6.895 Essential Coding Theory	November 10, 2004
Lecture 18	
Lecturer: Madhu Sudan	*Scribe: Venkat Chandar*

1 Overview

In this lecture we will review the Alon et al [ABNNR] code construction and present a generalization due to Alon, Edmonds, and Luby [AEL]. Several algorithmic results due to Guruswami and Indyk will also be discussed. A suggested reference for this material is the survey paper "Error-correcting Codes and Expander Graphs", by Guruswami.

2 Alon, Brooks, Naor, Naor, and Roth [ABNNR] Construction

Consider the following question: Given a code C_0, how can we form a family of codes $\mathbf{C} = \{C_n\}$ that achieve the same rate and relative distance as C_0 over a larger alphabet? We will see that one can view the ABNNR construction as an answer to this question when C_0 is a repetition code.

Recall the ABNNR construction. We take an $(n, k, \delta \cdot n)_2$-code C and a bipartite graph G that is a d-regular (γ, δ) expander with n left vertices and n right vertices. To form a new code over a larger alphabet, we first encode a length k message using C. This gives us a length n message m. We assign each bit of m to a left vertex of G. Then, to produce the final encoding, we noted that each right vertex of G has d neighbors, so we assign to each right vertex the d-bit string corresponding to the values of its neighbors.

Theorem 1 *Given an $(n, k, \delta \cdot n)$ code C and a bipartite, d-regular graph G that is a (γ, δ) expander, the encoding process above creates an $(n, \frac{k}{d}, \gamma \cdot \delta \cdot n)_{2^d}$-code.*

Proof

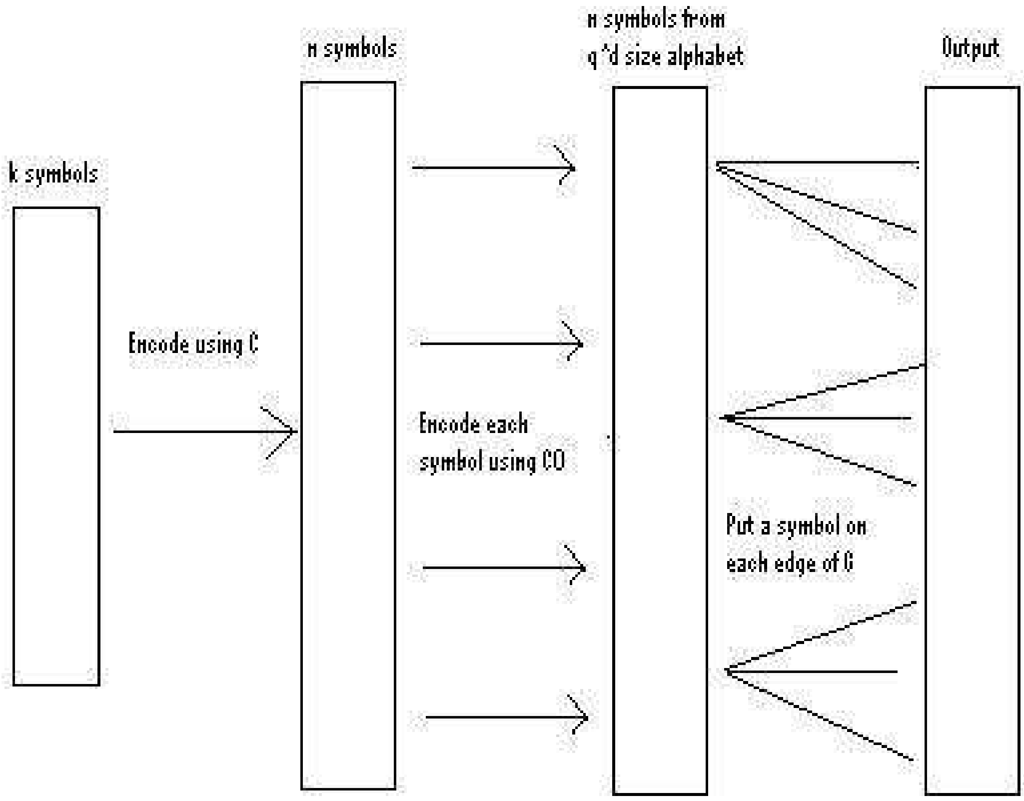
We start with k bits, which can be interpreted as $\frac{k}{d}$ elements of $\{0,1\}^d$, so the new code reduces the rate by a factor of d. Assuming C is additive, we know that any codeword of C has weight at least $\delta \cdot n$. The expander properties of G then force at least $\gamma \cdot \delta \cdot n$ of the d-tuples we output to be nonzero, so any codeword of the final code has weight at least $\gamma \cdot \delta \cdot n$. Therefore, this encoding function produces an $(n, \frac{k}{d}, \gamma \cdot \delta \cdot n)_{2^d}$-code.
■

Before we move on to the AEL construction, let us compare the performance of the ABNNR code to algebraic geometry codes. As noted earlier, ABNNR reduces the rate by a factor of d. The relative distance, however, is multiplied by γ, and as d becomes large,γ can approach $1 - \frac{1}{d}$. Thus, ABNNR gives us a code with rate $\Omega(\epsilon)$ and relative distance $1 - \epsilon$ as d becomes large. This is close to the singleton bound, but the price we pay for such performance is an exponential increase in the alphabet size, which is $2^{O(\frac{1}{\epsilon})}$.

The algebraic geometry codes are $[n, k, n - k - \frac{n}{\sqrt{q}-1}]_q$ codes. If we take $k = \frac{\epsilon \cdot n}{2}, \frac{n}{\sqrt{q}-1} = \frac{\epsilon \cdot n}{2}$, then the AG code has rate $\Omega(\epsilon)$ and relative distance $1 - \epsilon$. The alphabet size, however, is only $q = O(\frac{1}{\epsilon^2})$, so the AG code has a much better dependence on ϵ than the ABNNR code. The ABNNR code, however, is much simpler to understand conceptually.

3 Alon, Edmonds, and Luby [AEL] Generalization

Alon, Edmonds, and Luby noticed that in the ABNNR construction, we are essentially assigning values to the edges of the expander graph G using a repetition code on the left vertices. Instead of this naive



Encoding Process for AEL Code

method, the AEL construction takes another parameter, a code C_0. This code will be used to assign values to the edges of G instead of a simple repetition code.

Before we define the encoding process, we will need to define a new type of graph.

Definition 2 *Let G be a d-regular, bipartite graph with a set L of left vertices and a set R of right vertices satisfying $|L| = |R| = n$. G is a (d, ϵ)-uniform graph if $\forall X \subseteq L, Y \subseteq R, \beta = \frac{|X|}{n}, \gamma = \frac{|Y|}{n}$, the number of edges between X and Y is $\geq (\beta \cdot \gamma - \epsilon) \cdot d \cdot n$.*

Claim 3 *$\forall \epsilon, \exists d < \infty$ such that $\forall$ sufficiently large n, (d, ϵ)-uniform graphs exist.*

Now we can present the AEL construction. This construction requires an $(n, k, D)_{q^a}$ code C, a $(d, R \cdot d, \delta \cdot d)_q$ code C_0, and a (d, ϵ)-uniform graph G. G has n left vertices and n right vertices, and we also must have $a = R \cdot d$.

To encode, we start with with a message of the code C, i.e. k elements from an alphabet of size q^a. We encode this message using C, resulting in n elements from the same alphabet. Now, as in ABNNR, we assign each of the n elements to a left vertex of G. The key difference between AEL and ABNNR is how we assign values to the edges of G. Each left vertex has been assigned an element of $q^a = q^{R \cdot d}$. This element can be interpreted as a message of length $R \cdot d$ over an alphabet of size q, which means we can encode this element using C_0. This results in d elements from an alphabet of size q, so we can place one of these elements on each edge leaving the vertex. Then, each right vertex is assigned the d-tuple corresponding to the edges incident to it.

Theorem 4 *The AEL construction produces an* $(n, R \cdot k, \delta - \frac{\epsilon \cdot n}{D})_{q^d}$ *code*

Proof

For convenience, define $\beta = \frac{D}{n}$. β is the relative distance of C. Then, we want to show that AEL constructs an $(n, R \cdot k, \delta - \frac{\epsilon}{\beta})_{q^d}$-code. First, since we assign each right vertex a d-tuple from an alphabet of size q, the output has length n over an alphabet of size q^d. We start with k elements from an alphabet of size $q^a = q^{R \cdot d}$, so the message length is $R \cdot k$. To bound the distance, we must use the properties of G.

Assume C and C_0 are additive codes, so that as usual we only have to deal with the weight of a codeword. Consider the first step of the encoding procedure. From the definition of C, we end up with a codeword of length n and weight at least $D = \beta \cdot n$. We want to bound the weight of the output word, so we must bound the number of right vertices that have all their incident edges assigned to zero. Let X be the subset of the left vertices of G that are not zero. We know $|X| \geq \beta \cdot n$. Let Y be the set of right vertices of G that are assigned the value 0.

Now, because C_0 has relative distance δ, we know that every element of X will have at most $(1-\delta) \cdot d$ of its edges set to 0, since the weight of any nonzero codeword of C_0 is at least $\delta \cdot d$. Thus, the number of edges leaving X that are labelled zero is $\leq |X| \cdot (1 - \delta) \cdot d$. Since every member of Y is labelled 0, this implies that the number of edges from X to $Y \leq |X| \cdot (1 - \delta) \cdot d$. On the other hand, G is a (d, ϵ) uniform graph, which means that the number of edges from X to $Y \geq (\frac{|X| \cdot |Y|}{n^2} - \epsilon) \cdot d \cdot n$. Putting this together, we get

$$(\frac{|X| \cdot |Y|}{n^2} - \epsilon) \cdot d \cdot n \leq \{\text{number of edges from } X \text{ to Y}\} \leq |X| \cdot (1 - \delta) \cdot d$$

This gives us the following bound on $|Y|$:

$$|Y| \leq (1 - \delta + \frac{\epsilon \cdot n}{|X|}) \cdot n$$

But $|X| \geq \beta \cdot n$, so we find that the relative distance must be $\geq 1 - \delta + \frac{\epsilon}{\beta}$. Therefore, AEL constructs an $(n, R \cdot k, \delta - \frac{\epsilon}{\beta})_{q^d}$-code. ■

The AEL construction answers our original question of how to construct a family of codes with (essentially) the same rate and relative distance as a given code C_0. Consider using the AEL construction with C having rate $1 - \tau$, i.e. $k = (1 - \tau) \cdot n$. Then, $\beta = \frac{D}{n}$ becomes fixed at $O(\tau)$. Now, pick $\epsilon = O(\tau^2)$, and $d = O(\frac{1}{\tau^4})$. A (d, ϵ)-uniform graph exists for this choice of d and ϵ, so the AEL construction gives us an $(n, R \cdot (1 - \tau) \cdot n, \delta \cdot (1 - \tau) \cdot n)_{q^{O(\frac{1}{\tau^4})}}$-code. Thus, we have lost only a $1 - \tau$ factor in the rate and relative distance, although the alphabet size is much larger.

4 Decoding AEL Codes

The natural way to decode AEL codes is to reverse the steps of the encoding procedure. That is, given an output message, we can travel backwards on the edges of G to get candidate codewords for each left vertex . Then, we use a decoding algorithm for the code C_0 to get the message associated with each left vertex. Once we have these values, we can use a decoding algorithm for C to get the original message.

Theorem 5 (Guruswami and Indyk) $\forall R, \delta$ *such that* $R + \delta < 1$, $\exists n_0 < \infty$ *such that* $\forall n > n_0, \exists q$*-ary codes of rate* R *and relative distance* $\geq \delta$ *that are uniqely decodable from* $\leq \frac{\delta}{2}$ *fraction of errors in linear time.*

Sketch of Proof

Consider the natural decoding method suggested above. Assume that our output has $\tau \cdot n$ errors. These errors get propagated along the edges of G back to the left vertices. We assume that there is an algorithm to decode C_0 in constant time. This means that for each left vertex we should be able to decode correctly, provided that fewer than $\frac{\delta \cdot d}{2}$ edges coming into this vertex are in error. Finally, we assume that there is an algorithm for decoding C in linear time, if there are $\leq \frac{\beta \cdot n}{2}$ errors.

Define X to be the set of left vertices of G that see too many errors to be decoded properly, and let Y be the set of uncorrputed right vertices at the output. Then, our decoding method will work provided that $|X| < \frac{\beta \cdot n}{2}$. We know that $|Y| = (1-\tau) \cdot n$. Since G is a (d, ϵ)-uniform graph, we must have at least $(\frac{|X| \cdot (1-\tau)}{n} - \epsilon) \cdot d \cdot n$ edges between X and Y. However, every vertex in X has at least $\frac{\delta}{2} \cdot d$ errors, which means that every vertex in X has at most $(1 - \frac{\delta}{2}) \cdot d$ neighbors in Y. Thus, we get the bounds

$$(\frac{|X| \cdot (1-\tau)}{n} - \epsilon) \cdot d \cdot n \leq \{\text{number of edges from } X \text{ to Y}\} \leq (1 - \frac{\delta}{2}) \cdot d \cdot |X|$$

This reduces to

$$|X| \leq \frac{\epsilon \cdot n}{\frac{\delta}{2} - \tau}$$

Thus, if we choose $\tau \leq \frac{\delta}{2} - \frac{2 \cdot \epsilon}{\beta}$ then $|X| \leq \frac{\beta}{2} \cdot n$. Therefore, we can decode from a fraction $\tau = \frac{\delta}{2} - \frac{2 \cdot \epsilon}{\beta}$ of errors in linear time by choosing C and C_0 appropriately.

5 Extractors

The motivation for (d, ϵ)-uniform graphs was that we wanted graphs that "look" random. If we look at a random d-regular bipartite graph and pick a random subset X of left vertices and a random subset Y of right vertices, we expect that $|X| \cdot \frac{d \cdot |Y|}{n} = \frac{|X| \cdot |Y|}{n^2} \cdot d \cdot n$ edges go between X and Y. Thus, (d, ϵ)-uniform graphs look random in the sense that any choice of subsets X, Y for these graphs comes close to meeting the expected number of edges between the two subsets.

Consider what happens if we choose a random subset of size $\frac{n}{2}$ from the left vertices of a bipartite graph. We can choose a random edge leaving this subset, and ask about the probability that this edge is incident to a particular right vertex. In particular, we can try to find graphs for which any right vertex is equally likely to be the endpoint of this randomly chosen edge. Such graphs are called extractors, and we will see that they are closely related to error-correcting codes and list-decoding.