18.413: Error-Correcting Codes Lab April 15, 2004

Lecture 17

Lecturer: Daniel A. Spielman

17.1 Developments in iterative decoding

My main purpose in this lecture is to tell you some of the more interesting developments in the field of iterative decoding. Unfortunately, a lot has happened, so there is much that I will have to leave out. I hope that some of you will make up for this with your final projects.

17.2 Achieving Capacity on the BEC

You will recall that the capacity of the binary erasure channel (BEC) with erasure probability p is $1 - p$. It was proved [LMSS01] that LDPC codes can achieve the capacity of these channels. In particular, for any $\epsilon > 0$, there exists an infinite family of LDPC codes that have arbitrarily low error probability on the BEC with erasure probability $1 - p - \epsilon$. To explain how these codes are constructed, I will recall some material from Lecture 13.

In that lecture, we observed for the (3,6)-regular constructions,

- If the probability that each incomming message to a parity node is ? is a, then the probability that an outgoing message is ? is $(1 - (1 - a)^5)$.
- If the initial erasure probability was p_0 and the probability that each incomming message to an "=" (bit) node is ? is b, then the probability that a message leaving such a node is ? is $p_0 b^2$.

To produce better codes, we will use carefully specified irregular constructions. In particular, we will speciry the fraction of nodes of each type of each degree. While it might seem strange, we will do this from the perspective of edges. For example, the fractions for the bit nodes will be specified by a sequence

$$\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{d_l},$$

where d_l is the maximum degree of a bit node. The number λ_i specified the fraction of *edges* that are connected to bit nodes of degree i. We will also specify a sequence $\rho_1, \ldots, \rho_{d_r}$, where ρ_i is the fraction of edges that are connected to parity nodes of degree i. The parameters must satisfy

$$\sum \lambda_i = 1 \qquad \text{and} \qquad \sum \rho_i = 1,$$

and a condition ensuring that the number of edges specified by both sequences is the same. The reason that we specify the fractions in this form is that we obtain the equations:

$$a_{t+1} = p_0 \sum_i \lambda_i b_t^{i-1},$$
$$b_t = \sum_i \rho_i (1 - (1 - a_t)^{i-i}.$$

Thus, in order for the decoding to converge with high probability as the code grows large, we only need that the plot of $\rho(x)$ lies above $\lambda(y)$, where

$$\rho(x) = \sum_i \rho_i (1 - (1 - x)^{i-i},$$
$$\lambda(y) = p_0 \sum_i \lambda_i y^{i-1}.$$

Thus, the problem of designing erasure-correcting codes is reduced to the problem of finding the right polynomials. For example, here are two degree sequences that I checked work at $p = .048$

$$\lambda_2 = .2594, \lambda_3 = .37910, \lambda_{11} = .127423, \lambda_{12} = .237538, \rho_7 = 1.$$

Here are the two curves that we get for these polynomials at $p = .48$.

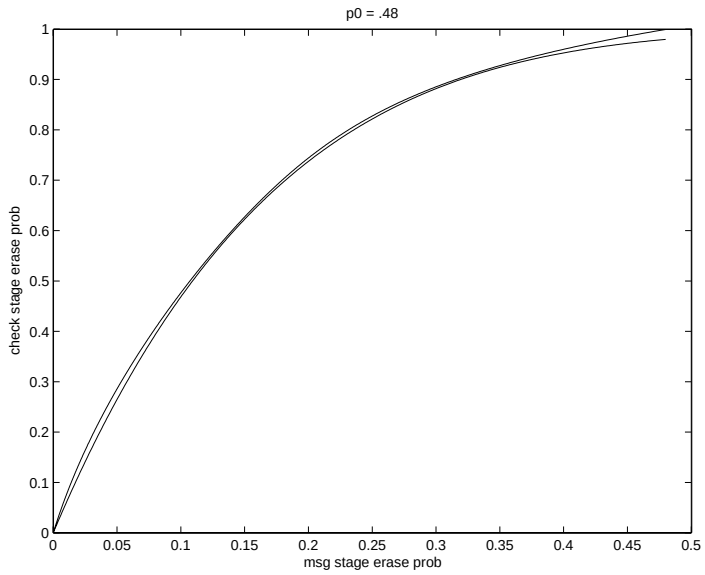

Figure 17.1: Very close curves, using the above degree sequences

Compare this with the curves for the standard $\lambda_3 = 1, \rho_6 = 1$:

One of the big questions in the area of iterative decoding is whether one can approach the capacities of channels such as the Gaussian and BSC. We now know that it is possible to come very close, but we don't know if one can come closer than every ϵ.

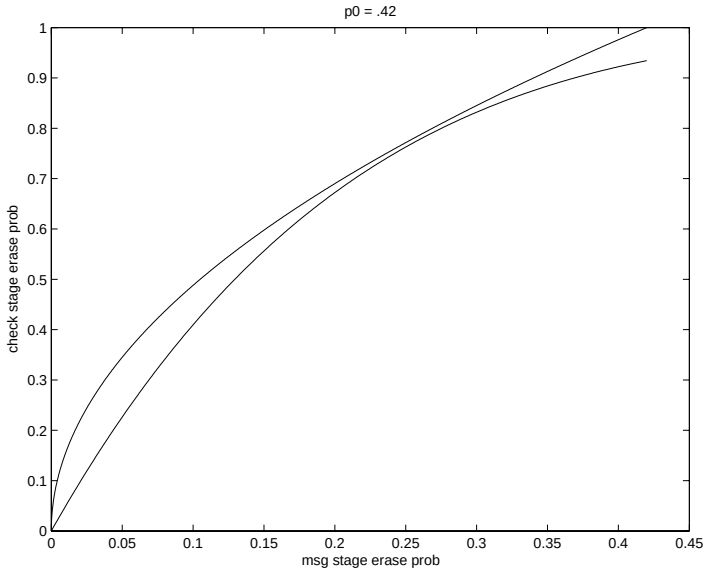

Figure 17.2: The curves for the standard (3,6)-graph

17.3 Encoding

One problem with LDPC codes is that it is not clear how to encode them efficiently. Clearly, the method that we used in Project 1 is unacceptable. However, for good families of irregular codes there is a fast encoder: we can essentially use the erasure correction algorithm to encode. That is, we set all the message bits, do a little bit more computation to find the values of a few other bits, and then treat the problem of encoding the rest as an erasure correction problem. Fortunately, this problem is even easier than that of erasure correction, because here we can choose the set of erasures that is easiest to correct, rather than the set output by the channel. The details of this were worked out by Richardson and Urbanke [RU01b].

I should also point out that Jin, Khandekar and McEliece showed how to construct irregular repeat-accumulate codes that also achieve capacity of the BEC, and that these can be encoded in linear time.

17.4 Density Evolution

One of the most important developments that we have dodged in this class is Density Evolution [RU01a]. Density Evolution extends the techniques used to analyze the behavior of LDPC on the BEC to more general channels. It is particularly effective on the Gaussian channel. The idea is to compute the distributions of messages being sent at each iteration. The difficulty is that these distributions might not have nice descriptions.

The Density Evolution approach is to represent a sketch of the density functions of these distributions. Richardson and Urbanke realized that for the particular operations that take place at bit and

parity nodes in belief propagation decoding, it is possible to compute the sketches of the density functions with high accuracy. This allows one to use computer computations to rigorously analyze how the algorithm will behave. This approach had a big impact, and inspired the less-rigorous but more easily applicable EXIT chart approach.

17.5 Exit Charts, revisited

I would now like to examine how EXIT charts are constructed for serially concatenated codes. To begin, let me recall the structure of the decoder:

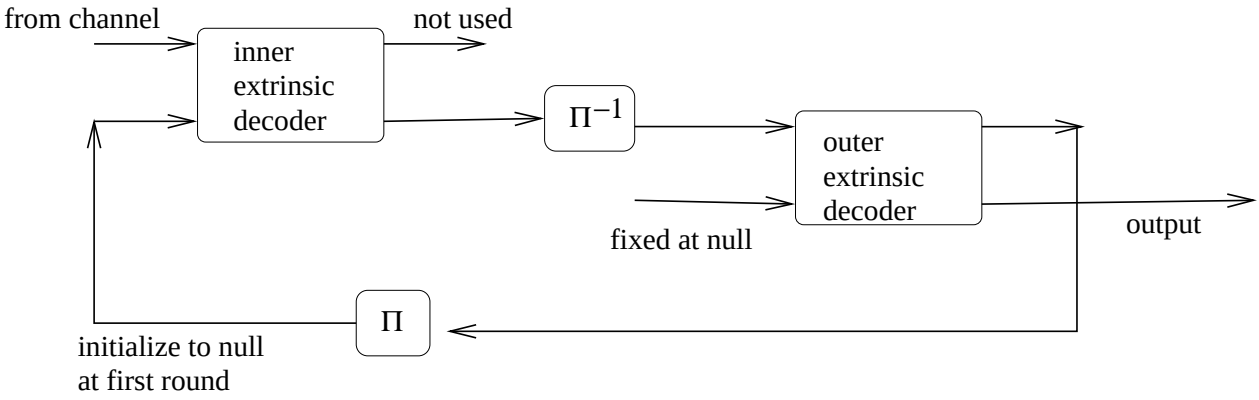

Figure 17.3: The description of the serially-concatenated code decoder by Benedetto, et. al.

As the roles of the inner and outer codes are different, their exit charts will be constructed differently. As the outer code decoder does not receive any inputs from the channel, its construction is easier. The outer decoder has just one input containing estimates for each coded bit. That is, its input looks like what one gets by passing the coded bits through a channel. So, we construct the EXIT charts by passing the coded bits through a channel, and plotting a point with X-coordinate the input channel capacity and Y-coordinate the observed capacity of the meta-channel (i.e. at the output of the decoder).

The exit charts for the inner decoder are slightly more complicated because these decoders have two inputs: one from the outer decoder and one from the channel. To form these charts, we will fix the channel. That is, we will get a different curve for each channel. We then pass two inputs to the decoder: the result of passing the coded bits through the actually channel, and the result of passing the message bits through a channel whose capacity we vary. As the capacity of this channel varies, we trace out a curve by measuring the observed capacity of the meta-channel.

17.6 Why we use bad codes to make good codes

Many have been mystified by why iterative coding schemes mainly combine codes that are viewed as "bad". That is, they use parity-check codes, or very simple convolutional codes. If one attempts to replace these with more powerful codes, the resulting iterative scheme is almost always worse.

For example, if one uses more powerful codes instead of parity checks in LDPC codes or if one uses complicated convolutional codes in Turbo codes, they almost never work well. The reason for this can been seen in EXIT charts. Ten Brink observed that the areas under the curves in the EXIT charts are almost always the same, and just seem to depend upon the rate of the code. In fact, for codes over the erasure channel, this was proved by Ashikhmin, Kramer and ten Brink: the area under an outer-code EXIT chart curve equals 1 minus the rate of the code (IEEE Transactions on Information Theory, to appear 2004). They prove a similar result for the inner codes: the area under the curve equals the capacity of the actual channel divided by the rate of the code.

Applying these results for the BEC, or their conjectured analogs for other channels, we find that a classically good code would have an unfortunately shaped EXIT chart: it would looks like a step function. The reason is that these codes will do well for any noise level less than the rate of the code, and that doing well in this region would require all the area of the curve. This will create trouble because the place where we would enter the curve would be on the wrong side of the step.

References

[LMSS01] Luby, Mitzenmacher, Shokrollahi, and Spielman. Efficient erasure correcting codes. *IEEETIT: IEEE Transactions on Information Theory*, 47, 2001.

[RU01a] Richardson and Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEETIT: IEEE Transactions on Information Theory*, 47, 2001.

[RU01b] Richardson and Urbanke. Efficient encoding of low-density parity-check codes. *IEEETIT: IEEE Transactions on Information Theory*, 47, 2001.