Application of Distance Spectrum Analysis to Turbo Code Performance Improvement

Mats Öberg* and Paul H. Siegel
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, California 92093-0407
E-mail: moberg@ucsd.edu, psiegel@ucsd.edu

Abstract

We present a simple method to improve the performance of turbo codes. Distance spectrum analysis is used to identify information bit positions affected by low-distance error events, which are few in number due to the sparseness of the spectrum. A modified encoder inserts dummy bits in these positions, resulting in a lower and steeper error floor in the bit-error-rate (BER) performance curve. For sufficiently large interleaver size, the only cost is a very slight reduction in the code rate. We illustrate the method using a rate 1/2 turbo code, with interleaver length $N = 10000$. The proposed method improves the BER by an order of magnitude at $E_b/N_0 = 2.5$ dB, with a rate loss of only 0.4%.

1 Introduction and Background

Turbo codes – parallel concatenated convolutional codes connected by an interleaver – were first introduced in 1993 by Berrou et. al [1], and are now widely recognized as a landmark development in error control coding. Two characteristic features of turbo code performance are the small bit-error-rate (BER) achieved even at very low signal-to-noise ratio (SNR) and the flattening of the error-rate curve – the so-called "error floor" – at moderate and high values of SNR.

Recently Perez et. al. [2] analyzed the codeword weight distribution and multiplicity – the distance spectrum – of turbo codes and offered an explanation for both of these phenomena. By considering input sequences of length equal to the interleaver size N, they derived a maximum-likelihood (ML) bound for the BER performance of turbo codes,

$$P_b \leq \sum_{d=d_{free}}^{2(\nu+N)} \frac{N_d \tilde{\omega}_d}{N} Q\left(\sqrt{d \frac{2RE_b}{N_0}}\right) \tag{1}$$

where N_d is the multiplicity of weight-d codewords, $\tilde{\omega}_d$ is the average weight of the information sequences causing weight-d codewords, R is the code rate, and ν is the

*This research is supported by the Royal Swedish Academy of Sciences and the Center for Wireless Communications at UCSD. The material in this paper was presented in part at the International Symposium on Turbo Codes & Related Topics, Brest, France, September 3-5, 1997.

memory of the constituent convolutional code. Although the iterative decoding procedure utilized in turbo codes is sub-optimal [3], its performance has been found to be very close to that of an optimal ML decoder. Thus the bound in (1) can be used to evaluate turbo code performance.

At moderate to high SNR, the BER bound is dominated by the free-distance term, and the performance approaches the *free-distance asymptote*, P_{free}, given by

$$P_{free} = \frac{N_{free}\tilde{\omega}_{free}}{N} Q\left(\sqrt{d_{free}\frac{2RE_b}{N_0}}\right) , \tag{2}$$

where d_{free} denotes the free distance of the code, with corresponding multiplicity N_{free} and average information weight $\tilde{\omega}_{free}$.

In [2], Perez, *et al.* studied a turbo code based upon a particular pseudo-random interleaver with size $N = 65536$. The code was found to have $d_{free} = 6$, $N_{free} = 3$, and $\tilde{w}_{free} = 2$. The simulated BER performance was shown to approach the free-distance asymptote, which had a shallow slope as a result of the relatively small free-distance of the turbo code. The observed error floor could therefore be interpreted as simply a manifestation of the relative flatness of the asymptote.

For this particular turbo code, the free distance codewords all have information weight 2, as well as a very small effective multiplicity,

$$\frac{N_{free}}{N} = \frac{3}{65536},$$

resulting in a small error coefficient in the expression (2) for the free-distance asymptote P_{free}. The small error coefficient accounts for the fact that, despite the small free-distance of this turbo code, the error floor appears at a rather low error rate ($BER \approx 10^{-6}$ at SNR=1 dB).

The analysis of "average" turbo codes in [2] shows that this is the typical situation and, moreover, the unusually small multiplicity extends to low-weight codewords in general. This "spectral sparseness" [2], which is not found in conventional convolutional codes, can be attributed to the parallel concatenated convolutional code structure of turbo codes and the properties of pseudo-random interleavers. A consequence of this spectral sparseness is that the free-distance asymptote dominates the error-rate bound even at very low SNR. This fact, along with the small effective multiplicity for free-distance codewords, accounts for the remarkable performance of turbo codes in the SNR region close to the Shannon limit.

Based upon this analysis of turbo code distance spectrum properties, two modifications to the turbo code design that would improve overall code performance were suggested in [2]. The error floor can be lowered, without a slope change, by increasing the interleaver size N, while maintaining the free distance and corresponding multiplicity. Alternatively, the slope of the error floor can be increased (negatively) by choosing constituent convolutional codes that increase the free distance of the turbo code, while keeping the corresponding multiplicity small.

In this paper, we will propose another approach to modifying the turbo code that does not require any change of the interleaver size or the constituent convolutional codes. By examining the interplay between the constituent convolutional codes and the pseudo-random interleaver, we derive a list of the non-zero bit positions in the information frames that produce minimum-weight turbo codewords. A modified encoder places dummy bits in these positions, and after the turbo decoding is completed, the contents of these

positions in the decoded frame are discarded. This process effectively removes the contribution to the BER of the free-distance error events, resulting in a lowering of the error floor and a change in its slope. By determining bit positions that correspond to other low-weight codewords, we can further improve code performance by applying this same procedure to those frame positions. The small multiplicity of low-weight codewords in the turbo code implies that the rate loss incurred by this encoder modification is negligible for large interleaver size.

The remainder of the paper is organized as follows. In Section 2, we discuss properties of the constituent convolutional encoders that play a role in the analysis of turbo code performance. In Section 3, we investigate the effect of the interleaver on the distance spectrum and, in particular, the set of low-weight codewords. Section 4 provides the details of the new method for improving the turbo code performance, as well as BER simulation results that confirm the expected gains. Section 5 summarizes our results.

2 Properties of the Constituent Encoders

A turbo encoder consists of two or more recursive systematic convolutional (RSC) encoders in parallel concatenation, along with interleavers which permute the input bits of the first encoder before applying them to the inputs of the other encoders. Input bits in a frame of length N are encoded by the first RSC, producing what we call the first dimension codeword. The same information frame is permuted by the interleaver and encoded by the second encoder, generating the second dimension sequence. A similar procedure is followed with any additional encoders. Since the constituent encoders are systematic, only the first information frame need be transmitted, along with the parity bits from each encoder. In this paper, we will consider only the case of two identical, rate 1/2 RSC encoders. To increase the overall rate of the encoder from 1/3 to 1/2, we follow the usual practice of puncturing every other parity bit in each dimension.

Berrou, *et al.* [4] proved a property of RSC encoders that plays an important role in the characterization of minimum-distance error events in turbo encoders. They showed that if $h(D)$ is the feedback polynomial of a RSC encoder, and if $h(D)$ is a divisor of $1 + D^L$, then $h(D)$ is also a divisor of $1 + D^{pL}$ for any integer $p \geq 1$. Let 0^i denote a run of 0's of length i. The result implies that an input stream $1\,0^{Lp-1}\,1$, where the two ones are separated by $pL - 1$ zeros, for any integer $p \geq 1$, will generate a trellis path that diverges from the all-zero path and then remerges after $pL + 1$ trellis steps. The encoder will leave the all-zero state in response to the first 1, and then will return to that state in response to the second 1.

Example 1 *Consider the memory-4, RSC encoder with parity-check polynomials* $h_0(D) = 1 + D + D^2 + D^3 + D^4$ *(octal 37) and* $h_1(D) = 1 + D^4$ *(octal 21) [2]. The feedback polynomial* $h_0(D)$ *divides* $1 + D^5$*. Therefore, any binary sequence* $1\,0^{5p-1}\,1$*,* $p \geq 1$*, will force the encoder to leave the all-zero state at the appearance of the first 1, and return to that state* $5p + 1$ *steps later, in response to the second 1. Fig. 1 illustrates the codeword corresponding to the case* $p = 1$ *. Note that the weight of the resulting codeword is 6.*

Example 2 *For an encoder with the same feedback polynomial* $h_0(D)$ *as in Example 1, any binary sequence* $h_0(D)^r$*,* $r \geq 1$*, will force the encoder to leave the all-zero state at the first 1, and return at the final 1. When* $r = 1$*, one gets the length-5 sequence* $1\,1\,1\,1\,1$*, which generates a weight-7 codeword.*

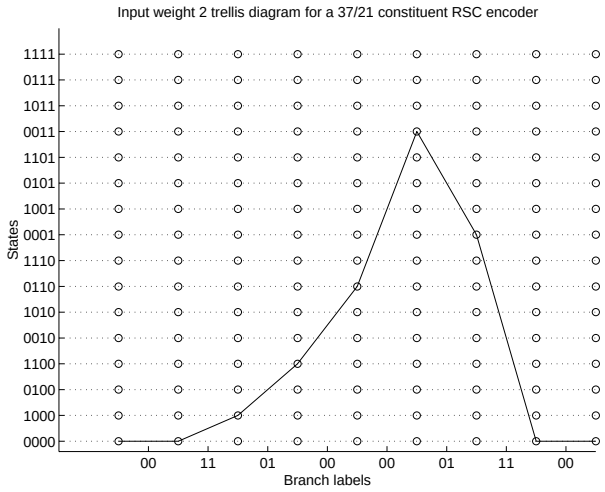

Figure 1: Response of a octal (37,21) RSC encoder with input pattern 0100001000.

The linearity of RSC encoders permits the interpretation of these observations in terms of error events. In the case of the feedback polynomial in the preceding examples, any pair of code sequences $a(D)$ and $b(D)$ that differ from another by $e(D) = 1 + D^{5p}$, $p \geq 1$, or $e(D) = h_0(D)^r$, $r \geq 1$, will correspond to trellis paths whose state sequences are distinct during the course of an error event of length equal to the degree of $e(D)$.

For a RSC encoder, one might expect that low-weight code sequences would be generated in response to low-weight input sequences that correspond to short error events in the trellis. In particular, the weight-2 input sequences discussed in Example 1, in which the non-zero bits are separated by a small multiple of the code constraint length $L = 5$, are good candidates for producing minimum-weight code sequences. For the rate 1/2, octal (37,21) code, the free distance is $d_{free} = 6$, and Fig. 1 depicts a minimum distance code sequence that is generated in response to a weight-2 input sequence in which the non-zero bit separation is one constraint length.

As discussed in [2], for the constituent convolutional code, the multiplicity of free-distance error events, N_{free}, will be roughly proportional to the frame length N, since there are few restrictions on the frame positions where the event may begin. In the next section, we will see that the introduction of an interleaver in a parallel concatenated RSC coding structure will dramatically restrict the possible starting positions of free-distance error events.

3 The Effect of the Interleaver

The remarkable power of turbo codes comes from the combination of the parallel concatenated RSC codes and the permutation of the information frame by the interleaver. The interleaver permutes the frame of information bits in the first dimension prior to their encoding by the encoder in the second dimension. Loosely speaking, the effect of this permutation is to ensure that low-weight error events occur in only one dimension. Depending on the choice of the permutation, the interleaver can affect both the distances and the multiplicities of error events.

Low-weight turbo codewords arise when the interleaver maps low-weight information frames that produce low-weight parity in the first dimension into frames that produce low-weight parity in the second dimension. In view of the discussion in the previous

section, therefore, the interleaver should avoid certain permutations of bit positions as much as possible. Specifically, define the polynomial $b(D)$ by

$$b(D) = \sum_{n=0}^{p\nu} b_n D^n = h_0(D)^p, \tag{3}$$

and let $\mathcal{M}_p$ denote the indices of non-zero coefficients of $b(D)$,

$$\mathcal{M}_p = \{n | b_n \neq 0\}. \tag{4}$$

Then, the interleaver should avoid the following mappings of subsets of bit positions:

$$\{i, i+Lp\} \longmapsto \{j, j+Lr\} \tag{5}$$

$$\{i, i+Lp, j, j+Lr\} \longmapsto \{k, k+Lu, m, m+Lv\} \tag{6}$$

$$\{i+\mathcal{M}_p\} \longmapsto \{j+\mathcal{M}_r\} \tag{7}$$

where L is the parameter in the divisibility condition of the previous section; i, j, k and m are positive integers $\leq N$; ν is the memory length of the constituent RSC encoder; and r, s, u and v are integers small enough that the corresponding information sequences produce low parity weight at the encoder output. Note that, in the mappings in (7) above, we require that $|\mathcal{M}_p| = |\mathcal{M}_r|$.

Remark: Permutations that map unions of the various subsets above to other unions of these subsets having equal information weight should also be avoided.

Analysis of random interleavers shows that an "average interleaver" can accomplish these desired objectives fairly well [5], and pseudo-randomly generated interleavers have been found to be generally superior to other structured interleavers. The superior performance does not typically arise from a large free distance, however. Rather, the best performing interleavers appear to gain their advantage through drastic reduction of the low-distance error event multiplicity. This "spectral thinning" is the property demonstrated through example and analysis in [2], and lies at the heart of the technique for improving turbo code performance as described in the next section.

We analyzed the distance spectrum of a turbo code based upon the aforementioned rate 1/2 constituent code and a particular pseudo-random interleaver of size $N = 10000$. (In the analysis, we assume that, at the end of the information frame, the encoders in both dimensions are driven to the all-zero state by appending appropriate tails of length ν.) The free distance of the turbo code is $d_{free} = 6$, with free-distance multiplicity $N_{free} = 4$. As was the case for the $N = 65536$ turbo code discussed in [2], all of the free-distance codewords correspond to weight-2 information sequences. We also determined the codewords of weight 16 or less that are generated by weight-2 inputs where the non-zero information bit positions in both dimensions are separated by a small multiple of the constituent code constraint length. In our search for mappings of bit patterns corresponding to powers of the feedback polynomial that mapped into a bit pattern corresponding to a power of the feedback polynomial, we found none which would result in codeword weight 16 or less.

Table 1 describes in more detail the search results for codewords of weight 10 or less. The table specifies the positions of the non-zero bits in information frames of weight 2 that generate code sequences with weight 6, 8, or 10. For each bit pair, the bit separation in the first dimension and the bit separation in the second dimension after permutation by the interleaver are indicated in the column labeled "Mapping". The bit separation is

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.