18.413: Error-Correcting Codes Lab April 8, 2004

Lecture 16

Lecturer: Daniel A. Spielman

16.1 Related Reading

- Fan, pp. 108–110.
- Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding, by S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara.

16.2 Decoding Modules

Before describing some variations on Turbo codes, I'll give a more general view of convolutional code decoders. To begin, we'll consider decoders for general convolutional codes, in which the input bits might not appear among the output. I'll let w denote the message bits, the input to the encoder, and x denote the coded bits, the output of the encoder.

An *extrinsic decoder* (called SISO in the Benedetto, et. al. paper) will take as input estimates for the probability that each bit in x and w is 1. We view these inputs as coming in two streams, one for x and one for w. The channel typically provides such estimates for the bits in w. One can view the other decoder in a turbo code as providing such estimates for the bit in x. The decoder will also have two output streams, computing the extrinsic probabilities that the bits in x and w are 1. Here, extrinsic means computing the probability while taking into account all of the input data, except the data item corresponding to the extimate being output. In particular, if the code is systematic, so the input appears inside the output, then the extrinsic outputs for w will not use the corresponding estimates on the w input stream, but may use the estimates from the x input. For example, if $x_1 = w_1$, then the computation of $\mathrm{P}^{ext}[w_1 = 1]$ will use the input probability that $x_1 = 1$, but not that $w_1 = 1$.

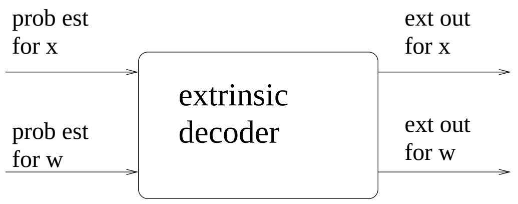

Figure 16.1: Extrinsic decoding module, as described in Benedetto, et. al.

Using this module, we can depict the Turbo decoding algorithm as follows:

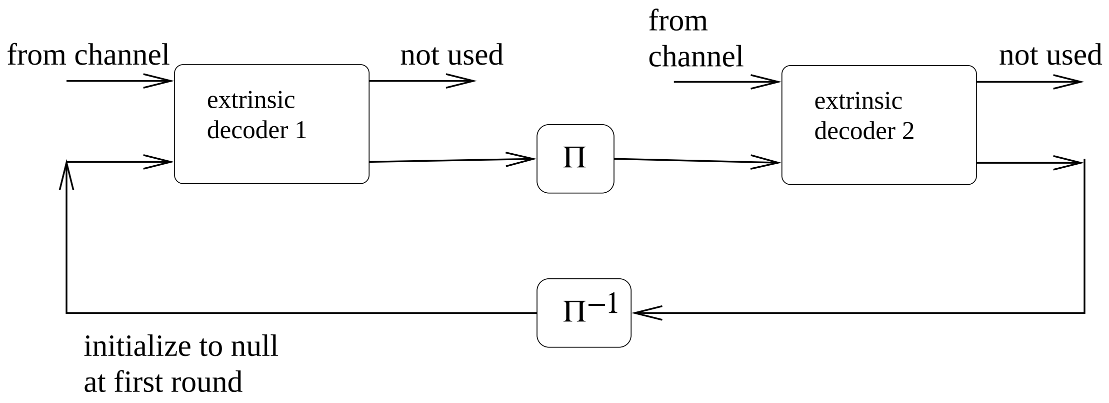

Figure 16.2: The description of the Turbo decoder by Benedetto, et. al.

One of the most-studied variations on Turbo codes are serially-concatenated codes. These are specified by one random permutation, and two convolutional codes. The input message, w, is first passed through the first convolutional encode to produce a word v. The word v is then passed through the permuter to produce v'. The word v' is then passed through the second convolutional encoder to produce the codeword x. Following Forney, the first encoder is called the outer code, and the second is called the inner code. For an picture, see Figure 16.3.

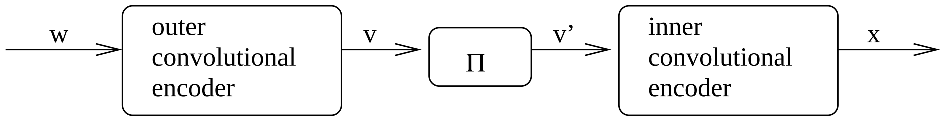

Figure 16.3: Serial concatenation scheme.

The decoder for serially-concatenated codes is very much like that for Turbo codes. Benedetto, et. al. describe it by the following figure:

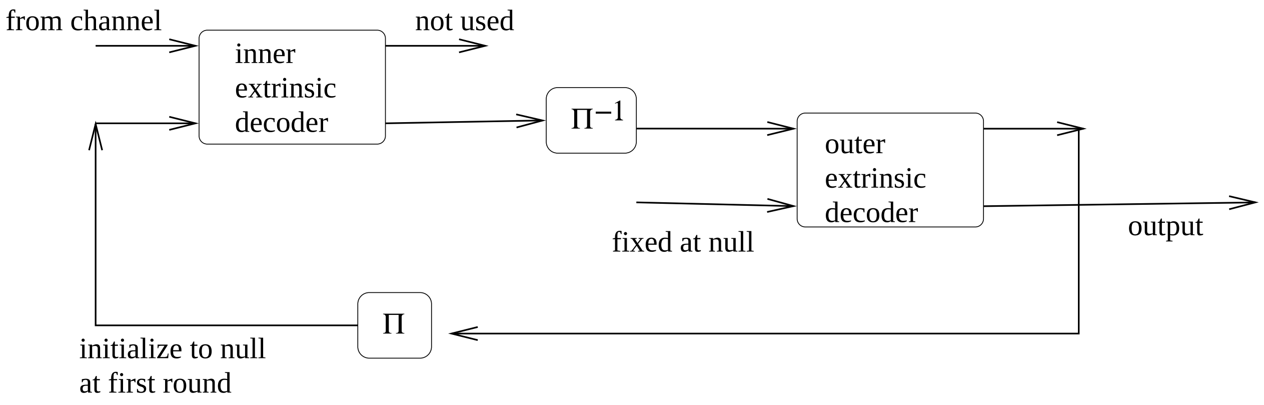

Figure 16.4: The description of the serially-concatenated code decoder by Benedetto, et. al.

The convolutional codes do not necessarily have to be recursive or systematic, although experiments reveal that the first should be systematic and the second should be recursive.

Some very simple realizations of the serial scheme seem to have good performance. For example, Repeat-Accumulate codes (see papers in Materials section) use a repetition code as the outer encoder, and an accumulator as the inner encoder, where an accumulator has the following description:

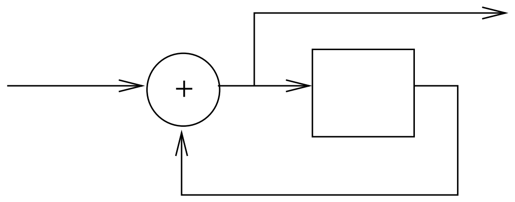

Figure 16.5: Accumulator.

16.3 Final Projects

Here are some suggestions for reasonable types of final projects. I'm open to others.

- Reproduce the experimental results from an interesting paper (see the Materials section of the course homepage). (1 person)
- In addition, produce EXIT charts for the coding system, and compare the behavior of the algorithm to the behavior predicted by the EXIT charts (2 people).
- Use EXIT charts and some optimization heuristics to try to improve the design of codes within a system. (3 people)
- For one of the papers that uses multiple ideas to obtain good codes, see what would happen if you dropped some of these ideas. For example, the GF (q) LDPC papers use larger fields and fancy distributions. Check what happens if you drop one of these. (1-2 people)
- Compare the behavior of various decoders for Turbo, LDPC, or other codes. There are many versions of the decoders that are less than ideal, but faster. (1-2 people).

The final projects will have two main products: a final report and an oral presentation. The presentations will be scheduled for the last 3 or 4 classes, and each student will take about 1/2 of a class. The final report should explain the experiment that was performed in the project and explain the results obtained. The explanation should assume as little background as possible on the part of the reader. In particular, one should not assume that the reader has taken this course. On the other hand, it is reasonable to assume that the reader has some general familiarity with coding theory.

The final project has three written parts: a proposal, a first draft, and a final draft. The proposal, due one week from today, should be a page or two explaining what you are going to do. If your proposal amounts to "I will reproduce the results of paper X", then you should flesh this out into a page or two so that you can convince me that you know what you are doing.

The first draft should contain all of the necessary expository material, and at least some preliminary experimental results. You will receive comments on your first draft that you will need to incorporate into the final draft. Unfortuntely, there can be no extensions on the deadline for the final project.

You may work in teams of up to three people on the final project. If you work in a team of three, I will expect one of you to give a presentation on the first day of presentations. If you work in a team, each team member must write their own final report. The magnitude of the final project should depend on the number of people in a team.

16.3.1 Important Note

While I would like to assume that the following note is unnecessary, I have been wrong in the past. You must be very careful not to plagerize material! There are many surveys and expository papers on this material, and, while you are welcome to read these for your education, you may not use sentences, figures, or organization from these works without attribution. Note again that this includes figures that you find on web pages. Similarly, you may not use code developed by others without attribution. I take violations of academic honesty very seriously. To quote my brother: "He's a very nice guy, but he's also a professor; and professors hate plagerism."