6.897 Algorithmic Introduction to Coding Theory	November 21, 2001
Lecture 18	
Lecturer: Madhu Sudan	*Scribe: Mihai Bădoiu*

Topics for today:

- Correct a lot of random errors in linear time.

In this lecture we focus on developing encoding and decoding algorithms for correcting p fraction of errors in linear time. The results presented are due to Forney, and Luby et al.

1 Correcting a fraction of p errors in polynomial time

The model used for this section is the Shannon model with a binary symmetric channel (BSC). Each bit is flipped with probability p.

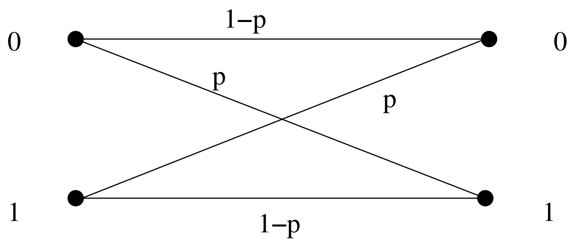

Figure 1: BSC. Each bit is flipped with probability p. Capacity is $1 - H(p)$

The encoding is going to be a function

$$E : \{0,1\}^k \to \{0,1\}^n$$

and the decoding

$$D : \{0,1\}^n \to \{0,1\}^k$$

such that the probability of decoding error goes to 0 and $\frac{k}{n} \to 1 - H(p)$.

Another channel which will be considered later in this lecture is the binary erasure channel (BEC) in which each bit can be replaced by "?" with probability p.

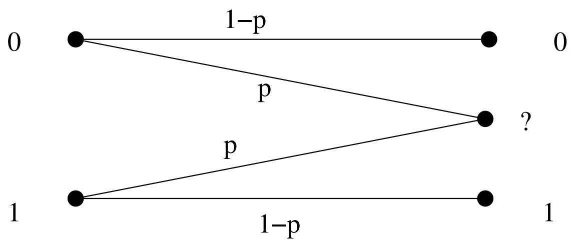

Figure 2: BEC. Each bit is replaced with "?" with probability p. Capacity is $1 - p$

Note that the capacity of BSC is less than the capacity of BEC.

Forney's main idea: pick a Seed Solomon code and concatenate it with a random linear function having the inner code small enough such that the decoding of the entire code takes polynomial time.

Let IC be a random linear code and $E = RS \circ IC$. For decoding we will use a Reed Solomon decoder for the outer code and brute force for the inner code. [1]

$$RS \rightarrow [n_1, (1-\epsilon)n_1, \epsilon n_1]_{n_1}$$
$$IC \rightarrow [n_2, (1-H(p)-\epsilon)n_2, (p+\delta)n_2]_2$$

where $2^{(1-H(p)-\epsilon)n_2} = n_1$. The concatenation of the two codes gives:

$$E = RS \circ IC = [n_1 n_2, (1-H(p)-2\epsilon)n_1 n_2, \epsilon(p+\delta)n_1 n_2]_2$$

We hope to correct a fraction of p of errors while the minimum distance of this code is some ϵ fraction as best. The question is: How are we going to use this encoding to have any success in decoding? Here is where the fact property that the erros are distributed at random is very useful. If we look at the picture of the code, we notice that the amount of errors in each block that we expect to see is a fraction p and the inner code is chosen so that it can handle a fraction of $p+\delta$ of errors. The probability that we get a decoding error in a block is exponentially small but only in its length, which is small, so you do expect a few errors, and the Reed Solomon code that we picked can handle a few errors.

The probability of decoding failure in a particular block after inner decoding is at most $e^{-c_\delta n_2}$. (from Shannon's theorem)

$$Pr[\text{more than } \frac{\epsilon n_1}{2} \text{ inner blocks are in error}] \leq e^{-c'_{\delta,\epsilon} n_1 n_2}$$

[2]

This result was due to Forney in 1966. We'll use other codes to improve the decoding time to be linear. Instead of a Reed Solomon code we'll use codes developed by Spielman.

Let the outer code OC be a Spielman code and the inner code IC be a random linear code: $E = OT \circ IC$. The size of the inner code will be of constant size.

$$Pr[\text{decoding the first block}] < e^{-c_\delta l}$$

$$Pr[\text{more than } \frac{\epsilon n_1}{2} \text{ inner blocks}] \leq e^{-c'_{\delta,\epsilon} n_1 l}$$

$$E \rightarrow [n_1, (1-\epsilon), \gamma n_1]_2$$

We will use Spielman codes as a black box, but we need the rate to be very close to 1, so we're not allowed to put too many parity check bits. Remember that the Spielman codes from the last lecture have rate $1/4$ and can handle some constant fraction of errors. The idea is to protect the parity check bits using a Spielman code. So if you replace the Reed Solomon code with Spielman codes you can get linear time encoding and decoding and exponentially small error probability.

If we get all check blocks then we can recover all message bits from γk errors.

$$RS \rightarrow \text{Spielman codes}$$

If we want the rate to be $1-H(p)-\epsilon$, running time of this is $O(2^{1/\epsilon^2} n)$. On the other hand, $O(n \log \frac{1}{\epsilon})$ seems to be achievable.

[1] Note that the brute force algorithm for decoding a linear code is exponential in the size of the code

[2] The constants can be improved by using more sophisticated error correcting codes.

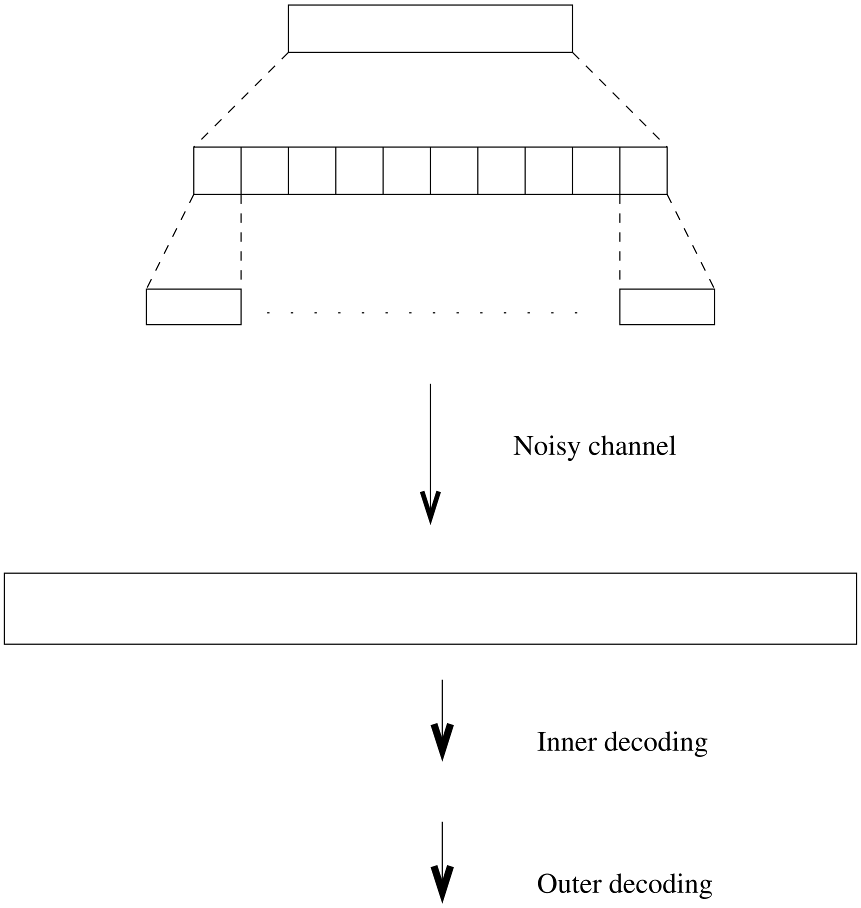

Figure 3: Overall picture of the error-correcting code.

2 Correcting a fraction of p errors in polynomial time for the binary erasure channel model

First we fix p parameter of BSC, δ the error probability of decoding and R the rate ($< 1 - H(p)$). Ask for the smallest n for which this is possible.

The results are from papers due to subsets of the following authors: Luby, Mitzenmacher, Shorcroccahi, Spielman, Stemann and Richardson, Urbanwke.

We will consider simpler assumptions for the binary erasure channel: the only bits which can be erased are the message bits. (i.e., no parity bits get erased) The erasure channel has capacity $1 - p$ which basically suggests that if total number of messages bits is at least the number of vertices on the left you're expected to decode.

2.1 The Algorithm

- Given partial assignment to left vertices, if there exists a right vertex with exactly one unknown left neighbor, update left neighbor

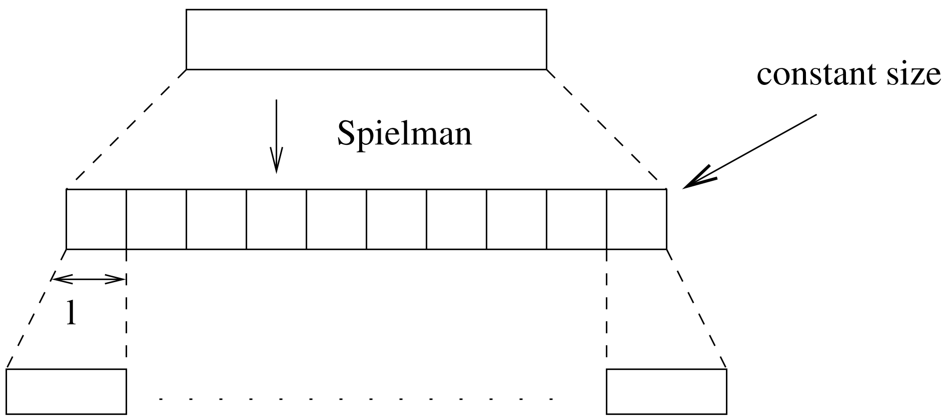

Figure 4: Concatenating Spielman codes with linear codes of constant size of $l = 1/\epsilon^2$

- Repeat step 1 until no more updates are possible.
- After this provess, if everything on the left is known you're successful otherwise you're not successful.

Basic idea: pick an irregular LDPC code. (i.e., degree on right part not regularly) The algorithm starts with p fraction erasures and ends with ϵ fraction erasures using $p+\epsilon$ fraction of parity check bits. We want to end up with 0 fraction erasures, but we are not allowed to solve a linear system. What we'll do instead is to put a Sipser-Spielman graph to correct the remaining errors.

Theorem 1 *Luby, Mitzenmacher, Shorcroccahi, Spielman, Stemann*
There exists a family of irregular LDPC codes that have rate $1 - p - \epsilon$ and can be decoded from p fraction of random errors in time $O(n \log \frac{1}{\epsilon})$ with error probability $2^{-\sqrt{n}}$.

The number of edges in the graph is chosen to be $O(n \log 1/\epsilon)$.
Sketch of Proof

- We have 2 random processes going on: one is the errors (erasures) and the other one is the choosing of the code. We will pick the erasures first and the code later.
- Analyze decoding while picking code.

■

The question we'll answer: what is the probability to recover a certain bit? We recover it if it is not erased or the neighbors of a right vertex has the "right" property. The property we'll need is that the other neighbor vertices of the right vertex are known (not erased or one of their neighbor has the right property), so we have an AND-OR evaluation tree.

We'll assume it's a tree (we have no cycles - almost true since it's a random graph) and the degree of the nodes on the left is chosen at random according to a Poisson distribution.

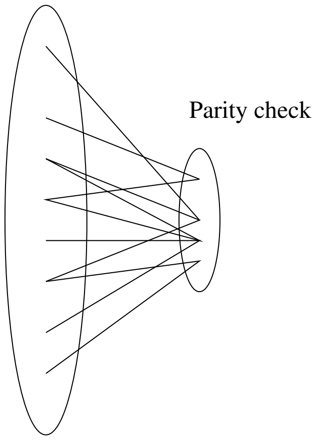

Figure 5: The Sipser Spielman graph: If the size of $|A| = k$, $|B| = \epsilon k/4$. The rate of the code is very close to 1.

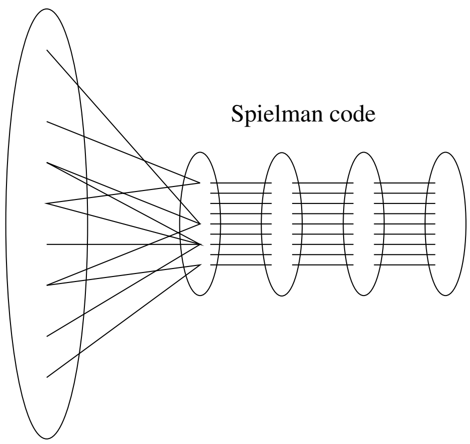

Figure 6: How to get Spielman codes with higher rate than $1/4$. Each Spielman code inside can handle γk errors.

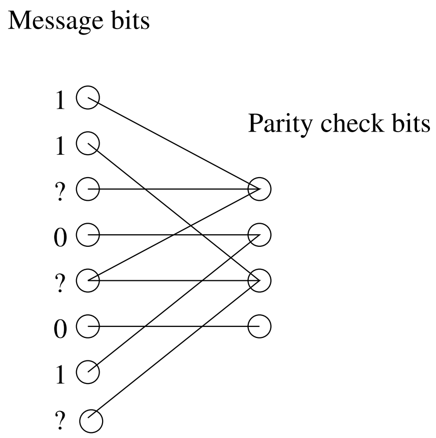

Figure 7: The bits on the right side of the graph are parity bits of its neighbors from the left side

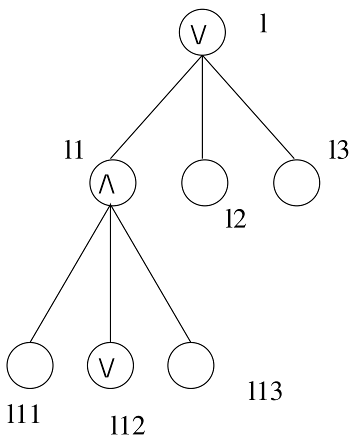

Figure 8: The AND-OR tree of evaluations