Systematic feed-forward convolutional encoders are better than other encoders with an M-algorithm decoder

Osthoff, Harro; Anderson, John B; Johannesson, Rolf; Lin, Chin-foo

Published in:
IEEE Transactions on Information Theory

DOI:
10.1109/18.661531

Published: 1998-01-01

Citation for published version (APA):


Systematic Feed-Forward Convolutional Encoders Are Better than Other Encoders with an $M$-Algorithm Decoder

Harro Osthoff, John B. Anderson, Fellow, IEEE,
Rolf Johannesson, Fellow, IEEE, and Chin-foo Lin

Abstract—Consider nonbacktracking convolutional decoders that keep a fixed number of trellis survivors. It is shown that the error performance of these depends on the early part of the distance profile and on the number of survivors kept, and not on the free distance or the details of the code generators. Particularly, the encoder may be feed-forward systematic without loss. Furthermore, this kind of encoder solves the correct path loss problem in reduced-search decoders. Other kinds do not. Therefore, with almost any other decoding method than the Viterbi algorithm, systematic feed-forward encoders should be used. The conclusions in this correspondence run counter to much accepted wisdom about convolutional codes.

Index Terms—Correct-path loss problem, list decoder, $M$-algorithm decoder, systematic feed-forward encoders.

I. INTRODUCTION

Growing evidence in recent research has shown that the error performance of a channel decoder depends primarily on the decoder’s resources, and only secondarily on the qualities of the error-correcting code itself, such as its free distance and the space-state size of its minimal encoder. It is well established, for example, that the performance of a sequential decoder of the Fano or stack algorithm type depends on how many code trellis paths the decoder can visit. The precise choice of the code matters little, so long as the code has a good distance profile and a reasonable memory.

In a traditional sequential decoder, the design begins by assuming a long code and by assuming that the decoding of a data symbol will be correct with certainty; an analysis then finds the expected number of code trellis paths (or tree paths, or tree branches) that are required to do this. So long as the code rate is less than the cutoff rate, this expectation is finite, but the actual path number outcome may on occasion exceed any bound, a phenomenon called erasure. The working storage for these paths and the speed in which they may be viewed comprise the resources of the decoder, and it is the need for these that depends weakly on the code.

In more recent times, this design philosophy has often been reversed. Rather than fix the error probability at zero, the storage of the paths is fixed; by means of analysis and measurement, the error probability or the number of correctable errors is found. Two examples of this philosophy are the Viterbi algorithm and the $M$-algorithm.
algorithm (list decoder). In the first, the storage is fixed to the encoder state-space size, and depends therefore critically on the code. In the second, the storage is fixed at $M$ paths, independently of the code. Decoders of the $M$-algorithm type, whose storage is fixed a priori, show a strong independence between error performance and code.

The purpose of this correspondence is to relate the storage and error performance of simple breadth-first decoding of convolutional codes, and to investigate how dependent they really are. We will show that the important aspect of a code is the early growth of its state-space size, and depends therefore critically on the code. We do this by two methods, for best-neighbor in the trellis, or they can be long. The second measure of error begins at a certain trellis level, given that decoding has been error event probability of a decoder is the probability that a trellis path splits from the correct path and end when it merges again. Error performance has two meanings in this correspondence. The proper value for $t_{d2}$, where $t_{d2}$, the decoder decision depth, is the number of trellis levels behind the front of the search at which a data bit is released as output. In each basic cycle of the algorithm, a data bit is released at $t_{d2}$, behind the front, all paths are extended one trellis level deeper, and the best $M$ survivors are retained. We also delete any paths that do not correspond to the released data bit and that represent state duplications of a better path in storage, although these features do not much change the decoder performance. Further implementation details appear in [3].

The proper value for $t_{d2}$ (see [3] or [14]) for nonbacktracking decoders and known, concrete codes is approximately

$$t_{d2} \approx 2e/\rho c$$

where $R = b/c$ is the code rate, $\epsilon$ is the number of errors the decoder is assumed to correct, and $\rho$ is the Gilbert–Varshamov parameter, i.e., the solution of

$$h_B(\rho) = 1 - R$$

where $h_B(\cdot)$ is the binary entropy function. We use this $t_{d2}$ in our decoders, although considerably shorter values may be used without compromising $P_w$ very much. Regarding the size of $M$, certain properties are known. If the design of the decoder is to guarantee the correction of any $e$ errors in the length $t_{d2}$, a bounded-distance decoder (BDD) design, then $M$ must satisfy [15]–[17]

$$M \approx (2^{1-R} - 1)^{-e}.$$
On the other hand, if the design specifies that the decoder must achieve a certain $P_{e\text{c}}$, then the path numbers are entirely different; this is discussed in [3] and [18].

It is well known that a good computational performance for sequential decoding requires a rapid initial growth of the column distances. This led to the introduction of the distance profile [1]:

Definition: Let $C$ be a convolutional code encoded by a minimal-basic generator $G$ [19] of memory $m$. The $(m + 1)$-tuple

$$d^e = (d^e_0, d^e_1, \ldots, d^e_m)$$

where $d^e_j$, $0 \leq j \leq m$, is the $j$th-order column distance of $G$, is called the distance profile of the code $C$.

A generator matrix of memory $m$ is said to have a distance profile $d^e$ superior to a distance profile $d^{e'}$ of another generator matrix of the same rate $R$ and memory $m$, if there is some $l$ such that

$$d^e_j = d^{e'}_j, \quad j = 0, 1, \ldots, l - 1$$

$$d^e_j > d^{e'}_j, \quad j = l.$$

(6)

Moreover, a convolutional code is said to have an optimum distance profile (ODP) if there exists no generator matrix of the same rate and memory whose code has a better distance profile. An ODP generator causes the fastest possible initial growth of the minimal separation between the encoded paths diverging at the root in a code tree.

Furthermore, it has been shown [20] that there exists a binary, rate $R = h/c$, time-invariant convolutional code with a generator matrix of memory $m$ whose column distances satisfy the inequality

$$d^e_j \geq \rho c(j + 1)$$

(7)

for $0 \leq j \leq m$, and where $\rho$ is the Gilbert–Varshamov parameter. From (7) it follows that at length $t_D = j + 1$ we can have a distance $d$ between paths out of a trellis node that is at least $\rho c t_D$, or equivalently

$$t_D \leq d/\rho c.$$  

(8)

This again suggests rule (3) for the decoder decision depth.

The organization of the correspondence is as follows. In Section II we give the observed $P_{e\text{c}}$ versus $E_b/N_0$ for an $M$-algorithm decoder with best-$d_{t_{\text{dec}}}$ QLI, and feed-forward systematic encoders, and find that $P_{e\text{c}}$ depends only on the distance profiles and $M$. In Section III we estimate $P_{e\text{c}}$ by an error sequence counting argument, and come to the same conclusion. Section IV compares the $P_{e\text{c}}$ of the $M$ and Viterbi algorithms. In Section V we turn to the BER, a quantity that depends critically on how the decoder recovers from losses of the correct path; we conclude that only the systematic encoder can improve upon the Viterbi BER.

II. DISTANCE PROFILE AND $M$: SIMULATIONS

In this section we will establish by tests of an actual encoder/decoder that the error event performance of an $M$-algorithm type decoder depends for the BSC almost entirely on the distance profile and apparently not on the details of the code generators or on the free distance. The tests measure the error event frequency, as discussed in Section I, and thus do not measure the efficiency of recovery from an error event (this is studied in Section V). (In the following simulations we used frames consisting of 1024 information bits. The number of simulated frames was chosen such that at least 100 distinct error events occurred; for example, with BSC crossovers $p = 0.037$ and 0.033, the numbers of frames were respectively 250,000 and 500,000 or more. In addition, there were selected tests of very long frames to confirm that catastrophic events occur with negligible probability.)

Fig. 1 plots the observed $P_{e\text{c}}$ against $E_b/N_0$ for a systematic and a nonsystematic encoder that have the same distance profile, as well as a best-$d_{t_{\text{dec}}}$ QLI encoder, whose distance profile is approximately the same. All encoders have the same memory, which means that they all use the same resources. At the $M$'s in the figure [16, 32, 64, 128] the difference in the measured $P_{e\text{c}}$ is statistically insignificant across the three encoders. This occurs despite the fact that $d_{t_{\text{dec}}}$ varies widely among the three encoders. We have noticed similar behavior for smaller $M$'s in our earlier work [13].

A very different result occurs when encoders with widely varying early distance profiles are tested. Fig. 2 compares the observed $P_{e\text{c}}$ for the memory 12 best-$d_{t_{\text{dec}}}$ QLI encoder, decoded with $M = 32$, to $P_{e\text{c}}$ for the encoder with both generators reversed. This reversal creates an encoder with the same $d_{t_{\text{dec}}}$, but with a distance profile that develops late in the trellis rather than early. Despite the identical free distance, $P_{e\text{c}}$ for the reversed encoder is nearly tenfold worse.

Fig. 3 compares $P_{e\text{c}}$ tests at $M = 32$ for a sequence of five encoders, all with $d_{t_{\text{dec}}} = 10$, each of whose distance profile successively underbounds the others. The result is a sequence of widely varying $P_{e\text{c}}$ curves arranged in the same order.

Almost all the variation in decoder $P_{e\text{c}}$ for the encoders that we tested can be traced to variations in the early distance profile. It is reasonable that variation later in the distance profile might affect $P_{e\text{c}}$ for large $M$, but these $M$ are of less practical interest. The following hardly affect $P_{e\text{c}}$: The type of encoder (systematic or not, QLI or
we compare for various length, and consequently they have the same distance profile. In Fig. 4 experimental observations is needed for these two generator matrices are equivalent over the first memory or fewer channel errors. Furthermore, for this ODP nonsystematic convolutional generator matrix (octal notation) \( G_{\text{sys}} = (74041567512 54041567512) \) with \( d_{\text{free}} = 25 \). By long division of the generator polynomials and truncation after degree \( m \), we obtain the memory \( m = 31 \) ODP systematic convolutional generator matrix \( G_{\text{sys}} = (400000000000 67115143222) \) with \( d_{\text{free}} = 16 \). These two generator matrices are equivalent over the first memory length, and consequently they have the same distance profile. In Fig. 4 we compare for various \( M \) the error event probability \( P_{\text{ev}} \) observed at the root of the code tree for these two encoders. The outcomes of the two simulations are virtually identical at each \( M \).

III. DISTANCE PROFILE AND \( M \): COUNTING ESTIMATE

Because the error rates among different encoders with the same distance profile are so close, a more accurate measurement than experimental observations is needed for \( P_{\text{ev}} \). Also, experiments do not give accurate estimates at small \( P_{\text{ev}} \). We therefore estimate \( P_{\text{ev}} \) combinatorically by counting the number of channel error sequences that can lead to a decoding error in the first data symbol. Let the BSC crossover probability be \( p \), the decoder decision depth be \( t_{12} \) trellis levels, or \( n_{12} = c t_{12} \) bits, and suppose the encoder, \( M \), and \( t_{12} \) are sufficient to guarantee the correction of all combinations of \( e \) or fewer channel errors. Furthermore, for this \( M \)-algorithm decoder suppose that \( N(d) \) channel error sequences of Hamming weight \( d \) can potentially lead to decoder error. Then the error event probability is not), the free distance, the encoder memory. Depending on \( M \), the last two must exceed a minimum threshold.

As a final example, we compare a nonsystematic encoder to its equivalent systematic encoder. Consider the memory \( m = 31 \) QLI ODP nonsystematic convolutional generator matrix (octal notation) \( G_{\text{non-sys}} = (74041567512 54041567512) \) with \( d_{\text{free}} = 25 \). By long division of the generator polynomials and truncation after degree \( m \), we obtain the memory \( m = 31 \) ODP systematic convolutional generator matrix \( G_{\text{sys}} = (400000000000 67115143222) \) with \( d_{\text{free}} = 16 \). These two generator matrices are equivalent over the first memory length, and consequently they have the same distance profile. In Fig. 4 we compare for various \( M \) the error event probability \( P_{\text{ev}} \) observed at the root of the code tree for these two encoders. The outcomes of the two simulations are virtually identical at each \( M \).

\[
P_{\text{ev}} \leq \sum_{d=1}^{n_{12}} N(d) p^d (1-p)^{n_{12}-d}.
\]

The remaining task is to count the sequences that can lead to decoder error. This is done by a search of the decoder trellis with \( M \) paths retained, once for each candidate error sequence. As the search progresses, at each level there will be a certain number of trellis paths at or below the Hamming distance of the correct path. If this number never exceeds \( M \), the correct path cannot be dropped, it eventually is closer to the received path than any incorrect-subset path, and decoding of the first trellis branch is correct with certainty. If the number exceeds \( M \), the correct path can be worse than \( M \) paths at some trellis level, or it can be tied for worst among the \( M \). In either case, we count the error sequence in the total \( N(d) \), which makes (9) an overbound.

It can happen that the errors in a sequence concentrate later in the sequence, and because the decoder keeps only \( M \) paths, all incorrect subset paths are dropped before the errors are reached. In this case, correct decoding of the first symbol is certain; the reduced search has in fact rescued the decoder, so far as the first branch is concerned. If the candidate error sequences are tested in the right order, most can...
be skipped over for this reason and the counting becomes a relatively efficient procedure.

If the correct path fails to be unquestionably within the best paths, the most likely event is that it is one Hamming unit too heavy and that it is therefore tied with others as a candidate to be dropped. We assign a probability of $\frac{1}{2^7}$ to the event that the tie-break keeps the correct path; we thus apply a factor of $\frac{1}{2^7}$ to (9). Finally, the tail of (9) beyond, say, weight $119$, is easy to compute, if we assume a decoder error is certain. Beyond a certain $119$, this expression decays rapidly. In order to reduce the calculation of $10^7$, it can be substituted for the tail in (9) at the earliest convenient moment; the effect will make (9) a little more of an overbound.

We turn now to results obtained with (9). Fig. 5 plots the counting estimate of $P_{ev}$ for another collection of encoders with similar good distance profiles and relatively small $\delta$. This time, the encoders are the best-$d_{\text{free}}$ encoders of memories $49$, $54$, and $48/48$, in addition to an ODP feed-forward systematic encoder, denoted ODP-FFS. The $P_{ev}$ estimates are almost identical for the same $\delta$, and in fact the traces of the encoders resemble the intertwined strands of a rope. We claim this is because the early ranges of the distance profiles are very similar. The same progression occurs for these encoders in the counting estimate, as shown in Fig. 6, which treats $M = 4, 8, 16, 32$. Occasionally, the ordering of a single pair of encoders reverses at a particular $M$ (e.g., at $M = 4$, the order is $a b c$). Overall, however, the pattern is clear: A nested set of distance profiles leads to a nested set of $P_{ev}$ curves.

![Figure 5](image1.png)

**Fig. 5.** Counting estimate of the event error probabilities for a set of encoders with different memories but with similar early ranges of the distance profiles.

- BDF10 = $G_{41} = (46727542)$, $m = 10$, $d_{\text{free}} = 14$;
- BDF6 = $G_{52} = (557444)$, $m = 6$, $d_{\text{free}} = 10$;
- BDF4 = $G_{53} = (40722)$, $m = 4$, $d_{\text{free}} = 7$;
- ODP-FFS = $G_{54} = (40007153)$, $m = 11$, $d_{\text{free}} = 9$.

![Figure 6](image2.png)

**Fig. 6.** Counting estimate of the event error probabilities for the same encoders as in Fig. 5.

- $a = G_{51} = (46727542)$, $b = G_{52} = (557444)$, $c = G_{53} = (40722)$, $d = G_{54} = (40007153)$, $e = G_{55}$.

IV. COMPARISON OF THE $M$ AND VITERBI ALGORITHMS

The object of a reduced-search decoder is to reduce the survivor number—and hopefully, the computation—for the same error perfor-
We have shown that $P_{av}$ for a decoder of the $M$-algorithm type will depend almost entirely on the distance profile, but we have not shown how much better it might be than the Viterbi algorithm (VA) for a given number of survivors. Actually, the answer depends strongly on whether the error criterion is bounded distance or error rate.

We have already explored the BDD criterion for all rates in an earlier paper [15]. At rate 1/2, for example, convolutional codes that correct $e$ or fewer errors closely follow the rule $M \approx (1 + \sqrt{2})^e \approx (2.414)^e$, while the VA requires state size $\approx 4^e$; i.e., the disparity in survivors grows exponentially as $\approx (1.7)^e$.

If the decoder is to be judged by its $P_{av}$, the disparity turns out to be much less. At very large $E_b/N_0$, the comparison must track the BDD result, since the lightest weight uncorrected error sequence will dominate the $P_{av}$ calculation. At relatively high $P_{av}$, we show the test results in Fig. 7. Nonsystematic best-$d_{	ext{free}}$ encoders are compared; the $M$-algorithm decodes the $m = 9$ encoder at $M = 16, 32, 64, 128$, while the VA works with the three encoders of memory 7, 8, and 9, whose survivor numbers are 128, 256, and 512. Some study of the figure will show that the VA requires somewhat more than twice the survivors for the same $P_{av}$.

Fig. 8 aims at relatively low $P_{av}$ and is based on the counting estimate. The VA error event probability is computed from [3, p. 223]

$$P_{av,VA} \approx \sum_{d \geq d_{	ext{free}}} \alpha(d) [4p(1-p)]^{d/2}$$

(11)

in which $\alpha(d)$ is the number of paths of weight $d$ in the trellis. The figure works by comparing the VA with nonsystematic ODP encoders to an $M$-algorithm (MA) that retains 1/4 the survivors; the MA works always with the $m = 16$ ODP systematic encoder (400000, 671166).

For the combination (VA size 16)/(MA size 4), for which the VA uses an $m = 4$ ODP encoder, the VA is 0.5 dB better. For the combination (VA size 64)/(MA size 16), the VA gain drops to about 0.2 dB. By the time (VA size 256)/(MA size 64) is reached, the MA is actually a little better. Note that all the codes in the figure have nearly the same early distance profile.

As one would expect, we are seeing here that the VA/MA comparison begins to feel the influence of the bounded distance case as $P_{av}$ drops. We can predict that the VA will require 2–4 times the survivors that the $M$ algorithm does, depending on the decoder error rate. It is widely acknowledged that the MA executes 1.5–2 times the calculations of the VA per survivor kept, so that this rate of 2–4 needs to be reduced somewhat. We can claim that overall the MA has up to half the cost of the VA, when $P_{av}$ is the criterion.

V. THE PATH LOSS ISSUE

With the probability of error design criterion, more errors than are guaranteed corrected by the list size $M$ may occur sometimes, or even often. These errors may lead to loss of the correct path from
the decoder memory, an event that the decoder must deal with. Tests show that the overall bit-error rate of an \( M \)-algorithm decoder with a nonsystematic encoder is much worse than the same decoder with a systematic encoder having a similar distance profile. Yet the error event rate is almost identical. The reason is that once the error event occurs the rest of the frame almost always follows an incorrect trellis path. This is the correct-path loss problem. Tests show that with systematic encoders the \( M \) decoder quickly recovers a lost correct path, and can even outperform the Viterbi decoder that works with a matching nonsystematic encoder. An example is shown in Fig. 9, which compares the BER of systematic, nonsystematic, and QLI encoders at \( M = 16 \) and 128. All have approximately the same initial distance profile. The free distance of the systematic encoder is by far the least, yet its BER is more than ten times better.

Fig. 10, which is a companion figure to Fig. 7, compares the BER for some systematic and nonsystematic encoders with the same distance profiles. We then compared tests of the three types of encoders, all of memory \( m = 20 \).

\[
G_{91} = G_{11}, \quad G_{92} = G_{12}, \quad \text{and} \quad G_{93} = G_{13}.
\]

does it change the \( M \) needed to correct \( e \) errors, so long as \( e \) falls within the powers of the systematic encoder.

A suggestion of why systematic encoders offer rapid recovery of a lost correct path may be found by considering the trellises of rate \( R = 1/2 \) random systematic and nonsystematic encoders. Suppose the correct path is the all-zero one and no errors occur for a time, and consider an arbitrary trellis node. The 0-branch (the one that inserts a zero into the encoder shift register) is the one leading back to the correct path. For a systematic encoder, the distance increment of this “correct” branch is 0.5 on the average, while the incorrect branch has increment 1.5. For a nonsystematic encoder, these average increments are both 1, and give no particular direction of the search back to the correct path. Finally, we should use a feed-forward, rather than a feedback, systematic encoder, because a driving sequence of zeroes does not in general lead back to state 0 with the feedback encoders.

VI. CONCLUSION

We have given strong evidence that only feed-forward systematic encoders should be used with reduced-search decoders. To review the train of logic, we first showed that nonbacktracking decoder error performance depends almost entirely on the number of survivors kept and on the early part of the distance profile. We then compared tests of systematic and nonsystematic encoders with the same distance profile, and showed that because the systematic encoder allows quick recovery of a lost correct path, it has a much better overall error rate than a nonsystematic encoder has. Since both kinds of encoders have the same error rate in the absence of correct-path loss, systematic encoders are clearly superior to nonsystematic ones.

We have also reached other conclusions. Apparently, the survivor numbers required in a bounded-distance decoder also depend only
very weakly on free distance, encoder state-space size, and the precise digits of the code generators; instead, they are simply and directly related to the desired error correction.

While we have studied only the performance of the $M$ algorithm in this correspondence, the estimates of the maximum BDD storage size such as (5) are known to hold for any breadth-first decoder. In addition, our conclusions about distance profiles in Sections II and III are known to apply to backtracking decoders [22]. We can thus conjecture that our conclusions extend quite widely.

The conclusions here do run counter to much accepted wisdom about convolutional codes. For instance, much of the search for good codes has focused on maximizing the free distance for a given encoder memory. Yet we show that the free distance has almost no effect on the performance of a reduced decoder with a fixed $M$ survivors, so long as the distance exceeds a certain minimum. Second, our conclusions about distance profiles in Sections II and III are known to apply to backtracking decoders [22]. We can thus conjecture that our conclusions extend quite widely.

The conclusions here do run counter to much accepted wisdom about convolutional codes. For instance, much of the search for good codes has focused on maximizing the free distance for a given encoder memory. Yet we show that the free distance has almost no effect on the performance of a reduced decoder with a fixed $M$ survivors, so long as the distance exceeds a certain minimum. Second, our conclusions about distance profiles in Sections II and III are known to apply to backtracking decoders [22]. We can thus conjecture that our conclusions extend quite widely.

The conclusions here do run counter to much accepted wisdom about convolutional codes. For instance, much of the search for good codes has focused on maximizing the free distance for a given encoder memory. Yet we show that the free distance has almost no effect on the performance of a reduced decoder with a fixed $M$ survivors, so long as the distance exceeds a certain minimum. Second, our conclusions about distance profiles in Sections II and III are known to apply to backtracking decoders [22]. We can thus conjecture that our conclusions extend quite widely.

The conclusions here do run counter to much accepted wisdom about convolutional codes. For instance, much of the search for good codes has focused on maximizing the free distance for a given encoder memory. Yet we show that the free distance has almost no effect on the performance of a reduced decoder with a fixed $M$ survivors, so long as the distance exceeds a certain minimum. Second, our conclusions about distance profiles in Sections II and III are known to apply to backtracking decoders [22]. We can thus conjecture that our conclusions extend quite widely.