optimum free distance codes in [8]. The best deterministic tie-breaking rule (1-step lexicographic or 1-step anti-lexicographic) was used in the calculations. The information bit error probability curves are shown in Fig. 5. Computer simulation results are also shown for each of these codes. Although the fit is not exact, the error (caused by simulation inaccuracies and the deterministic, i.e., unfair, tie-breaking rules) is within the expected range. The straight line corresponds to an uncoded system for which the probability of information bit error equals the crossover probability $p$ of the binary symmetric channel.

VII. EXPRESSING THE INFORMATION BIT ERROR PROBABILITY AS A TAYLOR SERIES IN $p$

The Markov chain approach to computing the probability of information bit error as a function of the crossover probability $p$ of a binary symmetric channel leads to a rational fraction in $p$. These fractions can be expanded in a Taylor series to show the dominant terms. The 2-state, rate 1/2 code probability of information bit error with the lexicographic tie-breaking rule can be expanded into

$$P_e = 7p^2 - 12p^3 - 11p^4 + 46p^5 + 9p^6 + O(p^7).$$

The coin-flip and anti-lexicographic tie-breakers have

$$P_e = 7p^2 - 8p^3 - 31p^4 + 64p^5 + 86p^6 + O(p^7)$$

and

$$P_e = 7p^2 - 8p^3 - 29p^4 + 64p^5 + 47p^6 + O(p^7)$$

respectively. This shows that the best-to-worst rules for small $p$ are 1) the lexicographic rule, 2) the coin-flip rule, and 3) the anti-lexicographic rule.

We note also that the first term in the Taylor series expansion can be anticipated from the union bound, which states that a code with free distance $d_{free}$ has a probability of information bit error upper bounded by [8]

$$P_e < Kp^{\frac{d_{free}}{2}} + \cdots$$

where $K$ is a constant depending on the path multiplicity of the code. The 2-state code has distance 3, so $P_e < Kp^{3/2}$. Hence, the Taylor series expansion can only start with a term in $p^3$, which corresponds to the above calculations.

It is possible for other codes to interpolate the points we obtained for different values of $p$ to get an idea of the Taylor series expansion for those codes. Since the free distance of the 4-state code is 5, the Taylor series expansion must start with a term in $p^5$, and interpolation for various tie-breaking rules can provide an estimate of the leading coefficients for those rules.

VIII. PROBABILITY OF INFORMATION BIT ERROR OF A CODED SYSTEM VERSUS AN UNCODED SYSTEM

Figs. 4 and 5 show the probability of information bit error as a function of $p$ for different codes, as well as $P_e = p$ for an uncoded system. An interesting point on these figures is the crossover probability $p$ below which a coded system performs better than the uncoded system. Table I shows this value for the codes we studied, along with the crossover probability at which the channel capacity equals the code rate.

The table shows that the crossover probability below which coding performs better than no coding is closely related to the channel capacity. In fact, for the codes studied, as constraint length increases, the crossover probability below which the coded system outperforms the uncoded system approaches the probability at which the capacity

$$C = 1 - H_2(p) = 1 + p \log p + (1 - p) \log (1 - p)$$
of the BSC equals the rate of the code, as predicted by Shannon’s coding theorem [9].

Another interesting comparison is gained from an analysis of the weight structure of a code obtained by examining the loops in the state diagram of the encoder [8]. Long codewords with low weight, which are the best candidates for causing multiple bit errors, are generated by cycling around the lowest average weight loop in the state diagram. If the channel crossover probability approaches half the value of the minimum average weight loop, the decoder is in danger of choosing one of these long low-weight codewords, thus causing multiple bit errors. This can result in coded performance becoming worse than uncoded performance. For example, the minimum average weight loop for the three rate 1/2 codes in the above table is 0.50 for the 2-state code and 0.25 for both the 4- and 8-state codes. This suggests that poor performance, i.e., worse than uncoded, should occur at channel crossover probabilities of about 0.25 for the 2-state code and 0.125 for the 4- and 8-state codes. The results shown in Table I are consistent with these predictions.

IX. CONCLUSION

In this paper we have described a Markovian approach to calculating the performance of the Viterbi algorithm in decoding convolutional codes used as source codes or channel codes. The Markov chain associated with the Viterbi algorithm was studied in detail for the 2-state, rate 1/2 code. We computed this code’s expected Hamming distortion as a function of the source distribution and its probability of information bit error as a function of the binary symmetric channel crossover probability.

Problems related to this method of calculation were described, and results for different rates and constraint length codes were compared to computer simulations. Our approach also results in a Taylor series expansion that describes a code’s performance for small \( p \) and is consistent with upper bounds previously computed. Finally, we noted that for the codes examined, the crossover probability above which a coded system performs worse than an uncoded system is consistent with what would be expected from Shannon’s coding theorem and from an analysis of the weight structure of the code.

Although the Markovian approach has limitations, especially for larger constraint lengths, it gives insight into the behavior of the Viterbi algorithm for channel decoding and source encoding. Extensions to erasure channels, such as shown in Fig. 1, and to nonbinary sources, are possible.

REFERENCES