6.962 Graduate Seminar in Communications

Turbo-Like Codes: Structure, Design Criteria, and Variations

Presenter: J. Nicholas Laneman

October 18, 2000

1 Summary

Broadening our treatment of capacity-approaching codes, we explore turbo-like codes and related topics. In contrast to our primarily graph-based views of low-density parity check (LDPC) and repeat-accumulate (RA) codes, we examine turbo codes in their traditional parallel and serial concatenated structures, summarize the concept of extrinsic information exchanged in the iterative decoding algorithm, and review several ways in which turbo-like codes can be analyzed and designed, in particular, an appealing class of new techniques based upon extrinsic information transfer (EXIT) charts for characterizing and visualizing iterative decoding dynamics in low SNR regimes. Finally, we mention some interesting variations on the turbo-decoding theme.

1.1 Code Structure

Turbo codes originally appeared as parallel concatenations of two convolutional codes [1]. Turbo-like serial concatenations also arose [2], as well as generalizations in a number of directions, *e.g.*, concatenations of multiple block and/or convolutional codes. In contrast to conventional concatenated coding schemes [3], turbo-like codes employ relatively simple codes, *e.g.*, $4-16$ states, very long interleavers, *e.g.*, 10^4-10^5 bits, between the codes, and iterative decoding between the codes.

1.1.1 Parallel Concatenation

Fig. 1(a) depicts a block diagram for parallel concatenation of interleaved codes, *e.g.*, turbo codes. One code encodes the input bit sequence $\mathbf{u}$ to produce code sequence $\mathbf{c}_1$, while the other code encodes the permuted bit-sequence $\pi(\mathbf{u})$ to produce code sequence $\mathbf{c}_2$. The two code sequences $\mathbf{c}_1$

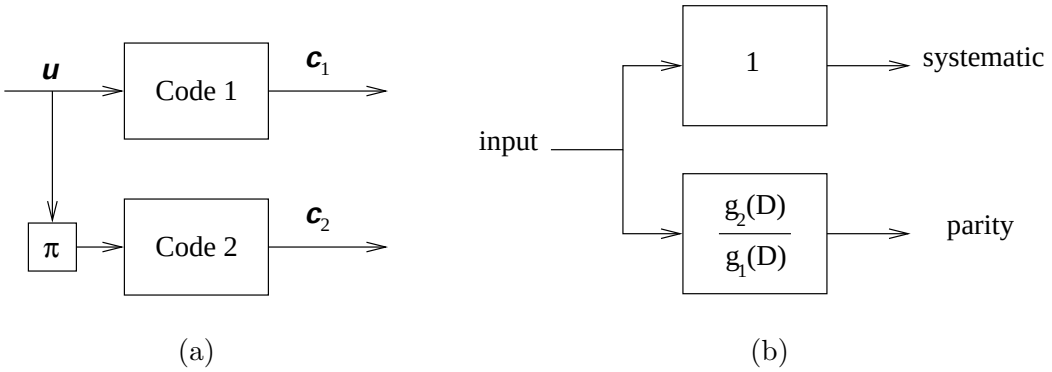

Figure 1: Parallel concatenation of interleaved codes: (a) generate structure, (b) component rate $R = 1/2$ recursive systematic convolutional code.

and $\mathbf{c}_2$ are multiplexed onto the channel, and potentially punctured to produce overall codes of higher rate.

The original turbo code employs identical, $R = 1/2$ recursive systematic convolutional codes of the form shown in Fig. 1(b). These component codes have two generator polynomials. The identity generator polynomial 1 corresponds to the "systematic" part of the code, so that the input bit sequence appears unmodified as part of the code sequence. An infinite impulse response (IIR) generator polynomial $g_2(D)/g_1(D)$, where $g_1(D)$ and $g_2(D)$ are polynomials of finite degree in the standard delay variable D, corresponds to the "recursive" part of the code, so that the output is fed back through the polynomial $g_1(D)$. Since both component codes are systematic, the bits of $\mathbf{c}_2$ corresponding to $\pi(\mathbf{u})$ can be punctured, *i.e.*, not transmitted over the channel, to obtain an overall rate $R = 1/3$ code. Further puncturing of the parity bits from $\mathbf{c}_1$ and $\mathbf{c}_2$ result in higher coding rates.

It's natural to wonder why the code structure depicted by Fig. 1 might be appealing. We will examine this question more formally in Section 1.3. Indeed, the set of output sequences from the component code in Fig. 1(b) is equivalent to the set of output sequences from a non-recursive component code with generator polynomials $g_1(D)$ and $g_2(D)$; however, he mappings between input and output sequences in the two cases are quite different. Qualitatively, the net effect of interleaving and the recursive form of the component codes ensures that very few codewords have low weight. This property is important because, since the overall code is linear, the weight of each codeword can be viewed as the weight of an error event with respect to the all-zero codeword. For

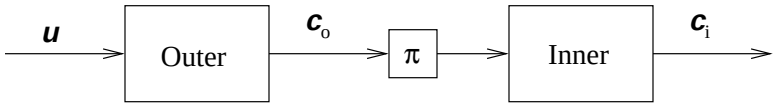

Figure 2: Serial concatenation of interleaved codes.

example, only certain input sequences (multiples of $g_1(D)$) will generate low-weight outputs from the first code. Ideally, the interleaver breaks up the structure of these special inputs so that the corresponding output from the second code is of high weight (with high probability).

1.1.2 Serial Concatenation

Fig. 2 depicts a block diagram for serial concatenation of interleaved codes. The outer code encodes the input bit sequence $\mathbf{u}$ to produce code sequence $\mathbf{c}_o$, while the inner code encodes the permuted code sequence $\pi(\mathbf{c}_o)$ to produce code sequence $\mathbf{c}_i$. For serial concatenations employing convolutional codes, the inner code is typically recursive for essentially the same reasons as for parallel concatenations. Provided the permuted outer codeword $\pi(\mathbf{c}_o)$ is not one of the special inputs to the recursive inner code that produces low-weight outputs, the inner codeword $\mathbf{c}_i$ will have high-weight.

1.2 Iterative Soft Decoding

As observed in our previous sessions, iterative soft decoding of the constituent codes in a turbo-like code using the maximum *a posteriori* (MAP) decoding algorithm [4] corresponds to a particular schedule of the sum-product algorithm in the associated factor graph for the code [5]. Specifically, iterations consist of passing messages in a forward/backward cycle along one code trellis, then between trellises through the interleaver, and finally in a forward/backward cycle along the other code trellis. Typically, $10 - 20$ iterations are performed.

Beyond reviewing this observation in a representative example factor graph, we do not discuss the turbo-decoding algorithm in detail. However, the notion of "extrinsic information" has become so rooted in the turbo code literature that we emphasize it both for clarification purposes and for later use in our discussion. In the factor graph interpretation, message passing between trellises through the interleaver corresponds to the transfer of extrinsic information in turbo decoding.

Extrinsic information is perhaps most straightforward to interpret in the context of parallel

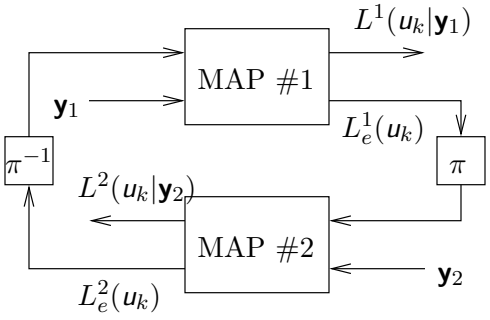

Figure 3: Block diagram for iterative decoding of parallel concatenated codes.

concatenation. First, consider a component code individually. Suppose the coded sequences $\mathbf{c}$ is transmitted over a memoryless channel, and received sequence $\mathbf{y}$ is observed. The MAP algorithm computes

$$L(u_k|\mathbf{y}) = \ln \frac{\Pr[u_k = +1|\mathbf{y}]}{\Pr[u_k = -1|\mathbf{y}]} \ . \tag{1}$$

For a memoryless channel, in particular for the BSC and AWGN channel, using properties of systematic codes, we can manipulate (1) into the form [6]

$$L(u_k|\mathbf{y}) = \underbrace{L_c\, y_k}_{\text{channel likelihood}} + \underbrace{L_a(u_k)}_{\textit{a priori}\text{ likelihood}} + \underbrace{L_e(u_k)}_{\text{extrinsic likelihood}} \ , \tag{2}$$

where L_c is a constant capturing, *e.g.*, the crossover probability of a BSC and the SNR of an AWGN channel. It is apparent from (2) that the extrinsic information captures everything we learn about u_k from the code structure and other channel observations, *i.e.*, the likelihood or information "extrinsic" to both our *a priori* knowledge about u_k and the knowledge we obtain from the channel observation y_k.

Now consider the block diagram in Fig. 3 for iterative decoding of the code in Fig. 1(a). Let the channel outputs corresponding to (punctured) transmitted code sequences $\mathbf{c}_1$ and $\mathbf{c}_2$ be $\mathbf{y}_1$ and $\mathbf{y}_2$, respectively. For the first iteration using the first code, we assume that the u_k are equally-likely, so that $L_a(u_k) = 0$. When we compute (2) for the first code and its corresponding channel outputs,

we can compute the extrinsic information

$$L_e^1(u_k) = \underbrace{L^1(u_k|\mathbf{y}_1)}_{\text{MAP alg. \#1}} - \underbrace{L_c\, y_{1,k}}_{\text{channel obs.}} \tag{3}$$

and use (3), after suitable permutation by $\pi(\cdot)$, as the *a priori* likelihood when we apply the MAP algorithm for the second code and its corresponding channel outputs. The extrinsic information obtained from this decoding becomes

$$L_e^2(u_k) = \underbrace{L^2(u_k|\mathbf{y}_2)}_{\text{MAP alg. \#2}} - \underbrace{L_c\, y_{2,k}}_{\text{from channel obs.}} - \underbrace{\pi(L_e^1(u_{k'}))_k}_{\text{MAP alg. \#1}} . \tag{4}$$

Continuing with the second iteration, the extrinsic likelihood from the first MAP decoder uses (4) and becomes

$$L_e^1(u_k) = \underbrace{L^1(u_k|\mathbf{y}_1)}_{\text{MAP alg. \#1}} - \underbrace{L_c\, y_{1,k}}_{\text{from channel obs.}} - \underbrace{\pi^{-1}(L_e^2(u_{k'}))_k}_{\text{MAP alg. \#2}} . \tag{5}$$

Final decoding is performed by setting

$$\hat{u}_k = \begin{cases} +1, & L^1(u_k|y_1) \geq 0 \\ -1, & L^1(u_k|y_1) < 0 \end{cases} , \tag{6}$$

or, similarly, by using the output of the second MAP decoding algorithm.

1.3 Design Criteria

Since the rather surprising discovery of turbo codes, many researches have attempted to characterized their performance, by means of analysis and simulations, and thereby design more efficient coding schemes. We discuss two such approaches. As background material, we quickly summarize the insights gained from considering (effectively non-realizable) maximum-likelihood (ML) decoding of the codes and a union-bound error analysis. As we might expect, this analysis is suitable only for moderate to high SNR, but nevertheless offers some perspective on the appropriate structures for the codes. We will then focus our attention on interesting new characterizations of the iterative decoding process in terms of extrinsic information transfer (EXIT) charts. These EXIT characterizations seem particularly appealing for low SNR regimes near the Shannon limit.

1.3.1 ML Decoding

By examining the ML decoding performance of turbo-like codes, we can essentially compare the inherent capabilities of the codes themselves to other classes of codes, such as, *e.g.*, single block and convolutional codes. Such an analysis might indicate key properties that make turbo-like codes particularly effective, and might suggest design rules for improving the codes.

By computing weight distributions on the codewords in the overall codes as a function of the interleaver size N and using union bounds to the bit-error probability on additive white Gaussian noise (AWGN) channels, [7] and [2] consider the performance of parallel concatenations and serial concatenations, respectively. For fixed constituent codes and increasing (random) interleaver length N, the authors make the following observations:

- Parallel Concatenation

    1. The number of codewords with minimum weight decreases in the length of the interleaver roughly as $1/N$ (the so-called *interleaver gain*).
    2. Asymptotically (both small bit-error probability and large SNR) the performance is dominated by the minimum distance error event (the so-called *error floor*).
    3. The recursive structure of the constituent convolutional codes highlighted in Section 1.1.1 is required for good turbo code performance, in the sense that increasing the length of the interleaver has little effect on performance for non-recursive codes.
    4. To first order, the key parameter to maximize for the component codes is the effective free distance.

- Serial Concatenation

    1. The use of an inner recursive convolutional encoder always yields an interleaver gain.
    2. To first order, the key parameter to maximize for the inner code is the effective free distance.
    3. If the outer code has free distance d_f^o, the interleaver gain is roughly $N^{-d_f^o/2}$ for d_f^o even and $N^{-(d_f^o+1)/2}$ for d_f^o odd. Thus an outer code with large and, if consistent with rate constraints, odd, free distance should be employed.
    4. Non-recursive convolutional codes are convenient as outer codes, because they have few input sequences generating free-distance error events, and the corresponding input

weights (number of bit-errors) are small.

Comparisons between simulated performance of iterative decoding for parallel and serial concatenations indicates smaller gaps to capacity for serial concatenated codes at small bit-error probabilities, *e.g.*, $< 10^{-5}$, and smaller gaps to capacity for parallel concatenated codes at moderate bit-error probabilities, *e.g.*, $> 10^{-5}$. The low bit-error probability result appears to stem from the higher interleaver gain (lower error floor) of the serial concatenation. The moderate bit-error probability result is not clearly explained in [2].

1.3.2 Iterative Decoding

The design rules for turbo-like codes obtained based upon observations from a union bound analysis of ML decoding are supported by simulated performance curves for iterative decoding. It might seem more natural to design the codes based on analysis of the iterative decoding algorithm itself, if such an analysis is tractable. Similar to one-dimensional approximation methods, *e.g.*, Gaussian approximations [8], used to design LDPC codes, several researchers are developing ways of viewing the dynamics of iterative decoding of turbo-like codes.

For the purposes of our discussion, we will focus on the extrinsic information transfer (EXIT) charts introduced and utilized for designing turbo-like codes in [9, 10, 11]. We note for further study the work of [12] and [13], both of which, by contrast, use SNR measures.

We can build upon our development of extrinsic information in Section 1.2. Fig. 3 helps visualize the basic idea of ten Brink's work. The coupled MAP decoders transfer extrinsic information in a feedback loop, each using the other's suitably permuted extrinsic information as *a priori* information. From a systems perspective, it seems natural to consider the transfer functions

$$L_e^2(u_k) = T_1(L_e^1(u_k), L_c\, y_{1,k}), \qquad L_e^1(u_k) = T_2(L_e^2(u_k), L_c\, y_{2,k}), \tag{7}$$

i.e., the transfer functions of input extrinsic information (from the other decoder), as well as channel input, to the decoder output extrinsic information. To consider transfer functions of this form, ten Brink examines the transfer of average mutual information $I(u; L_e^i(u))$ between the uncoded bits and the extrinsic informations.[1] He assumes that the extrinsic informations can be well-modeled

[1] To translate our notation into ten Brink's, set *a posteriori* probability $L^i(u_k|\mathbf{y}_i) = D_i$, channel likelihood $L_c\, y_k = Z_i$, *a priori* information $L_a^i(u_k) = A_i$, and extrinsic information $L_e^i(u_k) = E_i$.

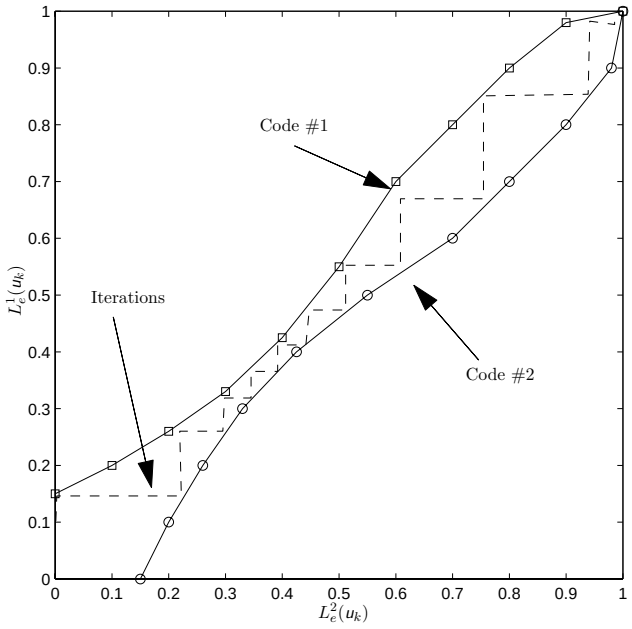

Figure 4: Representative extrinsic information transfer (EXIT) chart for symmetric parallel concatenated turbo codes. Data read from [9].

by independent Gaussian sequences, and computes empirical average mutual information transfer functions for the MAP decoding algorithms of various codes under varying channel SNR conditions.

By overlaying the transfer functions on the same graph as in Fig. 4, we arrive at the EXIT chart. As Fig. 4 indicates, the two transfer functions for the codes contain a gap through which the decoding iterations zig-zag, beginning at $L_e^2(u_k) = 0$ for equally-likely inputs to the first decoder. As long as there is a gap between the curves, the iterations progress upward through the gap and converge. The particular shape of the curve depends upon the structure of the component codes, and the codes can be selected according to how well their curves match. Generally speaking, the gap narrows with decreasing channel SNR, to a point (the "turbo cliff", as ten Brink calls it) at which the two curves intersect, rendering further iterations become ineffective.

Although the empirical results are based upon a Gaussian approximation for the extrinsic informations, simulated paths of the iterations tend to follow the envelope defined by ten Brink's curves. With this relatively simple chart, we are able to visualize some important characteristics of the codes in an intuitive way. For example, the narrower the gap, the slower the iterations progress, so that more iterations are required for convergence. Simulated performance results for turbo codes have demonstrated this trait since the appearance of [1], namely, more iterations required to reach

lower target error probabilities, and especially at low channel SNR. Moreover, we obtain these insights and can design more effective codes in the low SNR regime with (presumably) significantly less computation than is required for extensive bit-error rate simulations. Tools that shorten the design loop, even if approximate, can be very valuable for building insight, and ultimately much better codes.

As a final example of the usefulness of this technique, we should mention that ten Brink has developed [10] a serial concatenated turbo-like code consisting of a rate $R = 1/2$ outer repetition code followed by an interleaver and a particular inner rate $R = 1$ recursive convolutional code that operates within 0.1 dB of the Shannon limit. The key observation for this design appears to have come from examining the EXIT chart of several rate $R = 1$ recursive convolutional codes that have a wide gap with respect to the repetition code (straight line from the point $(0, 0)$ to $(1, 1)$). Unfortunately, such recursive codes have transfer function near zero for small input extrinsic information, so ten Brink "dopes" the code by treating it as a rate $R = 1/2$ systematic recursive convolutional code and puncturing between the systematic and parity bits in appropriate proportions to improve the EXIT characteristic near zero. More details of the EXIT charts for and design of serially concatenated codes can be found in [10, 11].

1.4 Variations

If time permits, we will briefly mention several variations on the turbo-decoding theme in which several receiver operations are jointly performed in an iterative fashion. These include: "turbo equalization" [14], in which the inner code of Fig. 2 represents an inter-symbol interference (ISI) channel; "turbo DPSK" [15], in which the inner code of Fig. 2 is a differential phase-shift-keying (PSK) modulator; as well as several joint source-channel and multiple access examples mentioned in [16].

References

[1] Claude Berrou and Alain Galvieux, "Near optimum error correcting codes and decoding: Turbo-codes," *IEEE Trans. on Communications*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.

[2] Segio Benedetto, Dariush Divsalar, Guido Montorsi, and Fabrizio Pollora, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans.*

Inform. Theory, vol. 44, no. 3, pp. 909–926, May 1998.

[3] G. David Forney, Jr., *Concatenated Codes*, M.I.T. Press, Cambridge, MA, 1966.

[4] Lalit. R. Bahl, John Cocke, Frederick Jelinek, and Josef Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[5] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger, "Factor graphs and the sum-product algorithm," Submitted to *IEEE Trans. Inform. Theory*, July 1998.

[6] Joachim Hagenauer, Elke Offer, and Lutz Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[7] Sergio Benedetto and Guido Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 409–428, Mar. 1996.

[8] Sae-Young Chung, Thomas J. Richardson, and Rüdiger Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," Submitted to *IEEE Trans. Inform. Theory*, Sept. 2000.

[9] Stephan ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," Submitted to *IEEE Trans. on Communications*, Mar. 2000.

[10] Stephan ten Brink, "A rate one-half code for approaching the Shannon limit by 0.1 dB," *IEEE Elec. Letters*, vol. 36, no. 15, pp. 1293–1294, July 2000.

[11] Stephan ten Brink, "Design of serially concatenated codes based on iterative decoding convergence," in *Proc. Int. Symp. on Turbo Codes & Related Topics*, Brest, France, Sept. 2000, pp. 319–322.

[12] Hesham El Gamal and A. Roger Hammons, Jr., "Analyzing the turbo decoder using the Gaussian approximation," in *Proc. Int. Symp. Inform. Theory*, Sorrento, Italy, June 2000, p. 319.

[13] Darisuh Divsalar, Sam Dolinar, and Fabrizio Pollara, "Low complexity turbo-like codes," in *Proc. Int. Symp. on Turbo Codes & Related Topics*, Brest, France, Sept. 2000, pp. 73–80.

[14] Michael Tüchler, Ralf Kötter, and Andrew Singer, "Turbo equalization: Principles and new results," Submitted to *IEEE Trans. on Communications*, Aug. 2000.

[15] Peter Hoeher and John Lodge, "Turbo DPSK: Iterative differential PSK demodulation and channel decoding," *IEEE Trans. on Communications*, vol. 47, no. 6, pp. 837–843, June 1999.

[16] Joachim Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proc. Int. Symp. Turbo Codes & Related Topics*, Brest, France, Sept. 1997, pp. 1–11.