# 1 Review: Tanner Codes

In the previous lecture we showed how to obtain codes by using properties of pseudorandom graphs. The construction we obtained is called Tanner codes [SS96], and we analyzed their distance and rate.

Today we will describe an algorithm to **decode Tanner codes** [Zem01]. Additionally, we saw last time that Tanner codes can only exist for small (constant) relative distances. In the second part of this lecture we will also describe a **distance amplification procedure** by [ABN+92].

We start by reviewing Tanner codes. Consider a bipartite expander graph $H = (L, R, E)$, with partitions $L, R$ each having $n$ vertices. We additionally require this graph to be $r$-biregular (each vertex has degree exactly $r$) and $\epsilon$-pseudorandom.

To define a Tanner code, we will assign each edge of $H$ to a bit of the codeword $C$. Each one of the incident edges to a vertex belongs to some local code $C_0$. Formally, we define the Tanner code as

$$X(H, C_0) = \{x \in \{0,1\}^E \mid \forall v \in V(H), C_{|E(v)} \in C_0\}$$

Of course, this definition only makes sense syntactically if $H$ is $r$-biregular.

Assume $C_0$ is a binary linear code. Last class we proved that

**Theorem** (Informal)**.** *If $H$ is $\epsilon$-pseudorandom and $\delta(C_0) = \delta$ then*

$$\delta(X(H, C_0)) \geq \delta_0(\delta_0 - \epsilon)$$

Recall the bipartite version of $\epsilon$-pseudorandomness.

**Definition 9.1** (Bipartite version of $\epsilon$-pseudorandomness)**.** An $r$-biregular bipartite graph $H = (L, R, E)$ is $\epsilon$-pseudorandom if for all $S \subseteq L, T \subseteq R$

$$\left| |E(S,T)| - \frac{r|S||T|}{n} \right| \leq \epsilon r \sqrt{|S||T|}$$

We also saw last time that pseudorandomness is intimately connected to the spectral expansion of the graph. In particular, we stated the following lemma (without proof).

**Lemma 9.2.** *If $H$ is the double cover of $\lambda$-spectral expander, $H$ is $\frac{\lambda}{r}$-pseudorandom.*

**Definition 9.3** (Double cover)**.** The double cover of a graph $G = (V, E)$ is the bipartite graph $H = (L, R, E')$ defined as follows. Let the vertices of $G$ be $V = \{u^{(1)}, \ldots, u^{(|V|)}\}$. Then $L, R$ are two distinct copies of $V$

$$L = \{u_L^{(1)}, \ldots, u_L^{(|V|)}\}, \quad R = \{u_R^{(1)}, \ldots, u_R^{(|V|)}\}$$

For each edge $(u^{(i)}, u^{(j)})$ of $G$ we add the edges $(u_L^{(i)}, u_R^{(j)})$ and $(u_L^{(j)}, u_R^{(i)})$ to $H$.

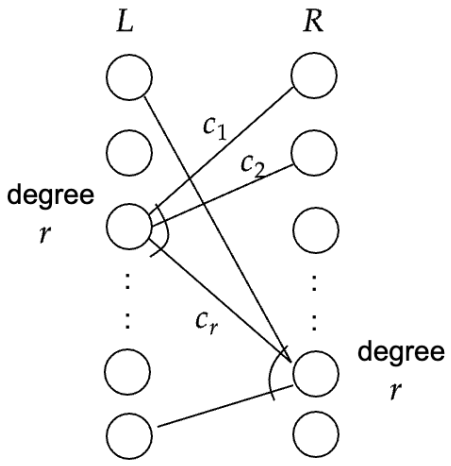


Figure 9.1: An example of an $r$-regular bipartite expander $H = (L, R, E)$. The edges correspond to codeword bits and all edges incident to a vertex correspond to the local code $C_0$. In the graph above $c = (c_1, c_2, \ldots, c_r) \in C_0$.

The reason Tanner codes are great, is because we have explicit constructions of $r$-regular graphs that are very close to pseudorandom.

**Lemma 9.4.** *There are explicit (near-Ramanujan) $\lambda$-spectral expanders with $\lambda \leq O(\sqrt{r})$*

To construct good Tanner codes, we do not even need a graph with optimal spectral expansion. For example, a near-Ramanujan graph of degree $O(1/\epsilon^2)$ will be $\epsilon$-pseudorandom. Then we can find a good constant-size local code $C_0$ by brute force. The following lemma lower bounds the rate of the resulting Tanner code with respect to the rate of the local code $C_0$.

**Lemma 9.5.** *The rate of a Tanner code $X(H, C_0)$ satifies*

$$R(X(H, C_0)) \geq 2R(C_0) - 1$$

We proved the above statement last class by counting the number of constraints in the Tanner code (each vertex imposes one local-code constraint). So if our chosen local code meets the GV bound

$$R(C_0) \approx 1 - h(\delta_0)$$

$$\Rightarrow R(X(H, C_0)) \approx 1 - 2h(\delta_0) \approx 1 - 2h(\sqrt{\delta})$$

An important remark is that the rate is only positive when $\delta \leq \left[h^{-1}(1/2)\right]^2 \approx (0.11)^2$, so this construction only works for local codes with small constant distance and the resulting Tanner code will also have small (constant) distance. Subsequent work has improved this constraint.

## 2 Decoding Tanner codes

In this section we will see how to decode Tanner codes. The algorithm we will see is due to Zemor [Zem01], who proposed the following algorithm that can decode errors up to a quarter fraction of

the distance. The previous algorithm by Sipser and Spielman [SS96] could only decode a lower fraction of errors.

The intuition is similar to the decoding of concatenated codes. We try to decode each code *locally* first. In particular, each node will try to locally fix its corresponding bits to the nearest codeword in $C_0$.

But what if the two nodes that share an edge do not agree on whether to flip the bit of this edge? We will solve this problem by considering the **double cover** of our graph and have the nodes in $L$ alternate with the nodes in $R$ in fixing their incident edges. The double cover graph will not create local conflicts with the vertices, since it is bipartite.

**Algorithm Intuition.**

- **Input.**
  - $\epsilon$-pseudorandom double cover graph $H = (L, R, E)$
  - Local code $C_0$
  - Tanner code $X(H, C_0)$
  - Noisy codeword $y \in \mathbb{F}_2^E$ (with $\leq e$ errors)
- **Left-side decoding.** For all $u \in L$

$$y_{|E(u))} \leftarrow \text{ closest codeword in } C_0 \text{ to } y_{|E(u))}$$

  - Break ties arbitrarily
  - The above operation can be done in parallel, since the nodes in $L$ do not share edges
- **Right-side decoding.** Repeat the above for all $v \in R$

It is possible that the right-side decoding procedure may undo some of the corrections from the left-side decoding. Regardless, once decoding is complete the algorithm will stabilize.

**Algorithm.**

1. For $O(\log n)$ rounds
   (a) Perform left-side decoding
   (b) Perform right-side decoding
2. If the algorithm has not converged to a codeword, declare failure
3. Otherwise return $y$

**Theorem 9.6.** *If $H$ is $\epsilon$-pseudorandom and $\delta(C_0) = \delta_0$, for any $\gamma > 0$, then the above algorithm corrects $(1-\gamma)\frac{\delta_0}{2}\left(\frac{\delta_0}{2} - \epsilon\right)$ fraction of errors in $O_\gamma(\log n) = A(\gamma)\log n$ rounds.*

**Comments.**

- A direct implementation of the above algorithm will run in $O(n \log n)$ time, since local corrections are done in linear time. A more careful implementation can improve the runtime to $O(n)$. Additionally, the algorithm is highly parallelizable; it takes $O(\log n)$ time to run in parallel, and it can be computed using a log-depth circuit.

- The theorem above states that the above algorithm can correct up to 1/4 of the distance bound. As we saw with concatenated codes, this can be improved.
- This algorithm has many nice properties: it is fast in practice, easy to implement, and it correct more errors than we can theoretically prove in many cases.

*Proof Sketch.* Since the number of errors is low enough, there exists a unique codeword $c^* \in X(H, C_0)$ that we want to decode to. Our goal is to define a 'potential function' that will quantify our progress towards $c^*$. A natural first choice for this function is the number of errors remaining.

Instead, we will track the number of 'unfixed nodes' after each iteration, i.e. the nodes whose neighborhood is not a codeword, formally

$$v \in L \cup R \text{ is unfixed if } y_{|E(v)} \notin c^*_{|E(v)}$$

Define $S_i$ to be the number of unfixed nodes in $L$ right after left-decoding during round $i$ for $i \in \{1, 2, \ldots, O(\log n)\}$.

$$S_i = \left\{ u \in L \mid y_{|E(u)} \neq c^*_{|E(u)} \text{ after round } i \text{ of left-side decoding} \right\}$$

Analogously, define $T_i$ to be the number of unfixed nodes in $R$ right after right-decoding during round $i$ for $i \in \{1, 2, \ldots, O(\log n)\}$.

$$T_i = \left\{ v \in R \mid y_{|E(v)} \neq c^*_{|Evu)} \text{ after round } i \text{ of right-side decoding} \right\}$$

Note that once one of $S_i, T_i$ becomes empty, the other will also become empty and the algorithm will converge. Let $e \coloneqq \frac{\delta_0}{2}\left(\frac{\delta_0}{2} - \epsilon\right)(1-\gamma)nr$ be the bound on the number of errors.

**Claim 9.7.** *The number of unfixed nodes after the first left-side decoding cannot be too large*

$$|S_1| \leq \frac{e}{\frac{\delta_0 r}{2}} = \left(\frac{\delta}{2} - \epsilon\right)(1-\gamma)n$$

*Proof.* Each $u \in S_1$ contributes at least $\frac{\delta_0 r}{2}$ errors in $y$ with respect to $c^*$.

Note that this is a counting argument that doesn't make use of the pseudorandomness of the underlying graph. □

**Observation 9.8.** Consider the situation after the right-side decoding and in particular the erroneous edges $y_e \neq c^*_e$. Color them red. If we still have red edges, this means that the nodes in $T_1$ had a lot of incident red edges before right-side decoding (in the opposite case, the local decoding would eliminate any small number of reds).

In particular, every node in $T_1$ must have had at least $\frac{\delta_0 r}{2}$ errors (red edges) prior to the right-side decoding step. Of course, these edges can only come from vertices in $S_1$. Thus

$$|E(S_1, T_1)| \geq \frac{\delta_0 r}{2}|T_1|$$

*Intuition.* $\epsilon$-pseudorandomness implies that the edges of the graph are 'spread-out', meaning not too many edges can go from subset $S_1$ to $T_1$. Hence the above inequality cannot hold for sufficiently large $|T_1|$.

**Fact 9.9.** *Consider the red edges before right-side decoding and after the first left-side decoding*

- *All of them touch* $S_1$
- *Every* $v \in T_1$ *touches at least* $\frac{\delta_0 r}{2}$ *red edges*

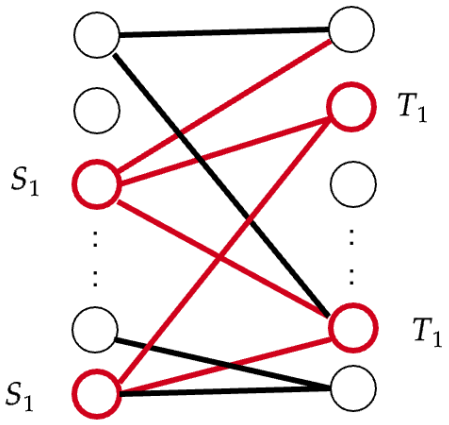


Figure 9.2: Proof-by-picture for Fact 9.9. Consider the graph above after the first left-side decoding and before the right-side decoding. All red edges must touch a vertex in $S_1$. Any vertex in $T_1$ must have a large number of red edges, otherwise locally decoding to the nearest codeword of $C_0$ would fix all the errors.

The $\epsilon$-pseudorandomness property implies that

$$\frac{\delta_0 r|T_1|}{2} \leq |E(S_1, T_1)| \leq \frac{r|S_1||T_1|}{n} + \epsilon r\sqrt{|S_1||T_1|}$$

We will use the fact that $|S_1|$ is small to bound the size of $|T_1|$

$$\frac{\delta_0 r|T_1|}{2} \leq r\left(\frac{\delta_0}{2} - \epsilon\right)(1-\gamma)|T_1| + \frac{\epsilon r}{2}\left(|S_1| + |T_1|\right)$$

$$\Rightarrow |T_1| \leq \frac{\epsilon/2 \cdot |S_1|}{\frac{\delta_0}{2} - \left(\frac{\delta_0}{2} - \epsilon\right)(1-\gamma) - \frac{\epsilon}{2}} = \frac{\epsilon/2 \cdot |S_1|}{\frac{\epsilon}{2} + \gamma\left(\frac{\delta_0}{2} - \epsilon\right)} < \frac{|S_1|}{\left(1 + \gamma\left(\frac{\delta_0}{2\epsilon} - 1\right)\right)}$$

So choosing a small enough $\epsilon$ makes the size of $T_1$ decay geometrically. The argument can be repeated for the size of $S_2$ using $|T_1|$. In general

$$|T_i| \leq \frac{1}{1 + O(\gamma)}|S_i| \qquad |S_{i+1}| \leq \frac{1}{1 + O(\gamma)}|T_i|$$

As a result, after a logarithmic number of rounds we will reach a set $S_{O(\log n)}$ or $T_{O(\log n)}$, whose size is less than 1. This means that this set is empty, thus our algorithm has converged to $c^*$. □

**Comment about running time.** The above algorithm can be implemented in $O(n)$ time. This is because the number of nodes whose edges need fixing during round $i$ depends on the set of

unfixed nodes from round $i-1$. As an example, during the first right-side decoding, we only need to consider the neighbors of the vertices in $S_1$, which are at most $r|S_1| = O(|S_1|)$. Since the size of these sets decreases geometrically, the total work of the algorithm amortizes to $O(n)$.

To motivate the next section, we recall the positive characteristics of this algorithm. It is simple, efficient (in theory and in practice). However, the relative distance, although constant, must satisfy $\delta < (0.11)^2$. This means that we can only correct a tiny fraction of errors. Compare this with serial concatenation, which allowed us correct up to the Zyablov bound [Zya71].

In the next section, we will see a way to amplify the distance of our code, by trading-off some of our rate.

# 3 Distance Amplification

## 3.1 Motivation

We will be seeing the ABBNR transformation [ABN+92]. Say we already have a code $C \subseteq \mathbb{F}_2^n$, with positive rate $R(C) > 0$ and relative distance $\delta(C) > 0$. Assume additionally that this code is linear-time decodable (e.g. a Tanner code).

**Question.** Can we improve our distance to correct a higher fraction of errors, e.g. can we get $\delta(C) \to \frac{1}{2}$?

Recall that the Zyablov bound [Zya71] already showed us how to do this. However the codes we are considering now are nicer, for example they have linear-time decoding. We will see a way to amplify the distance of the 'nice' codes we constructed in the previous section.

## 3.2 Distance Amplification

**Setup.** We are given a bipartite graph $G = (L, R, E)$ that is $r$-biregular (actually $r$-right regular suffices) and $\epsilon$-pseudorandom. Additionally, given a code $C$ (not necessarily constant size - it could be a Tanner code), we will define a new code $G(C)$ that will have larger relative distance.

Assign the bits of the code $C$ to the left subset of vertices $L$. Each vertex of the right subset $R$ will be assigned the bits of its $r$ incident vertices from $L$ and will output them as a tuple. In particular, the vertices of $R$ will be assigned a symbol in an alphabet of size $2^r$.

**Definition 9.10** (Distance Amplification Code)**.** Given a binary code $C$ and an $r$-biregular bipartite graph $G = (L, R, E)$, we define a new code $G(C) \subseteq \Sigma^n$ over alphabet $\Sigma = \{0,1\}^r$ and codewords $G(C) = \{G(c) \mid c \in C\}$. In particular, each codeword $G(c) \in \Sigma^n$ is constructed as follows

$$G(c)_j = \langle c_{N_1(j)}, \ldots, c_{N_r(j)} \rangle$$

Where $N_k(j)$ is the $k^{th}$ neighbor of the vertex $j$ in $R$ of $G$.

**Comments.** Note that this code is not linear (it is 'almost linear' in some sense) and it is also over a larger alphabet.

**Question.** What is the rate of this code? We can think of $G(C)$ as increasing the redundancy of code $C$ by repeating a symbol $r$ times, so $R(G(C)) = \frac{R(C)}{r}$. This is not a big problem, since $C$ has small distance, which implies its rate is larger.

We will now study the distance of $G(C)$. We will prove in the following lemma that the distance of $G(C)$ can be arbitrarily large.

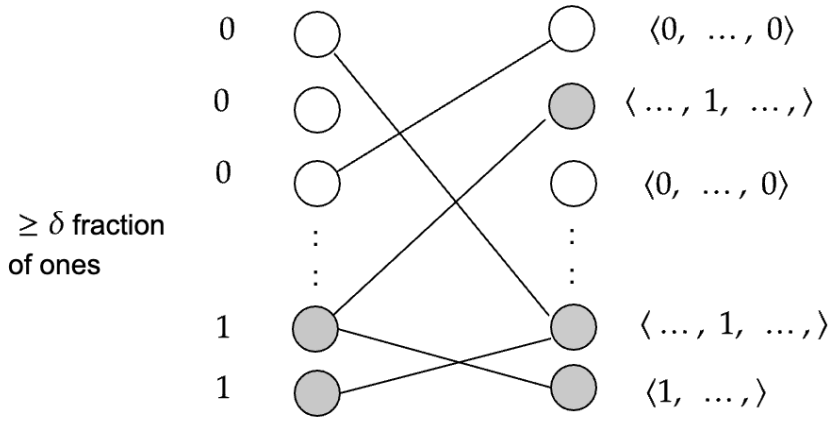


Figure 9.3: Schematic representation of the ABNNR distance amplification construction [ABN+92]. Given a code $C$, we use the bipartite graph to assign the bits of a codeword $c$ of $C$ to a codeword of a code $G(C)$ with an alphabet of size $2^r$. The psuedorandomness property of the underlying bipartite graph guarantees that the distance of $G(C)$ is amplified.

**Lemma 9.11.** *If $\delta(C) = \delta$ and $G$ is $\sqrt{\gamma\delta}$-pseudorandom then $\delta(G(C)) \geq 1 - \gamma$.*

*Proof.* First observe that the zero codeword $\mathbf{0} \in C$ is transformed to the zero codeword (all-zero tuples) in $G(C)$. Take any non-zero codeword $c$ of $C$ and embed it in $L$. From the distance property of $C$, we know that at least a $\delta$ fraction of the nodes in $L$ are assigned the bit 1. In particular, let $S = \{i \mid c_i \neq 0\}$ be the support of $c$.

**Fact.** For all $j \in N(S) = \{j \in R \mid \exists i \in S : (i, j) \in E\}$, $G(c)_j \neq \langle 0, \ldots, 0 \rangle$

Thus the weight of the new code is the neighborhood $N(S)$ of these vertices, which is not too small by the pseudorandomness / expansion property.

Let $T = R \backslash N(S) \implies E(S, T) = \emptyset$. The pseudorandomness property implies

$$0 = |E(S,T)| \geq \frac{r|S||T|}{n} - \epsilon r\sqrt{|S||T|}$$

Hence

$$\frac{|S||T|}{n} \leq \epsilon\sqrt{|S||T|} \implies |T| \leq \frac{\epsilon^2 n^2}{|S|} \leq \frac{\epsilon^2 n}{\delta}$$

The above expression implies that $|T| \leq \gamma n$ if $\epsilon \leq \sqrt{\gamma\delta}$. If $|T| \leq \gamma n$, then each non-zero codeword $c \in C$ is mapped to a non-zero codeword of weight at least $(1 - \gamma)n$, which concludes that the distance of $G(C)$ is at least $1 - \gamma$. To be precise, this argument is assuming that $G(C)$ is linear (and distance equals the minimum weight of non-zero codeword). Since $G(C)$ is not necessarily linear, one would need to apply the same argument to pairs of codewords $c_1, c_2 \in C$ and their neighborhood. □

**Notes.**

- The above construction allows one to obtain a code whose distance is as arbitrarily close to 1.

- Choosing the degree of the graph to be $r = O\left(\frac{1}{\gamma\delta}\right)$ suffices to construct a $\frac{1}{\gamma\delta}$-pseudorandom.
- **Upshot.** Can have explicit codes of relative distance $1 - \gamma$, of rate $\Omega(\gamma)$, and of (constant) alphabet size $2^{O(1/\gamma)}$. Such a code is almost-as-good as Reed-Solomon (meets the singleton bound up to constant factors), with constant alphabet.
- **Another upshot.** We can use the linear decoding to also decode these codes in linear time.

## References


[ABN+92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 1992. 1, 3.1, 9.3

[SS96] Michael Sipser and Daniel Spielman. Expander codes. *IEEE Transactions on Information Theory*, 1996. 1, 2

[Zem01] Gilles Zemor. On expander codes. *IEEE Transactions on Information Theory*, 2001. 1, 2

[Zya71] Victor Zyablov. An estimate of the complexity of constructing binary linear cascade codes. *Problemy Peredachi Informatsii*, 1971. 2, 3.1