6.897 Algorithmic Introduction to Coding Theory	November 28, 2001
Lecture 19	
Lecturer: Piotr Indyk	*Scribe: abhi shelat*

1 Expander-based codes

In this lecture we develop a new technique based on expander graphs to construct excelletnt error correcting codes. This novel construction, first observed by Alon, Bruck, Naor, Naor, and Roth in 1991 is a simple and elegant procedure that provides excellent distance amplification while keeping alphabet sizes small. Codes with state-of-the-art parameters in terms of rate and distance are construted using this technique.

The basic construction has two prerequisities:

1. An error correcting code C with parameters $[n, rn, \alpha n]_q$ where r is the rate of the code and α is the fractional distance of the code.

2. A bipartite graph $G = (A, B, E)$ where A is the set of left nodes, B is the set of right nodes, $|A| = |B| = n$, and the degree of each node in the graph is d.

Given these two objects, the construction works as follows. We line up the n positions of the code C with the left nodes of the bipartite graph. Each position of the code C corresponds to one node in A. For each codeword $c \in C$, we form a new codeword by distributing the symbols of x from left to right via the edges of the graph G. That is, for each node $b \in B$ on the right side of G, we form a symbol in the new code, C', by concatenating the symbols from the d left neighbors of b. The order in which the symbols are concatenated is arbitrary as long as it is pre-determined and fixed. Let $G(c)$ represent the new codeword formed by applying this construction to a codeword $c \in C$, and let $G(C) = \{G(c) | c \in C\}$ represent the new code. Note that $G(C)$ is defined over an alphabet of size q^d.

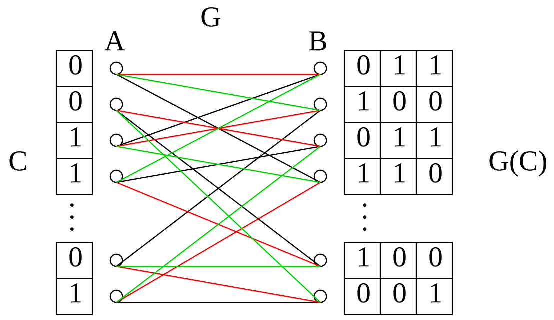

Figure 1: *Example of the construction. In this case, the graph G has degree 3. The symbol, 0, is copied to each of the nodes that are neighbors of the first position.*

In order for this construction to be interesting, there are additional properties required of the bipartite graph G. First, let us introduce the notion of an expander-graph.

Definition 1 *A bipartite graph G is an (α, β) expander if for any set $X \subset A$ of size $|X| \geq \alpha n$, the set of right neighbors of X denoted $\Gamma(X)$ has size $|\Gamma(X)| \geq \beta n$.*

If we take G to be an (α, β) expander, then our code construction from above amplifies the distance of the original code. In particular, the distance of $G(C)$ is at least βn. In order to see this, consider two codewords c_1 and c_2 from C. Since C has distance αn, the set of positions in which these two codewords differ must be at least αn. By the expander property of G, the set of right neighbors of these differing positions must have size at least βn which implies that at least βn symbols of the resulting codewords differ. Hence, the code $G(C)$ has distance at least βn.

As an example of this construction, suppose C is a rate $1/2$ code with fractional distance α. An example of such a code is a Justusen code as presented in Lecture ?. It is well-known that one can construct $(\alpha, 1-\epsilon)$ expander graphs that have degree $d = O(\frac{1}{\epsilon})$. Therefore, by following our construction, one can achieve codes with rate $\frac{1}{2} \times \frac{1}{d} = \Omega(\epsilon)$ and fractional distance $(1-\epsilon)$. More importantly, the size of the alphabet is $2^{O(1/\epsilon)}$ which is large, but constant with respect to the size of the block length.

The only codes with similar parameters are Algebraic codes which achieve a rate of at least

$$r \geq \epsilon - \frac{1}{\sqrt{\text{alphabet size}}}.$$

The rate is only slightly better than $\Omega(\epsilon)$ (the constant that hides in the Ω notation is small). However, AG codes are notoriously complicated to use (or even understand). Note that Random codes achieve a rate of at least $r \geq \epsilon - 1/\log(\text{alphabet size})$.

Up to this point, we have not focused on how to decode the expander-based codes. One can categorize decoding into two general categories: unique decoding, which outputs a single message for each received codeword, and list decoding which outputs a small list of possible messages for each received codeword. One of the important contributions of the Guruswami+Indyk paper is that they present decoding algorithms that allow recovery from a large fraction of errors. In particular, they show that if the following two conditions are met, then unique decoding can be achieved for error rates of up to $\frac{1-\epsilon}{2}$. The two required properties are :

1. C is decodable if there are less than $\alpha/2$ errors.

2. G is an (α, β) *mixer.*

Definition 2 *A graph G is an (α, β) mixer if for all subsets $Y \subset B$ such that the size of Y is at least $(1/2+\epsilon)n$, the fraction of nodes on the left side of G which have less than $d/2$ neighbors in Y is at most α.*

In Lemma 1, we will prove that such graphs exists using the probabilistic method. When G has the additional property of being a mixer, then we can use the intuitive majority voting function to efficiently decode symbol by symbol. That is, in order to determine the symbol corresponding to a left node, we take the majority of the set of symbols that are the right neighbors of this left node. We perform this voting for all left-hander symbols and then use the decoding algorithm for the C code to find the original message. The mixing property guarentees accurate decoding since if there are less than $1/2+\epsilon$ errors, then there are at least $(1/2+\epsilon)n$ uncorrupted symbols on the right side of G. Therefore, there are at most αn symbols on the left side of G which have less than $d/2$ neighbors in the uncorrupted set of symbols which implies that the result of the majority voting produces a codeword in C which has less than α errors. Therefore the decoding algorithm for C guarentees correct decoding.

In the rest of the lecture, we will apply this construction to different classes of codes to achieve novel parameters for interesting classes of codes.

In the first example, suppose we take C to be a Spielman code. From the previous lecture, we know that these codes are linear-time encodeable and decodeable codes which can handle a few errors. In practice, these codes can correctly decode if there are less than 10^{-7} fraction of errors. By using an appropriate mixer graph, one can achieve a rate $\Omega(1/d)$ rate code that can handle $1/2 - \epsilon$ errors.

Further improvements allow one to achieve optimal trade-offs between rate and distance. If C is a list-decodeable code, then the resulting expander-based code is also list-decodeable. Following a novel list-decoding method, one can achieve linear-time encoding and near-linear time decoding.

Lemma 1 *There exist (α, ϵ) mixer graphs with degree $d = O(1/\epsilon^2)$.*

Proof The proof uses the probabilistic method. We first pick a random bipartite graph with n left and right nodes in the following way: for each left node, pick d random right nodes and make them neighbors. Note that this approach might lead to irregular degrees in the right node, but this is a detail that we currently ignore (a better approach to this kind of sampling is to pick d permutations from S_n and generate a graph by connecting these permutations between the left and right nodes, but this makes the analysis much trickier). We would like to determine the probability that there is a set of right nodes Y of size $(1/2+\epsilon)n$ such that every subset X of left nodes of size $\geq \alpha n$ has at least $d/2$ neighbors in Y. In order to do this, we will instead find the probability of the converse, i.e, the probability that there exists of set X which contains a node that does not have $d/2$ neighbors in Y. We affectionately refer to this case as the *bad* case, and the former as the *good* one.

Suppose X and Y are fixed. The probability that a fixed node in X has $\leq d/2$ neighbors in Y is less than $e^{-a\epsilon^2 d}$ for some constant a [NOTE:expand this argument using Chernoff]. Since there are αn nodes in X, the probability that any node in X has $\leq d/2$ neighbots in Y is therefore $e^{-a\epsilon^2 d\alpha n}$. Since there are at most 2^n possible choices for X and 2^n possible choices for Y we have

$$\Pr[Y \subset B, |Y| \geq (1/2+\epsilon)n, |X| \geq \alpha n] \leq 2^n 2^n e^{-a\epsilon^2 d\alpha n} = e^{-2a\epsilon^2 d\alpha}$$

which is less than 1 when $d \sim 1/\epsilon^2$. Since this probability is less than 1, the probability of the *good* event is positive, and therefore, such a graph exists. ■