TURBO CODES WITH SHORT MEMORY COMPONENT CODES

SUN Jianhua, WANG Qi, Wai Ho MOW and LI Kwok Hung
Division of Communication Engineering
School of EEE, Block S1
Nanyang Technological University
Singapore 639798

Abstract

In spite of the impressive performance of Turbo coding scheme, the iterative decoder is still too complicated to implement for many real-world applications. In this work, the use of shorter memory component codes but larger number of decoding iterations is proposed to improve the performance-complexity tradeoff of the classical Turbo code. Our simulation results demonstrate that for some applications, this approach can actually improve the performance of a Turbo coding scheme with a lower decoding complexity.

I. Motivation of this Work

Turbo coding scheme was proposed by Berrou *et al.* in 1993 [1]. Its near-channel-capacity error-correcting power has astonished the whole coding community. In particular, it has now become a classical result that the rate-½ Turbo code of block size about 64K bits presented in [1] only requires a signal-to-noise ratio (SNR) of 0.7dB to achieve a bit error rate (BER) of 10^{-5} using 18 iterations of iterative decoding.

The classical Turbo code is a two-component concatenated code. Though the parallel concatenation structure of Turbo code and the use of recursive systematic convolutional codes as its component codes are rather unconventional, the key to the great success of Turbo coding scheme lies on the use of a form of iterative decoding, which was originally proposed by Gallager [5] for decoding his low-density parity-check codes. As a matter of fact, recent results have shown that channel capacity can also be approached by a rather straightforward application of iterative decoding to some very traditional codes such as low-density parity-check codes [6] and the traditional serially concatenated codes [7], as long as the block size is large enough. The significance of iterative decoding is that a near-optimal decoding performance can be achieved with a much lower complexity than all optimal decoding methods known so far. It is this low-complexity near-optimal decoding method that allow the otherwise impossible simulation study of the near-capacity performance of some old and new codes.

Though the iterative decoder is of low complexity as a near-optimal decoder, its implementation complexity is still considered to be too high for many real-world applications. From the application point of view, a simple-to-implement suboptimal decoder could be far more attractive than its complicated near-optimal alternative. In order to enable a wide class of future communication

systems to be benefited by the impressive performance of Turbo code, it is very important to improve the complexity-performance tradeoff of the iterative decoder. Two main approaches have been used in previous works to reduce the decoding complexity. They are: i) proposals of various simplified suboptimal versions of the component soft-input soft-output decoder; and ii) use of certain criteria to assess the convergence of and stop accordingly the iterative decoding process.

In the classical Turbo code [1], two rate-1/2 component convolutional codes of memory length 4 were used. Robertson [2, Sec. 7] showed that little performance degradation was resulted when one of the component convolutional code is replaced by one with memory length 2 for interleaver size 10,000 and 18 iterations. More recently, Hoeher [3, Fig. 5] demonstrated that even if the two component convolutional codes both have memory length 2 only, near-channel-capacity performance can be achieved for sufficiently large interleaver. The aforementioned results in [2] and [3] have motivated us to investigate the performance degradation of Turbo coding schemes associated with the use of component codes with various memory lengths. Specifically, we would like to consider the question: *"For the same level of decoding complexity, can the performance of a Turbo coding scheme be improved by shortening the memory lengths of its component convolutional codes?"*

The rest of this paper is organised as follows. Section II introduces the measure of decoding complexity to be adopted here. Section III compares the simulated performance of Turbo coding schemes with component codes of various memory lengths.

II. Complexity Measure

To enable the complexity of different Turbo coding schemes to be compared, we need to define a measure of decoding complexity. Generally speaking, the complexity of a decoder critically depends on its implementation details and the associated technology, and there is no single complexity measure that suits all applications. Even a particular implementation platform and an accurate measure of complexity is specified, the determination of such a measure for given decoding algorithms is usually too complicated to be useful. In order to gain some insights into the complexity-performance tradeoff for Turbo coding schemes with component codes of different memory lengths, a simple but meaningful measure is adopted here.

In our implementation of the component soft-input soft-output decoders, the forward-backward (or BCJR or MAP) algorithm with linear approximation of log likelihood ratios as in [4] is used. For software implementation on a single digital signal processor, the number of decoding operations of each use of the *i*th component decoder is proportional to 2^{M_i}, where M_i is the memory length of the *i*th component code. Further, the number of uses of each component decoder is equal to the number of decoding iterations denoted by *c*. Hence, a reasonable definition of the complexity factor is

$$C[M_1, M_{2,}, \#c] = (2^{M_1} + 2^{M_2}) \times c .$$

III. Comparison of Simulated Performance

We consider the Turbo coding scheme in [1]. The nonuniform interleaver of size 256 by 256 therein is used. The component codes are two rate-½ convolutional codes, and puncturing is used to obtain the resultant rate-½ Turbo code. Two recursive systematic convolutional codes of memory lengths two and four are used as the component codes. The octal representation of their generator pairs are (7,5) and (37,21) respectively. All possible pairings of the two codes give rise to three Turbo coding schemes, which can be conveniently denoted by [2,2], [4,2], and [4,4], i.e. the pair of memory lengths of the two component codes. The BER performance of these three schemes are simulated for various number of decoding iterations. The BER performance curves of 1, 3 and 6 iterations are shown in Figures 1, while those of 12 and 18 iterations are in Figure 2. Each BER curve is labelled in a way consistent with the arguments of the complexity factor defined in Section II. For example, [4,4,#18] in Figure 2 refers to the performance of the classical Turbo code obtained after 18 iterations, as mentioned in Section I. It should be noted that in Figures 1 and 2 the BER results below 10^{-4} are for reference purpose only and need not be very reliable (only 100 bit errors are accumulated).

From the graphs, it is interesting to see that at BER 10^{-2}~10^{-4}, the relative performance of the three schemes for any fixed number of iterations remains almost unchanged. The SNR performance degrades about 0.2dB whenever a memory 4 component code is replaced by a memory 2 one. Namely, denoting by *c* the number of iterations,

$$C[4,4,\#c] = C[4,2,\#c]\text{-}0.2\text{dB}$$
$$C[4,2,\#c] = C[2,2,\#c]\text{-}0.2\text{dB}$$

Based on this observation, the choice of component codes can be made independent of the required number of decoding iterations.

For the same number *c* of iterations, the ratio of complexity factors for the three schemes are

$$C[4,4,\#c]:C[4,2,\#c]:C[2,2,\#c] = 8:5:2$$

Therefore, by replacing every memory 4 component code by a memory 2 one, the number of iterations can almost be doubled without a significant increase in the complexity. Since the SNR loss caused by shortening by 2 the memory length of a component code is almost fixed at about 0.2dB, an overall performance improvement associated with the doubled number of iterations should be obtained. As an example, from the graphs, while $C[2,2,\#12]=C[4,4,\#3]$, their SNRs are 1.13dB and 1.55dB respectively. Thus, scheme [2,2] gives a SNR gain of 0.42dB over scheme [4,4]. This SNR improvement is large especially if the number of iterations of the more complicated scheme is small. For example, while $C[2,2,\#3]=24$ is less than $C[4,4,\#1]=32$, their SNRs are 1.95dB and 3.8dB respectively. Thus scheme [2,2] is not only simpler than scheme [4,4], but offers a SNR gain of 1.85dB!

IV. Concluding Remarks

The simulated performance of three Turbo coding schemes with component

codes of memory lengths both two, both four, and one two one four have been presented in Figures 1 and 2. It has been found that at BER 10^{-2}~10^{-4}, the Turbo coding scheme with shorter memory component codes can give better performance with a simpler decoding complexity. The complexity measure reflects the number of computational operations, and is especially suitable for DSP implementation. However, as can be seen in Figure 2, the error floor for the scheme with shorter memory component codes seem to occur at higher BER. Therefore, for low BER applications, it is important to ensure that the memory of component codes is long enough to avoid the effect of error floor.

As mentioned in Section I, use of short memory component codes is only one way to improve the complexity-performance tradeoff of a Turbo coding scheme. Further improvements are anticipated by combining this approach with other known approaches such as the use of suboptimal soft-input soft-output component decoders and stop criteria.

References

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo-codes," in *Proc. ICC'93*, Geneve, Switzerland, May 1993, pp. 1064–1071. Journal version appeared in *IEEE Trans. Commun*., pp. 1261-1271, Oct. 1996.

[2] P. Robertson, "Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes," in *Proc. GLOBECOM'94*, San Francisco, California, pp. 1298-1303, Dec. 1994.

[3] P. Hoeher, "New iterative ("Turbo") decoding algorithms," in *Proc. TURBO'97*, Brest, France, 3rd-5th September 1997.

[4] S. Benedetto and G. Monorsi, "Soft-Output Decoding Algorithms in Iterative Decoding of Turbo Codes," *TDA Progress Report 42-124*, Feb. 1996.

[5] R. G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, MA: MIT Press, 1963.

[6] D. J. C. MacKay and R. M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electron. Lett*., vol. 32, no.18, pp. 1645-1646, 1996.

[7] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design and Iterative Decoding," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, May 1998.

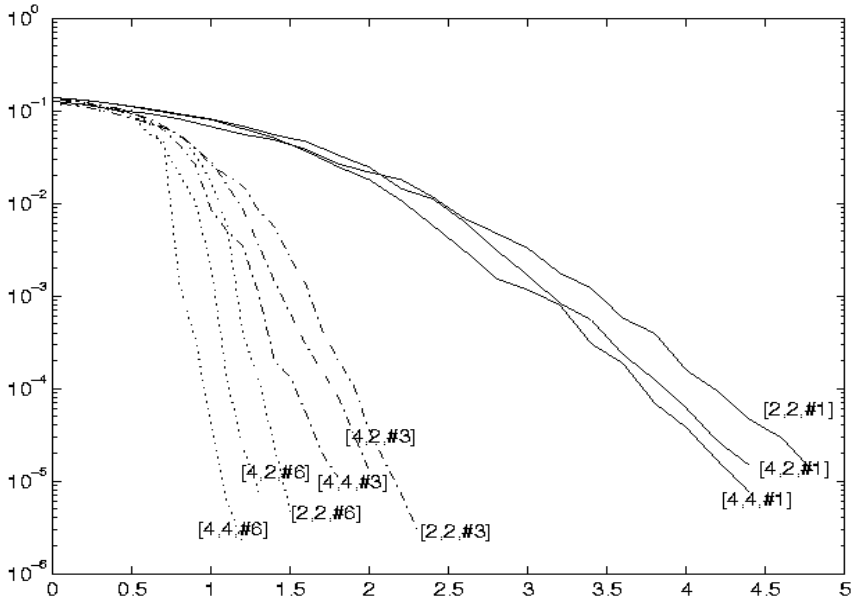

Figure 1. SNR vs. BER performance of 3 Turbo coding schemes with component codes of memory lengths 2 and 4 after 1, 3 and 6 decoding iterations

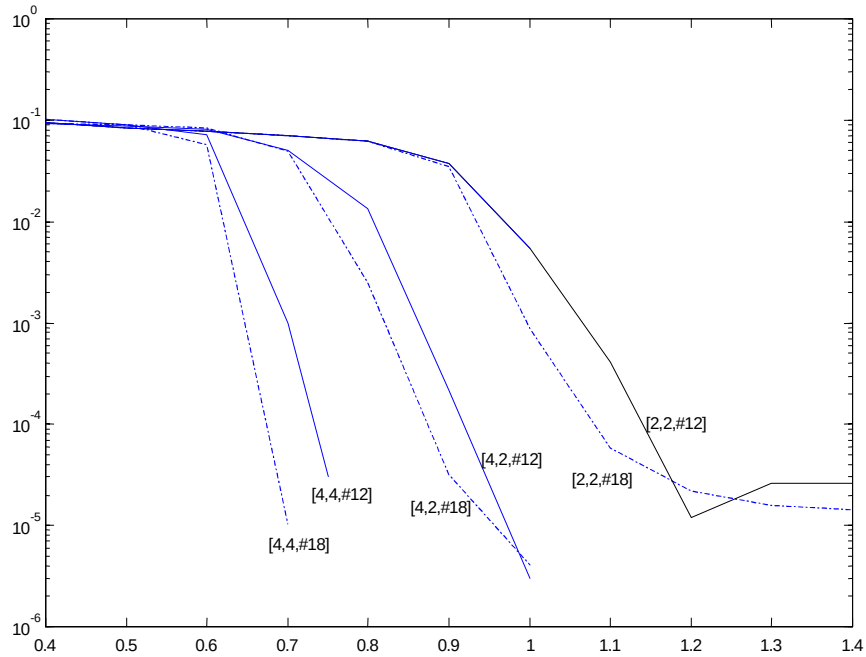

Figure 2. SNR vs. BER performance of 3 Turbo coding schemes with component codes of memory lengths 2 and 4 after 12 and 18 decoding iterations