6.897 Algorithmic Introduction to Coding Theory	November 14, 2001
Lecture 16	
Lecturer: Madhu Sudan	*Scribe: Chris Peikert*

Today's material is on:

- Low Density Parity Check (LDPC) Codes
- Linear-time Decodable Codes

The goal is to have very simple decoding algorithms. Recall that a linear code is specified by an $n \times (n-k)$ matrix H such that C is a codeword iff $CH = 0$. Gallager's observation about linear codes was that the parity check matrix gives some additional information about where the errors occurred. Suppose $(CH)_j = 1$; this indicates that the jth constraint is not satisfied, so some bit of C corresponding to a 1 in the jth column of H is wrong. Then if H has very few 1s in all of its columns, we get a lot of information about where an error lies in C. Gallager showed, by the probabilistic method, that one can approach the Gilbert-Varshamov bound on the distance of such a code by randomly placing a constant number of 1s per column. **Exercise: reconstruct Gallager's proof (don't try this alone!).**

Sipser and Spielman '97 show how to choose H constructively, and argue the distance of its associated code combinatorically. They view H as the adjacency matrix of a bipartite graph $B = (L, R, E)$ where $|L| = n$, $|R| = m = n - k$. The left vertices L represent the coordinates of codewords, and the right vertices R represent the constraints. A constraint is "satisfied" if the exclusive-or of its adjacent codeword coordinates is 0 (this corresponds exactly to matrix multiplication in GF_2). Then the number of 1s in each column of H is small iff the degrees of the right vertices is bounded from above. We will require an "almost random" structure to the graph; this is known as the "expander" property.

Definition 1 (bounded, regular, expander) *A bipartite graph $B = (L, R, E)$, $|L| = n$ is:*

- (c, d)-bounded *if its left vertices have degree at most c and its right vertices have degree at most d,*
- (c, d)-regular *if equality holds above,*
- *an* (α, δ)-expander *if $\forall S \subseteq L$, $|S| \leq \delta n$, the neighborhood of S, $\Gamma(S)$, is large: $\Gamma(S) \geq \alpha|S|$.*

Define C_B, the linear code derived from the graph B, naturally. Observe that 0 is a codeword of C_B, and that the rate of $C_B = k/n \geq (n-m)/n$ (the inequality follows from the fact that some constraints may be linearly dependent, which can only increase the rate). In particular, if B is (c, d)-regular, then $cn = dm \Rightarrow m/n = c/d \Rightarrow$ rate $\geq 1 - c/d$. The distance of C_B is given by the following lemma:

Lemma 2 *If B is an (α, δ)-expander and is (c, d)-bounded, then the distance of C_B is at least δn, provided $2\alpha > c$.*

Before proving the lemma, let's consider its consequences. In order for it to be useful, α will need to be very large. By boundedness of degrees, we must have $|\Gamma(S)| \leq c|S|$; the lemma requires that $|\Gamma(S)| \geq c|S|/2$. Fortunately, such expanders do exist.

Now we'll sketch a proof of Lemma 2, and fill in the holes as they appear. For any $S \subseteq L$, partition $\Gamma(S)$ into sets $A, B \subseteq R$, such that A contains the vertices of $\Gamma(S)$ which have exactly 1 neighbor in S, and B is the remainder of $\Gamma(S)$. Because C_B is linear, it is sufficient to show that it contains no codeword with Hamming weight $< \delta n$. Let S be the coordinates at which a vector is set to 1. We will show that if a vector has fewer than δn 1s, some constraint is unsatisfied. It will be sufficient to prove that $A \neq \emptyset$ when we fix any $|S| < \delta n$, because every constraint in A is unsatisfied.

Lemma 3 *If $B = (L, R, E)$ is a (c, d)-bounded, (α, δ)-expander, then for any $S \subseteq L$, $|S| < \delta n$, and A defined above, $|A| \geq (2\alpha - c)|S|$.*

Proof of Lemma 3 First, $\Gamma(S) = |A| + |B| > \alpha|S|$ by the expander property. Let e be the number of edges incident to S. Then we have $|A| + 2|B| \leq e \leq c|S|$. By multiplying and subtraction, we get $|A| \geq (2\alpha - c)|S|$, as desired. ■

We now give a simple, linear-time decoding algorithm for this code: *while there exists a coordinate such that more than half of its neighbors are unsatisfied constraints, flip that coordinate.*

To prove the correctness of this algorithm, it is enough to analyze how it decodes a corrupted zero codeword. It is clear that the algorithm requires no more than $m = n - k$ iterations, because the number of satisfied constraints decreases with each iteration. From now on assume that $\alpha > 3c/4$, and say the input vector has $\leq \epsilon n$ 1s. We prove that for suitable ϵ, the algorithm will terminate with the zero word.

Lemma 4 *If $0 < |S| \leq \delta n$, then there is some $j \in S$ such that j has more than $c/2$ of its neighbors in A.*

Lemma 5 *The number of 1s in the vector is always $\leq \delta n$ (for suitably chosen ϵ, to be determined).*

¿From these two claims, correctness follows: $|S| > 0$ implies that the algorithm has not terminated. And because $|S| \leq \delta n$ at all times, the algorithm terminates with $|S| = 0$. We now prove the lemmas.

Proof of Lemma 4 By Lemma 3, $|A| > (3c/2 - c)|S| \geq c|S|/2$ so some $j \in S$ must have at least $c/2$ neighbors in A. ■

Proof of Lemma 5 At the start, the number of unsatisfied constraints is $\leq c\epsilon n$. So $|A| \leq c\epsilon n$ throughout the algorithm. Then by Lemma 3 we have

$$|S| \leq \frac{|A|}{2\alpha - c} \leq \frac{c\epsilon}{2\alpha - c} n$$

so $|S| \leq \delta n \Leftarrow \epsilon = \delta(2\alpha - c)/c$. ■

Note: we have proven a linear bound on the number of iterations. In fact, with suitable data structures, each iteration can be made to run in $O(1)$ time.

Caveats:

- There is no known way of *encoding* other than by matrix multiplication, in $O(n^2)$ time.
- There are no known deterministic constructions (at press time...) of expanders with a good enough expansion factor ($\alpha > 3c/4$). But even so, this is a nonconstructive code with an explicit decoding algorithm, which is good.
- Variants of this algorithm work, requiring a smaller expansion factor.

To make improvements, Lemma 3 is the bottleneck; we want to reduce the ratio. We would like to define A such that its elements are those "constraints" (however defined) for which at most Δ of its associated coordinates are "wrong." But we can just view this as an error-correcting code having distance Δ. Then each constraint is "my neighbors should make a codeword of some other code." With this restriction, we get $|A| \geq ((\Delta + 1)\alpha - c)|S|$, and the analysis follows from there.