CS 880: Advanced Complexity Theory 3/8/06

Lecture 17: Linear Time Expander Codes

Instructor: Dieter van Melkebeek Scribe: Vinay Choudhary

In the last lecture, we use lossless bipartite expanders to construct the parity check matrix, which gives us codes with quadratic encoding time and linear decoding time. Specifically, using a lossless bipartite expander with left-degree d_L which is $(\alpha N, d_L(1-\epsilon))$-expanding, we can get a code with rate $r \geq 1 - \frac{M}{N}$ and relative distance $\delta \geq \alpha$ provided $\epsilon < \frac{1}{2}$. We also show how to correct up to $\frac{\alpha}{2}N$ errors in linear time provided $\epsilon < \frac{1}{4}$.

In this lecture, we improve that by constructing a generator matrix (using lossless expanders) so that we can reduce the encoding time to linear. We also look at another construction of linear time decodable good expander code, for which weaker expanders suffice in combination with a base code which is fixed for the entire family.

1 Linear Time Encodable and Decodable Good Codes

For the construction, we are going to use systematic codes. These are codes in which the codeword consists of message bits followed by some check bits. As mentioned at the end of last lecture, the strategy is to look at the 'N' side (Figure 1, Lecture 16) as the information and the 'M' side is considered as the parity checks. The encoding then is all the original N bits plus the parity check bits. Unfortunately, as we discussed, this code in itself is really bad. But we can use it as a building block to recursively build better codes.

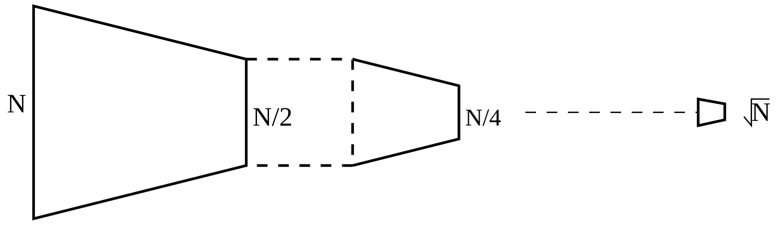

Figure 1: First attempt at a recursive construction of linear time encodable and decodable codes

The first idea is to use the lossless expanders from the last lecture to get a systematic code that encodes in linear time. We first obtain $\frac{N}{2}$ parity check bits from the original N bits. Then obtain $\frac{N}{4}$ bits from those $\frac{N}{2}$ bits. We continue this recursively (see Figure 1). Our encoding is the original N bits concatenated with everything that we have computed. The encoding time is linear in the number of input bits at each step and since we have a geometric series, the total encoding time is linear in N. Note that instead of going all the way to a constant number of bits, we can stop at $\sqrt{N}$ bits which are then encoded using *brute-force* (i.e. quadratic encoding time) method of last lecture.

Now consider the decoding. For the linear time decoding algorithm to work, we require that the errors are spread uniformly over all the steps. This is because each step has its own bound on the fraction of errors that can be corrected. But this cannot be guaranteed here. So as is, these codes aren't very good. Therefore, we apply another idea.

The decoding algorithm described in the last lecture doesn't work here but we can use it to achieve something here: "error-reduction". It turns out that these codes have a good error-reduction property which is what we will exploit. The proof of this property is an exercise in Homework 2.

Definition 1 (Error-reduction property). *Let $\epsilon < \frac{1}{4}$ and d be sufficiently high. Let w be a codeword that differs from the input in $\leq \frac{\alpha}{2}N$ positions, of which t are check bits, then the output of the decoding algorithm differs in the message bits from w in $\leq \frac{t}{2}$ bits.*

To achieve this, we compute parity bits using E_N, and then apply $C_{N/2}$ which we have constructed. At this stage, we have $2N$ bits in all. We then apply E_{2N} to all the bits (see Figure 2). Our encoding is everything that we have computed.

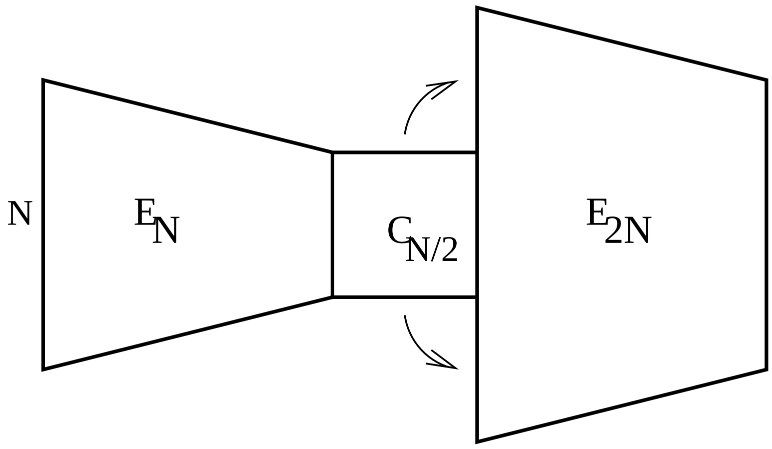

Figure 2: Linear time encodable and decodable good code

Using this construction, we can correct up to $\frac{\alpha}{2}N$ errors. Since the total number of bits in the codeword is $4N$, the rate is $\frac{1}{4}$. The encoding is linear time because we can write $T(N) = T(N/2) + O(n)$ where $T(N)$ is the time to encode N-bit message. Similarly, decoding is also linear time.

2 Linear Time Decodable Good Codes Based on Weaker Expanders

We use the double cover of a d-regular expander on N bits. The double cover of a graph is a bipartite graph obtained by placing its vertices on both sides and for each edge (u, v), connecting the left copy of u to the right copy of v. We note that the idea underlying the proof works in the general case, but the analysis is simpler for double covers. We will use the expander mixing lemma in the proof. Note that the expander mixing lemma holds for the double cover with λ_2 of the original graph. The proof of this can be obtained by simple modifications in the proof of the expander mixing lemma.

Suppose the edges of double cover are assigned 0 or 1. Consider any vertex of the double cover. It has exactly d edges. For each vertex, fix some ordering of the edges incident to it. We can then associate a d-bit strings to each vertex, viz. the one formed by taking the bits assigned to its incident edges in that fixed order. Now let C_0 be a $[d, r_0 d, \delta_0 d]$ code. We put the codeword bits on the edges of the expander. We say that w is valid iff at each vertex, the resulting d-bit string is in C_0.

The parameters of this construction are:

- $n = Nd$ because there are N vertices on the left side each having degree d.
- $r \geq \frac{Nd - 2N(1-r_0)d}{Nd} = 2r_0 - 1$. Here is the reason. Nd is the total number of free bits. For each vertex, $(1 - r_0)d$ is the number of constraints and hence, $2N(1 - r_0)d$ constraints in all. This is where we use the fact that C_0 is a linear code.
- $\delta \geq \delta_0(\delta_0 - \lambda_2)$.

Proof of last inequality. First we give some intuition. Consider a vertex on the left such that one of the edges incident to it is 1. Then at least $\delta_0 d$ of its edges must be 1. This is because the relative distance of C_0 is $\delta_0 d$. Then, each of the right end points of such edges have at least one of their edges which is 1, and hence, have at least $\delta_0 d$ of edges that are 1 and so on. The number of edges that are 1 is at least $\delta_0 d$ times the number of vertices on the left (or right) which have at least one of their incident edges marked 1. By expander mixing lemma, a small number of vertices will have a small number of edges between them and so to achieve this ratio of $\delta_0 d$, both the number of edges that are 1 and the number of vertices which are incident to these edges must be large enough.

Now the formal proof. Let $B = \{\text{Edges that are } 1\}$. S, T are vertices on the left and the right, respectively, that touch B. As explained above, every vertex in S has at least $\delta_0 d$ edges into T and every vertex in T has at least $\delta_0 d$ edges into S. We have, $\delta_0 d|S| \leq |E(S,T)|$ and $\delta_0 d|T| \leq |E(S,T)|$. Therefore, $\frac{\delta_0 d|S| + \delta_0 d|T|}{2} \leq |E(S,T)|$. Now using the fact that geometric mean is less than or equal to arithmetic mean, $\delta_0 d\sqrt{|S||T|} \leq |E(S,T)|$, which, by the expander mixing lemma, is at most $\frac{|S||T|d}{N} + \lambda_2 d\sqrt{|S||T|}$. Therefore,

$$\sqrt{|S||T|} \geq (\delta_0 - \lambda_0)N \tag{1}$$

For the same reason as earlier, $|S|\delta_0 d \leq |B|$ and $|T|\delta_0 d \leq |B|$ and we get,

$$\delta_0 d\sqrt{|S||T|} \leq |B| \tag{2}$$

From (1) and (2), $|B| \geq \delta_0(\delta_0 - \lambda_2)dN$. □

Theorem 1. *For any constant $a > 1$, we can correct up to $\frac{\delta_0}{2}(\frac{\delta_0}{2} - a\lambda_2)$-fraction of errors in linear time (i.e. $\Theta(\frac{1}{a-1}N)$).*

Proof. The algorithm is simply to do the following for all of the vertices on the left: decode the d-bit string on the edges incident to a vertex to the closest codeword in C_0. Some of the vertices would have less than $\frac{\delta_0}{2}$ "incorrect" edges incident to them and we would decode these correctly. For others, th d-bit string might be more than $\frac{\delta_0}{2}$ away from the right codeword, and we might be doing something junk, but that's fine. Call this a left phase. Then do the same for vertices on the right. This is a right phase. Now repeat until there is no change in some phase.

For the analysis, fix an underlying valid codeword w. Consider two successive phases: a left phase followed by a right phase. Let F be the set of vertices (on the left) that fail to decode to the correct C_0 codeword after the first phase and F' (on the right) be the corresponding set after the next phase. Let B denote the positions that disagree with w after the first phase.

Consider the vertices in F'. Each of these has at least $\frac{\delta_0 d}{2}$ incorrect edges all of which must their other end points in F. So we have,

$$\begin{aligned} \frac{delta_0 d}{2}|F'| \leq |E(F,F')| &\leq \frac{|F||F'|d}{N} + \lambda_2 d\sqrt{|F||F'|} \\ &\leq \phi d|F| + \lambda_2 d\frac{|F|+|F'|}{2} \end{aligned}$$

where $\phi = \frac{|F|}{N}$ and we used the exapnder mixing lemma to get the second inequality. Therefore,

$$(\delta_0 - \lambda_2 - 2\phi)|F'| \leq \lambda_2|F|$$

and if $\phi \leq \frac{\delta_0}{2} - a\lambda_2$, then, for $a > \frac{1}{2}$, we have,

$$|F'| \leq \frac{1}{2a-1}|F|$$

So we see that the number of bad vertices reduces by a constant fraction at every phase (if the condition $\phi < \frac{\delta_0}{2} - a\lambda_2$ is met). Initially, $|F^{(1)}|\frac{\delta_0 d}{2} \leq |B^{(0)}|$ and so we're fine as long as $|B^{(0)} \leq \frac{\delta_0}{2}\left(\frac{\delta_0}{2} - a\lambda_2\right)dN$.

To see that the time taken is linear, note that in any phase, we need to check only those vertices that are incident to the ones that changed in the previous phase. Therefore, the amount of work in any phase is linear in (can be at most d times) the number of vertices that changed in the previous phase, which is linear in the size of set F in the phase before that. Hence, the time taken decreases geometrically. The total time is $\left(\sum_{i=0}^{\infty}\left(\frac{1}{2a-1}\right)^i N\right) = O(\frac{1}{a-1}N)$.

□

For this code, $r \geq 2r_0 - 1$ and $\delta \geq \delta_0(\delta_0 - \lambda_0)$. From the Gilbert-Varshamov bound, we bound r by $1 - H(\delta_0)$ and if λ_0 is small, $\delta_0 \approx \delta_0^2$. This gives $r = 2(1 - H(\delta_0)) - 1 \approx 1 - H(\sqrt{\delta_0})$ (See Figure 2). Recently, it has been shown that we can improve this by using expanders again. We will just state the theorem.

Theorem 2. *There exist linear time encodable and decodable codes that can correct a fraction* $\frac{1}{4} - \epsilon$ *of errors with rate* $\Omega(\epsilon^3)$.

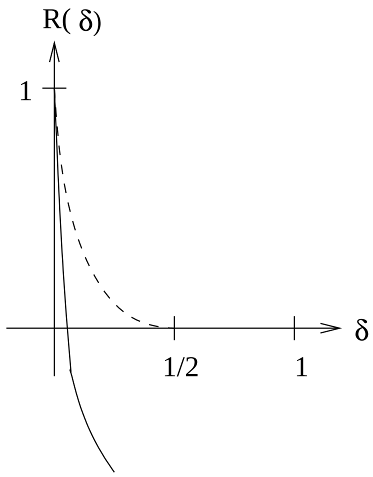

Figure 3: Comparison of the weak expander code and Gilbert-Varshamov bound.