6.440 Essential Coding Theory April 3 2013

Lecture 15

Lecturer: Madhu Sudan *Scribe: Zonghao Chen*

1 Introduction

Graph-based codes were introduced in the previous lecture. Definitions as well as important properties of those codes were presented with explicit decoding algorithms left over to this lecture.As mentioned previously, one of the main topics in the near future would be linear time algorithms. A linear time decoding algorithm of the Expander Codes will be discussed in the following part.
As the encoding process may take quadratic time for these graph-based codes, we will continue to explore codes that are linear time encodable and decodable. Spielman's family of codes will be presented in the second half of this scribe notes.

2 Linear Time Decoding Algorithm

2.1 Review

In the previous lecture, we discussed the expander codes. Before presenting the decoding algorithm, we may first review some of the definitions and lemmas.

Definition 1 *Given a subset of vertices $S \subseteq L$, its* **neighborhood** $\Gamma(S)$ *is the set of vertices in R which are adjacent to at least one vertex in S; that is,*

$$\Gamma(S) = \{v \in R | \exists u \in S \ s.t. \ (u, v) \in E\}.$$

Definition 2 *A graph is a (γ, δ)-**expander** if for all $S \subseteq L$ such that $|S| < \delta n$, we have*

$$|\Gamma(S)| \geq \gamma |S|.$$

Definition 3 *A graph is a (γ, δ)-**unique-expander** if for all $S \subseteq L$ such that $|S| < \delta n$, we have*

$$|\Gamma^{unique}(S)| \geq \gamma |S|.$$

Lemma 4 *If G is a (c, d)-regular (γ, δ)-expander, then G is a $(2\gamma - c, \delta)$-unique expander.*

Now we are ready to present the decoding algorithms of this code. Recall that when r is the received vector and H is the parity check matrix, the error can be determined from $r \cdot H$. In particular, if the jth bit of $r \cdot H$ is non-zero, an error must have occured at an index i such that $H_{ij} \neq 0$. Then we can consider performing decoding by flipping r_i. A vertex in R is called satisfied if it has an even number of non-zero neighbors.

2.2 Decoding Algorithm

If there exists a vertex r_i in L with more unsatisfied neighbors than satisfied neighbors, flip r_i.
To prove that this flipping algorothm leads to the desired result, we need to show the following:

1. This algorithm does terminate.
2. When the algorithm stops, we end in a codeword.
3. When the algorithm stops and we get a codeword $\widetilde{c}$, then $\widetilde{c} = c$.

Thus we need to show that the above three rules can be satisfied as long as the number of errors is small enough.Given a received vector x $(with \exists c \in C \ s.t. \delta(x,c) \leq \tau n < \frac{\delta}{c+1} n)$

Lemma 5 *The number of iteration of this decoding algorithm* $\leq \tau cn$

Corollary 6 $\delta(c, \widetilde{c}) \leq \tau(c+1)n < \delta n$

Proof Each bit error that occured can lead to at most c unsatisfied vertices on the right. Then the total number of unsatisfied vertices on the right should be $\leq \tau n$. Since each iteration can reduce the number of unsatisfied vertices by at least one, the algorithm terminates in at most τcn iterations. ■

Theorem 7 *If graph G is (c, d)regular and (r, s) expander with $\gamma > \frac{3}{4}c$, then the associated code has distance at least δ and can correct at least $\frac{\delta}{c+1}$fraction of errors.*

Proof We have shown that the algorithm does terminates and since $\delta(c, \widetilde{c}) < \delta n$, we will get the correct codeword if we do end with a codeword. All that remains to be proved is that the result must be a codeword. To argue that $\widetilde{c}$ must equals c, let $\widetilde{c} + c = y$ and note that y is a vector of weight less thanδn. Furthermore, since c is a codeword and hence every node on the right is satisfied, assigning $\widetilde{c}$ to the vertices on the left will induce the same assignments on the right as assigning y to the left.
Therefore, in order to argue that the Flip Algorithm would not terminate with the vertices on the left assigned to $\widetilde{c} \neq c$, we argue that many of the vertices on the right would be non-zero. Since $\widetilde{c}$ induces the same assignment to the right as y does, we will argue this in terms of y.
Let S be the set of non-zero left side vertices under an assignment of y, then$|S| < \delta n$. Therefore, since G is (γ, δ)-expander, $|\Gamma^{unique}(S)| > (2\gamma - c)|S| > 0$. Thus there must exist a vertex in S that is unsatisfied which contradicts the assumption that the Flip Algorithm has terminated. ■

3 Spielman's code

Though linear time decoding is possible for the expander codes as proved in the previous section, the encoding process may take quadratic time. Similar to the idea adopted above, the encoding can be faster if the generator matrix is sparse.Spielman's family of codes can be encoded and decoded in linear time, and will be presented in the following notes.

3.1 Encoding

Denote E_k the Spielman's code for message of length k. Code E_k will be described in terms of Spielman's codes for shorter message. Given a message $x \in \{0,1\}^k$, the corresponding codeword should be of length $4k$ the first k bits of which is the original message while the rest $3k$ are parity check bits. Define $G_k : \{0,1\}^k \to \{0,1\}^{k/2}$ a family of error reduction codes.
Given a message x of length k, the first $3k/2$ bits of encoding are the original message and the result of applying G_k to x (denoted u). Then the substring u is encoded with $E_{k/2}$ to generate the next $3k/2$ check bits (denoted v). The last k bits (denoted w) are generated by applying G_{2k} to (u, v).
Now we claim that this encoding procedure can be implemented in linear time. Assume the generator of G_l has $\leq cl \ 1's \ \forall \ l$, then

$$T_E(k) \leq ck + c(2k) + T_E(k/2)$$

This implies that $T_E(k) \leq 6ck \ = O(k)$

3.2 Decoding

Assume the original word is (x, u, v, w) and $(\widetilde{x}, \widetilde{u}, \widetilde{v}, \widetilde{w})$ is received. If the number of errors is small enough, we can decode with linear time algorithms. The decoding of this code will be implemented in several steps:

1. Error Reduction. Run the Flip Algorithm on $(\widetilde{u}, \widetilde{v}, \widetilde{w})$ until it teminates. Denote the result (u', v').
2. Decoding of $E_{k/2}$. Decode (u', v') with $E_{k/2}$ recursively to obtain a new vector u''.
3. Error Reduction. Run the Flip Algorithm on $(\widetilde{x}, u'')$ and obtain the recovered message x'.

Assume that the number of errors is at most τk. Then $\delta((u, v), (\widetilde{u}, \widetilde{v}) \leq \tau k$ and $\delta(w, \widetilde{w}) \leq \tau k$.After the error reduction in step 1, we have $\delta((u, v), (u', v')) \leq \frac{\tau k}{2}$. By induction, we may conclude that the step 2 can correct all errors in u, which means $u = u''$. Thus the error reduction in step 3 can recover x without error.
As for the running time of the decoding algorithm. From the fact that error reduction can be done in $O(k)$, we have

$$T_D(k) \leq O(k) + T_D(\frac{k}{2}) + O(k)$$

Which implies that $T_D(k) = O(k)$