Lecture 16

Lecturer: Madhu Sudan *Scribe: Themis Gouleakis*

1 Overview

In the previous lecture we discussed a graph based code due to Spielman [2] which can has linear-time encoding and decoding. The main result can be summarized in the following theorem.

Theorem 1 *There exist values $R > 0, \tau > 0$ and a family of codes $\{C_n \subseteq \mathbb{F}_2^n\}$ of rate R correcting τ fraction of errors with linear time encoder and decoder.*

The following corollary of the above theorem is for the case of binary symmetric channel:

Corollary 2 *For every $p \in (0, 1/2)$ and $\epsilon > 0$ there exists some $\delta > 0$ and encoding/decoding functions:*

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

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

for $R = 1 - H(p) - \epsilon$ such that E, D are computable in linear time and

$$\Pr[D(E(x) + \eta) \neq x] \leq 2^{-\delta \cdot n}$$

where η is the (random) error introduced by the channel.

2 Error reduction codes

The key ingredient we will need in order to prove theorem 1, will be the notion of error reduction codes. In the following, we will see how the encoding and decoding works for these codes.

2.1 Encoder

We are given a (c, d)-regular bipartite graph G, which has k left edges (and thus $k \cdot c/d$ right edges). The left edges of the graph correspond to the bits $\{x_1, ..., x_k\}$ of the message, whereas the right edges are the parity check bits $\{y_1, ..., y_m\}$, where

$$y_j = \sum_{\{i,j\} \in G} x_i \ \ mod \ \ 2$$

Then the codeword, is computed as follows:

$$E(x_1, ..., x_k) = (x_1, ..., x_k, y_1, ..., y_m)$$

So, the encoding can be done in linear time, since we consider the degrees of the graph G to be constant.

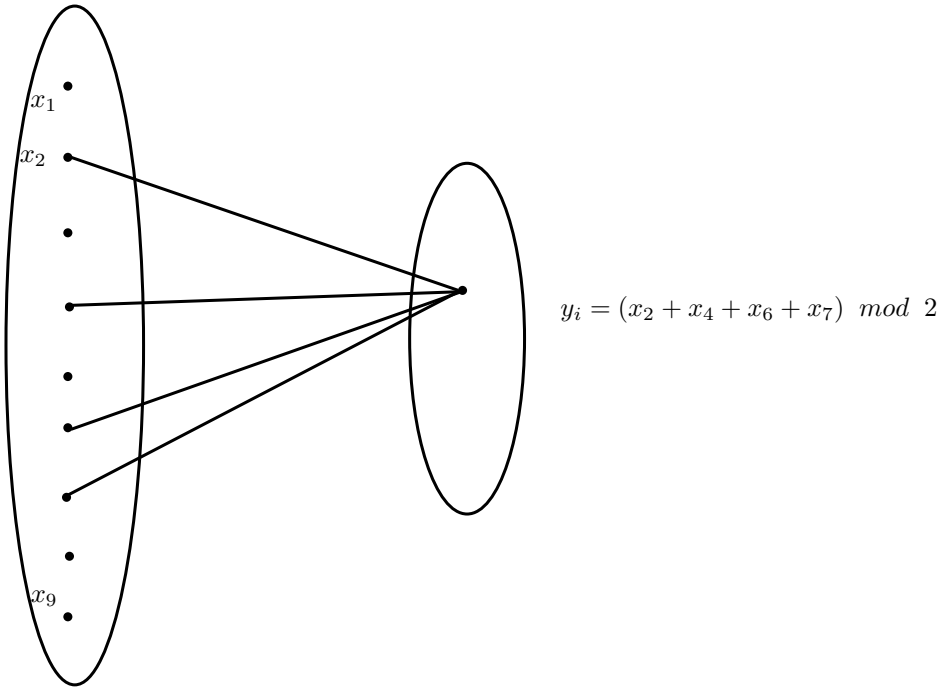

Figure 1: Example of a parity check bit.

2.2 Decoder (error reducer)

Let $(\tilde{x}, \tilde{y})$ be the (possibly) corrupted version of (x, y) that is given to the decoder and let $\hat{x}$ be the output of the decoder. We will call a right vertex of G **satisfied** if $\tilde{y_i} = \sum_{\{i,j\} \in G} \tilde{x_i} \ mod \ 2$. The decoding (or more accurately, error reducing) in this kind of codes is done using the following flipping algorithm:

ERROR-Reducer$(\tilde{x}, \tilde{y}) \to \hat{x}$

1. **While** there exists some $i \in [k]$ such that the i-th vertex on the left has more unsatisfied neighbors than satisfied ones **do** $\tilde{x_i} \leftarrow \neg \tilde{x_i}$.
2. **Return** $\hat{x} = \tilde{x}$

In other words, we flip the values of the bits in $\tilde{x}$, such that for each left vertex of G, the majority of its neighbors are satisfied when the algorithm terminates.

2.3 Good graphs for error reduction

The following definition describes a desirable property for graphs used in error reduction.

Definition 3 *A bipartite graph G is a τ- error reducer if for every $\tau_1, \tau_2 \leq \tau$ and $(x, y) = E_G(x)$, if $\delta(\tilde{x}, x) \leq \tau_1 \cdot k$, $\delta(\tilde{y}, y) \leq \tau_2 \cdot k$ then $\delta(\hat{x}, x) \leq \frac{\tau_2}{2} \cdot k$. Where $\delta(a, b)$ denotes the hamming distance between a and b and k is the size of the left independent set of G.*

Remark: Note that the upper bound on number of errors in the first part of the codeword after the algorithm has terminated, depends only on the number of errors in the second part. More importantly, if $\tau_2 = 0$,

then $\hat{x} = x$.

In the following lemma, we are going to give a sufficient condition for a graph G to be a τ-error reducer for some τ.

Lemma 4 *If G is a (c, d)-regular bipartite graph and a (γ, δ)-expander for some $\gamma > \frac{7}{8}c$ and an odd integer $c > 8$, then G is a $\frac{\delta}{2+c}$- error reducer.*

Proof *Let τ_1, τ_2 be such that*

$$\delta(\tilde{x}, x) \leq \tau_1 \cdot k;, \ \delta(\tilde{y}, y) \leq \tau_2 \cdot k$$

We first prove the following two claims:

***Claim 1**: The number of iterations of the flipping algorithm is at most: $\tau_2 \cdot k + \tau_1 \cdot c \cdot k$.*
***Proof**: This can be addressed as follows: Since the degrees of all left vertices are odd, after each flip, the number of unsatisfied right vertices decreases by at least 1. Moreover, this number is initially at most $\tau_2 \cdot k + \tau_1 \cdot c \cdot k$, since there can be at most $\tau_2 \cdot k$ unsatisfied vertices due to their value being incorrect ($\tilde{y}_i \neq y_i$) and each of the at most $\tau_1 \cdot k$ incorrect left vertices ($\tilde{x}_i \neq x_i$), can affect at most c right vertices and make them unsatisfied.*
***Claim 2**: Throughout the execution of the flipping algorithm: $\delta(\tilde{x}, x) \leq \tau_1 \cdot k + \tau_2 \cdot k + \tau_1 \cdot c \cdot k$.*
***Proof**: This follows easily from claim 1 using the fact that in each iteration, at most 1 bit can be changed.*

Furthermore, we want to choose the parameter τ to be such that $\delta(\tilde{x}, x) \leq \delta \cdot k$. In this way, we can use the expander graph property.
So, we get

$$\delta(\tilde{x}, x) \leq \tau_1 \cdot k + \tau_2 \cdot k + \tau_1 \cdot c \cdot k \leq \tau(2 + c) \leq \delta \cdot k \quad \Leftrightarrow$$

$$\tau \leq \frac{\delta}{2 + c}$$

***Claim 3:**After the flipping algorithm terminates, the following should hold:*

$$(2\gamma - c) \cdot |S| - \tau_2 \cdot k \leq \frac{c}{2}|S|$$

***Proof:** Consider the set S of the left vertices for which $\tilde{x}_i \neq x_i$. Using a lemma from previous lectures, we can get that S has at least $(2\gamma - c) \cdot |S|$ unique neighbors. Thus, it has at least $(2\gamma - c) \cdot |S| - \tau_2 \cdot k$ unaltered ($\tilde{y}_i = y_i$) unique neighbors. Notice that all these right vertices are unsatisfied, since only one of their neighbors is altered and they are not (so they have the different parity).*
So, if

$$(2\gamma - c) \cdot |S| - \tau_2 \cdot k > \frac{c}{2}|S|$$

after termination, then there will be at least $\frac{c}{2}|S|$ unsatisfied neighbors of S and by the pigeonhole principle at least one left vertex in S will have at least $\frac{c}{2}$ out of its c neighbors unsatisfied. So, the algorithm should not have terminated. So, claim 3 holds by contradiction.
From claim 3, we directly get:

$$|S| \leq \frac{\tau_2 \cdot k}{2\gamma - \frac{3}{2}c}$$

By comparing with the τ-error-reducer definition, we can see that we need:

$$2\gamma - \frac{3}{2}c \geq 2 \Leftrightarrow$$

$$\gamma \geq 1 + \frac{3}{4}c$$

We have that $\gamma \geq \frac{7}{8}c$, so it suffices that

$$\frac{7}{8}c \geq 1 + \frac{3}{4}c \Leftrightarrow$$

which holds from the hypothesis as well.

$$c \geq 8$$

■

3 Error correction codes from error reduction codes

Now, we are ready to use our error reduction codes in order to make error correction codes that correct all the errors given that they are not too many. The main idea behind that, is that we can use the error reduction guarantees of error reduction codes and apply them recursively in order to make error correcting codes.
Let

$$E_k : \{0,1\}^k \to \{0,1\}^{4k}$$
$$D_k : \{0,1\}^{4k} \to \{0,1\}^k$$

be the encoder and decoder of our error-correcting code (of message length k) respectively and

$$ERE_k : \{0,1\}^k \to \{0,1\}^{3k/2}$$
$$ERD_k : \{0,1\}^{3k/2} \to \{0,1\}^k$$

be the encoder and decoder of a τ-error reduction code of message length k. Our error correcting code, will be able to correct $\tau \cdot k$ errors (which is a $\tau/4$ fraction of the codeword).

3.1 Encoding

The encoder E_k does the following transformation: $x \to xyzw$.
The bit strings y, z, w are computed using the following transformations:

$$xy \leftarrow ERE_k(x)$$
$$z \leftarrow E_{k/2}$$
$$yzw \leftarrow ERE_{2k}(yz)$$

Note that the second step is a recursive application of the encoding function on a bitstring with half the size. When the size falls below some constant n, we can use any error-correcting code C of rate $R = 1/4$ which corrects a $\tau/4$ fraction of the errors.

3.2 Decoding

For the decoding procedure ($D_k(\tilde{x}\tilde{y}\tilde{z}\tilde{w}) = \hat{x} = x$), we work as follows:

$$\hat{y}\hat{z} \leftarrow ERD_{2k}(\tilde{y}\tilde{z}\tilde{w})$$
$$\hat{\hat{y}} \leftarrow D_{k/2}(\hat{y}\hat{z})$$
$$\hat{x} \leftarrow ERD_k(\hat{x}\hat{\hat{y}})$$

Our middle step for the decoding is recursive as well and we use the decoder of the code C we used above at the base case. Note that for the base case, we can use any type of code with the desired properties without

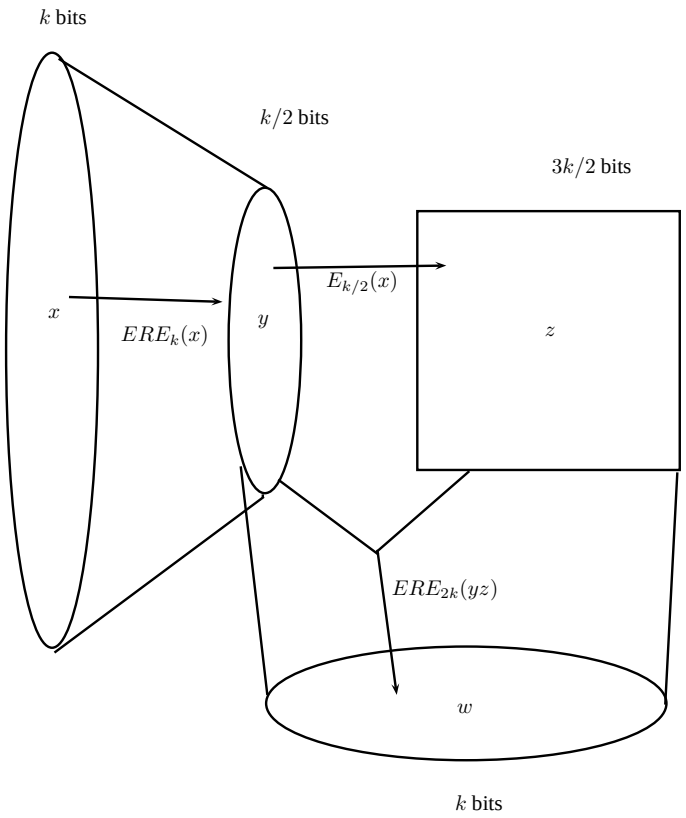

Figure 2: Construction of the $E_k(x)$ error correcting code

affecting the running time, since the size of the message being encoded and decoded will be constant.

In order to prove the correctness of decoding (i.e $\hat{x} = x$ if at most $\tau \cdot k$ bits are corrupted), we do the following:

$$\delta(\tilde{x}\tilde{y}\tilde{z}\tilde{w}) \leq \tau \cdot k \Rightarrow \delta(\tilde{w}, w) \leq \tau \cdot k \leq \tau \cdot (2k) \wedge \delta(\tilde{y}\tilde{z}, yz) \leq \tau \cdot (2k)$$

Since the graph involved in the error reduction code is a τ-error reducer, it follows that :

$$\delta(\hat{y}\hat{z}, yz) \leq \frac{\tau \cdot k}{2} = \frac{\tau}{4}(2k)$$

Since, by the induction hypothesis, our code can handle $\tau/4$ fraction of errors, we can recurse up to an $O(\log k)$ depth until we can apply the error-correcting code C which corrects all errors. Then, making our way up in the recursion we can prove by induction that $\hat{\hat{y}} = y$.
In other words $\delta(\hat{\hat{y}}, y) \leq \tau_2 \cdot k,\ \tau_2 = 0$ and it also holds that $\delta(\tilde{x}, x) \leq \tau \cdot k$. So, by the properties of the error reduction codes, since $\tau_2 = 0$, it follows that $\hat{x} = x$.

So, by applying a τ-error reducer recursively, we managed to construct an error correcting code of rate $R = 1/4$ which correct a $\tau/4$ fraction of errors. The key intuition for why this works, is that in the worst case (i.e when all the $\tau \cdot k$ errors appear in w) the error reducer halves the number of errors that are handed over to a similar subproblem of half the size. So, the conditions are still satisfied and we can reduce the size of the problem to constant and then apply any suitable error correcting code.

3.3 Running time analysis

We have already established that the running time for the error reduction encoding and decoding is linear. So, let c_0, c_1, c_2 be constants such that $T_{ERE}(k) = c_1 \cdot k$ is the running time for the encoding ERE_k, $T_{ERD}(k) = c_2 \cdot k$ is the running time for the decoding ERD_k and c_0 is the an upper bound for either encoding or decoding the code C (remember that only use codewords of up to some constant size for the code C).

We will prove by induction that the total running time of the encoding is $T_E(k) = 6c_1 \cdot k + c_0$ and for the decoding it is $T_D(k) = 6c_2 \cdot k + c_0 = O(k)$.
We have the following:

$$T_E(k) = T_{ERE}(k) + T_E(k/2) + T_{ERE}(2k) \Leftrightarrow$$
$$6c_1 \cdot k + c_0 = c_1 \cdot k + (3c_1 \cdot k + c_0) + 2c_1 \cdot k$$

which holds.
The analysis for the decoding running time is completely analogous.

4 Erasure correcting codes[1]

Consider the model of a binary erasure channel, where each transmitted bit is erased (i.e turns into a "?" symbol) with probability p and is received correctly with probability $1 - p$. It turns out that the capacity of this channel is $1 - p$. In this case we can do a similar encoding using bipartite graphs where the right vertices do a parity check of their neighbors. Now, consider the erased bits S which corresponds to a subset of the left vertices in the graph, and its neighborhood $\Gamma(S)$. In order to recover the erased bits, we run the following algorithm:

1. Find a degree 1 left vertex v, by which you can deduce the correct value of its neighbor in S (since it is the only erased neighbor, we can just compute the XOR of v with its non-erased neighbors and find the correct value).
2. Remove the left vertex corresponding newly discovered message bit from the set S and repeat step 1.

Notice that in step 2, by removing the vertex, the degrees of its neighbors within the new set S will decrease by 1. So, it is possible that some degree 2 vertices will become degree 1 and under certain conditions we will be able to correct all the erasures using this algorithm. In fact, there exist codes of rate $1 - p - \epsilon$ that correct a p fraction of random erasures (i.e probability p of erasure) in time $O(n \log(\frac{1}{\epsilon}))$.

References

[1] Michael Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2001.

[2] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.