An Overview of Utilizing Expanders to enhance Error-Correcting Codes

Ijay Narang

Abstract

A code is a collection of elements in Σ^n with parameters distance and rate (which measure quality and efficiency). The trade-off between these two parameters can be quantified with the Gilbert-Varshamov (GV) bound, which is proved with a random code. Thus, a major open problem is to explicitly provide a construction for a code which meets the GV bound. Recent approaches have utilized expander graphs, graphs which is simultaneously sparse and highly connected, to build codes with better distance and rate. In this survey, we motivate the use of expander graphs in constructing codes by (1) proving a generalized form of the expander-mixing lemma, (2) analyzing the distance and rate of Expander Codes and Tanner Codes, (3) surveying the use of expanders in distance amplification (ABNNR transform), and (4) outlining Ta-Shma's breakthrough construction of almost optimal, ϵ-balanced codes.

1 Introduction

Consider the following problem: a sender A has to transmit a string of bits to some receiver B over a noisy channel. A natural solution to this problem is for the sender to introduce some redundancy so that the receiver B can decode the message despite some bits being incorrect. Indeed, this is precisely the function of an error-correcting code. An optimal code is one which maintains correctness despite many corrupted bits while still utilizing an efficient number of bits. These characteristics can be quantified by the distance (which measures the difference between code-words) and rate (the ratio between the length of a code-word and the message it encodes), which naturally oppose each other. Utilizing a greedy construction (random code), the Gilbert-Varshamov bound states that for any $\delta \in (0, \frac{1}{2}$, there exists a code with block length n and distance d with rate $1 - H(\delta) - o_n(1)$ where $H(X) = -\sum_{i=1}^{n} p_i \log(p_i)$ is the Shannon entropy function. Modern approaches towards providing an explicit construction for a code which meets the bound include code concatenation and de-randomization utilizing expanders.

This survey focuses on the latter approach, and provides an overview of the applications of expanders in coding theory. As we are building up towards Ta-Shma's construction, which is the code closest to meeting the GV bound, we primarily focus on how different applications of expanders affect the distance and rate of particular codes. After recalling fundamental properties of linear codes, we cover both the combinatorial and spectral properties of expander graphs and their explicit connection via the Expander Mixing Lemma. With these tools, we discuss the distance and rate of Expander codes and their generalization, Tanner Codes, both of which are linear. We then describe the ANNBR transform, which utilizes expanders to create a new code with better relative distance and a larger alphabet. Finally, we discuss methods of de-randomizing codes and the s-wide replacement product, which are the core of the Ta-Shma code.

2 Background

2.1 Coding Theory Preliminaries

In this section, we introduce the notation that we will utilize throughout this survey [1]. Let us first define what a code is. We define an error correcting code (or block code) as

$$\mathcal{C} = \{c : c \in \Sigma^n\}$$

where we define Σ to be a finite alphabet (typically a finite field) and n to be the length of the code (also referred to as block length of $\mathcal{C}$). Every c corresponds to an element in $\mathcal{C}$ called a code-word. Let us refer to the length of our alphabet as $|\Sigma| = q$, so if our alphabet is binary, then $q = 2$. If $|\Sigma| = q$ then we say that our code is q-ary. For every code $\mathcal{C}$, we associate an encoding map G which takes in a message and outputs the corresponding code-word. In the case of a linear codes, we introduce later, the encoding map G is a generating matrix G. Such map maps the a message m in the set of messages $\mathcal{M}$ to a code-word c in $\mathcal{C}$.

Every code is associated with some parameters which measure its quality and efficiency. The first parameter is its rate $R(C)$, which is intuitively a fraction that measures the redundancy. $R(C) = \frac{k}{n}$, where $k = \frac{log|\mathcal{C}|}{log|\sum|}$ is the dimension of $\mathcal{C}$. The second parameter is the (minimum) distance. We refer to the Hamming distance between two code-words c_1, c_2 as $\Delta(c_1, c_2)$, the number of bits that c_1 and c_2 differ. We can thus, define the distance of a code $\mathcal{C}$, as $\Delta(\mathcal{C})$, the minimum hamming distance between all pairs of distinct code-words in $\mathcal{C}$. More formally,

$$\Delta(\mathcal{C}) = \underset{c_1 \neq c_2 \in \mathcal{C}}{min} \Delta(c_1, c_2)$$

The relative distance of a code is the normalized distance $\delta(\mathcal{C}) = \frac{\Delta(\mathcal{C})}{n}$. Additionally, we define the hamming weight of a code-word $wt(c_i)$ as the number of non-zero bits in the code-word and we define the weight of a code, $wt(\mathcal{C})$, as the minimum hamming weight across all code-words in $\mathcal{C}$. The support S of a code word is the set of coordinates of nonzero entries of the code-word.

As part of our survey, we will focus on linear codes, a subclass of codes that is widely used. We say that a code $\mathcal{C}$ is linear if any linear combination of the code words $c_1, c_2, ..., c_k$, where $k = dim(\mathcal{C})$, is also a code-word. For notation purposes, we will use the notation $[n, k, d]_q$ code to refer to q-ary linear code, with block length n, dimension k, and minimum distance d. For a linear code $\mathcal{C}$ of dimension k, a matrix $G \in \mathbb{F}_q^{n\times k}$, where $\mathbb{F}_q$ is the finite field of length q, is a generator matrix for $\mathcal{C}$ if its k columns span $\mathcal{C}$. That is to say, a linear code $\mathcal{C}$ can be viewed as the image $\text{Im}(G)$ of its generator matrix. For a message m, we can write a code-word $c \in \mathcal{C}$ as

$$Gm = c$$

In other words, the generator matrix is a special case of an encoding map that encodes a message m to a code-word $c \in \mathcal{C}$ using a linear transformation. An important fact (which we will use without proof) about linear codes is that its minimum distance $\Delta(\mathcal{C})$ is equal to the minimum weight $wt(\mathcal{C})$ over all non-zero code-words.

For every linear code $\mathcal{C}$, we define its parity check matrix to be a matrix H which is the null space of

G. Indeed, a code-word c is in $\mathcal{C}$ if and only if $Hc = 0$. Thus, we can fully define our code as

$$\mathcal{C} = \{c \in \mathbb{F}_q^n \mid Hc = 0.\}$$

Lastly, if $\mathcal{C}$ is a linear code, then we can define its dual code, $\mathcal{C}^\perp = \{z \in \mathbb{F}_q^n \mid \langle z, c\rangle = 0 \ \forall c \in \mathcal{C}\}$ where $\langle v, w\rangle$ denotes the dot product. An alternative view of the dual code $\mathcal{C}^\perp$ is that it has generator matrix $G^\perp = H^\top$ where H is the parity check matrix of $\mathcal{C}$. Many times, the dual of a code tend to have nice properties: a classical example being that the dual of the Hamming code is the Hadamard Code.

2.2 Expander Graphs

First, some notation: A d- regular graph is a graph $G = (V, E)$ where all vertices $v \in V$ have degree d. Additionally, if there is an edge between u and v in G, then we denote this edge $u \sim v$. Lastly, we let $G = (L \cup R, E)$ denote a bipartite graph where L and R represent the left and right partitions of the vertices of G.

The complete graph K_n is hyper-connected with the pitfall of utilizing $\binom{n}{2}$ edges. Expander graphs aim to mimic the properties of K_n while simultaneously remaining sparse. Such a graph has immediate applications towards implementing distributed networks, as being highly connected ensures that an adversary could not disrupt the network while being sparse minimizes maintenance costs. Indeed, this goal motivates the following definition of an edge-expander [2].

Definition 2.2.1 ***(Edge Expander)*** A d-regular graph G is (K, ϵ)-edge expander if for all subsets S of size at most K, the number of edges with only one vertex in S, $E(S)$ is at least $\epsilon|S|d$.

To compute whether a graph $G = (V, E)$ is an edge-expander, we desire $E(S)\ \forall S, \subset V$. From a computational perspective, it would be ideal to determine a graph's expanding properties based on the adjacency matrix A_G as that would be the natural representation of G. To that end, we first define the spectral notion of an expander and then characterize the relationship between edge- and spectral-expanders with the Expander Mixing Lemma.

Definition 2.2.3 ***(Spectral Expander)*** Let $G = (V, E)$ be a d-regular graph with n vertices and an adjacency matrix A_G that has eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. G is a λ-spectral expander if

$$\lambda(G) := \max\{|\lambda_2|, |\lambda_n|\} \leq \lambda.$$

Lemma 2.2.4 ***(General Expander Mixing Lemma)*** Let $G = (V, E)$ be d- regular and a λ- expander and let f and g be two functions such that $f, g : v \to \mathbb{R}$, then we must have that

$$\left|\mathbb{E}_{a\sim a'}[f(a) \cdot g(a')] - \mu_f \mu_g\right| \leq \lambda \sigma_f \sigma_g$$

where $\mu_f = \mathbb{E}_a[f(a)]$, $\mu_g = \mathbb{E}_a[g(a)]$, $\sigma_f^2 = \mathrm{Var}_a[f(a)]$, and $\sigma_g^2 = \mathrm{Var}_a[f(g)]$.

Proof. We slightly abuse notation so that $f = [f(v_1)\ f(v_2)\ \cdots\ f(v_n)]^\top$ and similarly, $g = [g(v_1)\ g(v_2)\ \cdots\ g(v_n)]^\top$.

Our key observation is that:

$$f^\top A_G g = \sum_{a\sim a'} f(a)g(a') = nd \cdot \mathbb{E}_{a\sim a'}[f(a)\cdot g(a')]$$

Because we know that A_G is symmetric, we know that there is an orthonormal eigenbasis (which is comprised of the normalized eigenvectors). Now, denote each eigenvector v_i, where v_i corresponds to the eigenvalue λ_i and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. We can write that $f = \sum_{i=1}^n \alpha_i v_i$ and $g = \sum_{i=1}^n \beta_i v_i$. Now, we compute:

$$f^\top A_G g = \big(\sum_{i=1}^n \alpha_i v_i\big)^\top A_G \big(\sum_{i=1}^n \beta_i v_i\big) = \sum_{i=1}^n \alpha_i \beta_i \lambda_i$$

Because we know that the first eigenvector of a d- regular graph is the all 1's vector $\vec{1}$ which corresponds to the eigenvalue d, we have that the $\alpha_1 = \frac{\langle \vec{1}, f\rangle}{\sqrt{n}}$ and $\beta_1 = \frac{\langle \vec{1}, g\rangle}{\sqrt{n}}$. Thus, we substitute this quantity into our sum to get:

$$\frac{\langle \vec{1}, f\rangle}{\sqrt{n}} \cdot \frac{\langle \vec{1}, g\rangle}{\sqrt{n}} \cdot d + \sum_{i=2}^n \alpha_i\beta_i\lambda_i = nd \cdot \mu_f\mu_g + \sum_{i=2}^n \alpha_i\beta_i\lambda_i$$

as $\langle \vec{1}, f\rangle = n\mu_f$ and $\langle \vec{1}, g\rangle = n\mu_g$. We now complete our proof by shifting the $nd \cdot \mu_f\mu_g$ to the LHS and bounding with the Triangle Inequality and Cauchy-Schwarz.

$$\Big|nd \cdot \mathbb{E}_{a\sim a'}[f(a)\cdot g(a')] - nd \cdot \mu_f\mu_g\Big| = \Big|\sum_{i=2}^n \lambda_i\alpha_i\beta_i\Big| \leq \lambda(G)\sum_{i=2}^n |\alpha_i\beta_i|$$

$$\implies \Big|\mathbb{E}_{a\sim a'}[f(a)\cdot g(a')] - \mu_f\mu_g\Big| \leq \frac{\lambda}{nd} \cdot ||f||_2 \cdot ||g||_2 \ \leq \lambda\sigma_f\sigma_g \qquad \square$$

It is worth noting that Lemma 2.2.4 is not the typical form of the well-known expander-mixing lemma, but rather a generalized form. Though this general form is a known fact, we were unable to find a formal proof for it in literature, which is why we provided it here. The standard Expander Mixing Lemma is often most useful because it is phrased in terms of some deterministic graph. Indeed, the proof for the ANNBR transform depends on the deterministic formulation of the Expander Mixing Lemma (Corollary 2.2.5).

Corollary 2.2.5 ***(Standard Expander Mixing Lemma)*** If $G = (V, E)$ is a d-regular, λ-spectral expander, then, for all $S, T \subseteq V$:

$$|e(S,T) - \frac{d|S||T|}{n}| \leq \lambda d\sqrt{|S||T|},$$

where $e(S,T) = |E(S,T)|$, the number of edges between S and T.

Proof. For any two sets S and T, let f and g be the indicator functions which represent the two sets respectively. This means that in vector form, we have $f = [f(v_1)\ f(v_2)\ \cdots\ f(v_n)]^\top$ where $f(v_i) = 1$ if $v_i \in S$ and 0 otherwise. g is defined equivalently for the set T. Applying Lemma 2.2.4 and rearranging terms yields the desired result. $\square$

3 Coding Theory Developments

3.1 Expander Codes

Let G be a bipartite graph $G = (L \cup R, E)$, with $|L| = n$ and $|R| = m$, we define the matrix H such that, for all $i \in L$, and $j \in R$, we have:

$$H_{ij} = \begin{cases} 1, & \text{if i and j are adjacent} \\ 0, & \text{otherwise} \end{cases}$$

Given H, we define the expander code [3, 2] to be:

$$\mathcal{C}(G) := \ker H = \{x \in \mathbb{F}_2^n : xH = 0\}$$

Informally, we can think of the nodes in left as the coordinates of a code-word, and each node in the right to be a constraint that requires the XOR of all the adjacent nodes to be zero.

Lemma 3.1.1: The rate of an expander code is $\geq 1 - m/n$.

Proof. We know that the dimension of the code is $k \geq n - m$. Thus the rate is $\frac{k}{n} \geq \frac{n-m}{n} = 1 - m/n$.

Moving to the analysis of expander code's distance, it turns out that with some expanding properties, we can reach a distance $\Omega(n)$, making expander codes good codes.

For any $S \subset L$, let us define the unique neighborhood, $U(S)$, to be:

$$U(S) = \{v \in R | N(v) \cap S = 1\}$$

Lemma 3.1.2: If a (d, D)-regular bipartite graph G is a an $(\alpha$,D(1-ϵ))-expander, then for all $S \subseteq L$ such that $|S| \leq \alpha n$

$$U(S) \geq D(1 - 2\epsilon)|S|$$

Proof. Let us first start by noting that since G is D-left-regular, then the number of edges coming out of S is $D|S|$ (there are no edges between the vertices in S). We also know that by definition of $U(S)$, the vertices in $U(S)$ receive only one edge from S, thus the number of edges from S to $N(S)\backslash U(S)$ is equal to $D|S| - |U(S)|$. We know, however, that every vertex in $N(S)$ but not in $U(S)$ has at least two edges coming from S, thus the number of edges from S to $N(S)\backslash U(S)$ is $\geq 2|N(S)\backslash U(S)| = 2(|N(S)| - |U(S)|)$. With these two information, we can conclude that

$$D|S| \geq 2|N(S)| - |U(S)|$$

However, by the property of $(\alpha, D(1 - \epsilon))$-expansion, we know that $|N(S)| \geq D(1 - \epsilon)|S|$. Thus,

$$U(S) \geq 2D(1 - \epsilon)|S| - D|S|$$

$$U(S) \geq D(1 - 2\epsilon)|S| \ \square$$

Lemma 3.1.3: Given Lemma 3.1.1, the distance of the expander code $\mathcal{C}$ is $\Delta(\mathcal{C}) > \alpha n$.

Proof. Let us assume, for the sake of contradiction, that there exists one (nonzero) codeword c with a hamming weight wt(c)$\leq \alpha n$ and support S. Since $|S| \leq \alpha n$, using the previous lemma, we get that $U(S) \geq D(1-2\epsilon)|S|$, which means that U(S) $\neq \emptyset$. Thus, there exists a codeword v $\in$ U(S). Since v $\in$ U(S), then N(S)$\cap$S=1, and thus, v has only one neighbor j in S, and the remaining neighbors of v are not in S. Thus, $\oplus_{i \in N(v)} x_i = x_j$ such that j $\in S$. Since j $\in$ S, then $x_j = 1$, thus, $\oplus_{i \in N(v)} x_i = 1$, which is a contradiction to the parity check needed for v.

We are, however, able to find a better bound for the distance.

Lemma 3.1.4: $\Delta(\mathcal{C}) \geq 2\alpha(1-\epsilon)n$, where $n = |L|$.

Proof. Let us assume for the sake of contradiction that, there exists a codeword $c \in \mathcal{C}$, such that its Hamming weight is $< 2\alpha(1-\epsilon)n$, and T is its support. For this proof, given that $\mathcal{C}$ is a linear code, we can prove the lemma by only proving that the weight is $\geq 2\alpha(1-\epsilon)n$ since the weight and distance are the same. Let S be some subset in T such that $S = \alpha n$. By Lemma 3.1.2, we get that

$$|U(S)| \geq (D - 2\epsilon)|S|$$

We know that the number of edges that are stemming from $T \backslash S$ is $D|T \backslash S|$, which can help us conclude that $D|T\ S| < |U(S)|$ by following these steps

$$D|T \backslash S| < |U(S)| = D(|T| - |S|) < D(2\alpha(1-\epsilon)n - |S|)$$

$$= D(2|S|(1-\epsilon) - |S|) = D(1-2\epsilon)|S| \leq |U(S)|$$

This means that $U(S) \neq \emptyset$, however, we know that $S \subseteq T$, thus $U(T) \neq \emptyset$. As a conclusion, $x \notin C$, which is a contradiction. □

3.2 Tanner Codes

Definition 3.2.1: Given a (d, D)-regular bipartite graph G, where $|L| = n$ and $|R| = m$, and a linear code $C_0 \in \mathbb{F}_2^D$, we define a *Tanner code* $T(G, C_0)$ of G and C_0 [4, 5] as

$$T(G, C_0) = \{x \in \mathbb{F}_2^D | \forall v \in R, x|_{N(v)} \in C_0\}$$

where $x|_{N(v)}$ are the bits in x in the bits that are adjacent to v.

Remark 3.2.1: Let local code $\mathcal{C}_0$ be the parity check code. Thus, the adjacency matrix of G is equivalent to the parity check $P(G)$ of the tanner code $T(G, \mathcal{C}_0)$.

Lemma 3.2.1: Given that C_0 is linear, then T(G,C_0) is also linear.

Proof. For any x_1, $x_2 \in \mathbb{F}_2^D$, then we get

$$(x_1|_V(S)) + (x_2|_V(S)) = (x_1 + x_2)|_V(S).$$

Thus, if x_1, $x_2 \in$ T(G,C_0), thus $(x_1+x_2)|_V(R) \in C_0$, and thus, $(x_1+x_2) \in T(G, C_0)$ so $T(G, C_0)$ is linear. $\square$

Lemma 3.2.2: Given that C_0 is linear, then

$$dim(T(G, C_0)) \geq n - m(D - dim(C_0))$$

Proof. We can start by noticing that for any $v \in R$, to meet the condition $c|_{N(v)} \in \mathcal{C}_0$ creates $d - dim(\mathcal{C}_\prime)$ independent linear constraints. With $|R| = m$, $\forall v$ R, $c|_{N(V)}$ creates at most $m(d - dim(\mathcal{C}_\prime))$ independent linear constraints, which gives the desired result. $\square$

Definition 3.2.2: ***(Edge-vertex incidence graph)*** For a d-regular graph $G = (V, E)$, $|V| = N$ and $|E| = \frac{Nd}{2}$, we can generate an edge-vertex incidence graph $H_0 = (L, R, E')$, where L has a node corresponding to each edge in G and R has a node corresponding to each vertex in V. E' has an edge (e,u_e) or (e,v_e) where e=$(u_e, v_e) \in L$.

Given this definition of edge-vertex incidence graph G', we can re-define a Tanner code as:

$$X(G, C_0) = \{x \in \mathbb{F}_2^{\frac{Nd}{2}} | \forall v \in V(G), x|_{N(v)} \in C_0\}$$

Lemma 3.2.3: Given a d-regular graph $G = (V, E)$, $|V| = N$ and $|E| = \frac{Nd}{2}$, G' its edge-vertex incidence graph, and linear code $\mathcal{C}_\prime$ with rate δ_0, the rate of the Tanner code is $\geq \frac{Nd}{2}(2\delta_0 - 1)$

Proof. Using the previous lemma, we substitute $n = \frac{Nd}{2}$, m with c, and we know that $dim(\mathcal{C}_\prime) = d\delta_0$

$$dim(X(G, \mathcal{C}_0)) \geq \frac{Nd}{2} - N(d - dim(\mathcal{C}_0)) = \frac{Nd}{2} - N(d - d\delta_0) \geq \frac{Nd}{2}(2\delta_0 - 1)$$

Thus, in order to achieve our goal in achieve our goal in having a positive rate, we just need $\delta_0 \in (1/2, 1]$. $\square$

Lemma 3.2.4: Given a d-regular graph $G = (V, E)$, $|V| = N$ and $|E| = \frac{Nd}{2}$, G' its edge-vertex incidence graph, and linear code $\mathcal{C}_\prime$ with rate δ_0 and distance Δ_0, we can lower bound the distance of Tanner code by Nd.

Proof. Let us define for a nonzero codeword $c \in \mathbb{F}_2^E$, its support $= \{v | c|_{E(v)} \neq 0\}$. S represents the vertices in G that are touched by the nonzero bits of c. Let $F = (S, E_F)$, where $E_F = \{f | c_f = 1\}$. We can notice that for any vertex in F, the degree is at least $\Delta_0 d$. Thus, the total number of edges in F is at least $\Delta_0 d \frac{|S|}{2}$. Thus, the weight of c is $wt(c) = |E_F| \geq \Delta_0 d \frac{|S|}{2}$. Thus, distance of the tanner code is $\geq \frac{\Delta_0 d |S|}{2}$. Thus, to meet the desired bound, we require that $|S| = \Omega(N)$. $\square$

Intuitively, Tanner codes are useful because, with a good choice of $\mathcal{C}_0$, they only require a small expansion factor. If we use a local code $\mathcal{C}_0$ of distance Δ_0, then we only need, for all subsets of size αn of a D-left-regular bipartite graph ($|L| = N$), an expansion factor of $\geq \frac{D}{\Delta_0}$, for a distance at least αn, while for parity check codes we require a factor of at least $\frac{D}{2}$ (since distance of parity check codes is 2). Such characteristic allows us to build Tanner codes with graphs that are easier to construct thanks to its lower distance.

3.3 Distance Amplification

The motivation behind this section is to develop a framework which enables us to construct new codes with desirable properties (such as larger relative distance) based off of codes we already have. The method which we will be presenting is called the ABNNR transformation [6], which achieves greater distance through utilizing a larger alphabet.

Definition 3.3.1 ***(ABNNR Transform)*** Let $G = (L \cup R, E)$ be a d- regular bipartite $\sqrt{\delta\gamma}$ -expander graph where $L = [n]$ is the left partition of G and $R = [m]$ is the right partition. Additionally, let $\mathcal{C}$ be a binary linear code with block length $n = |L|$ and relative distance δ. The code resulting from the ABNNR transform $G(\mathcal{C})$ is defined as the set of code words $G(\mathcal{C}) = \{G(c) | c \in \mathcal{C}\}$ where the i^{th} letter of $G(c)$ is defined as

$$G(c)[i] := \langle c[N_1(i)], \cdots, c[N_d(i)] \rangle$$

where $N_k(v)$ is the k^{th} neighbor of vertex v.

Intuitively, one way to interpret Definition 3.3.1 is as follows: Place a code word $c \in \mathcal{C}$ on the left partition of3.1 G such that the j^{th} bit of c is on vertex $j \in L$. Now, we can construct the i^{th} digit of $G(c)$ by looking at the vertex $i \in R$. This vertex has d neighbors, which are $N_1(i), N_2(i), \cdots, N_d(i)$. Because we placed bits of the code word c on each of the d neighbors of i, we now have a d- length string of such bits for the i^{th} letter of $G(c)$. In practice, we would rewrite this string in base 2^d. Iterating this process for all code words $c \in \mathcal{C}$ forms the code $G(\mathcal{C})$.

Lemma 3.3.2: The rate of the code $G(\mathcal{C})$ is $\frac{R(C)}{d}$.

Proof. Consider the i^{th} letter of the code-word $G(c)$. This code-word is composed of d characters from the code word c. Because G is d- regular, we know that every bit in c is repeated exactly d times. Thus, the rate of $G(\mathcal{C})$ is $\frac{R(C)}{d}$. $\square$

Lemma 3.3.3: If $G(\mathcal{C})$ is linear, then the relative distance of the code $G(\mathcal{C})$ is $\geq 1 - \gamma$.

Proof. Because we assumed that $G(\mathcal{C})$ is linear, the distance of $G(\mathcal{C})$ is equal to the minimum weight code-word of $G(\mathcal{C})$. Choose some non-zero code-word $c \in \mathcal{C}$. Because $\mathcal{C}$ is linear and has distance δ, we know that at least a fraction δ of the bits in c are non-zero. Denote $S = \{i \mid c[i] \neq 0\}$ to be the support of c. $\square$

Now, observe that for every $j \in N(S)$, we know that $G(c)[j]$ is non-zero as $G(c)[j]$ will obtain one of its d bits from S which is non-zero. Thus, the remainder of this proof is devoted to showing that $N(S)$ is sufficiently large.

Let $T := R \setminus N(S)$. By definition, we know that $E(S,T) = \emptyset \implies e(S,T) = 0$. Applying the standard expander mixing lemma yields

$$\left| 0 - \frac{d|S||T|}{n} \right| \leq \lambda d \sqrt{|S||T|} \implies |T| \leq \frac{\lambda^2 n^2}{|S|} \leq \frac{\lambda^2 n}{\delta}.$$

Because $\lambda = \sqrt{\gamma\delta}$, we know that $|T| \leq \gamma n$. Thus, we know that every code-word in $G(c)$ besides 0 has weight at least $(1-\gamma)n$, so $G(c)$ will have relative distance $1-\gamma$. $\square$

3.4 Ta-Shma Codes

Due to the complexity of this construction and the proofs for the corresponding distance and rate, we primarily describe the construction and its key insights. There are two interpretations of Ta-Shma's construction: an algebraic one and a combinatorial one. Our focus is on the combinatorial properties of the code, and more specifically the intuition behind the s- wide replacement product.

The Gilbert-Varshamov bound is obtained from an analysis of a random code. Thus, a promising mechanism for obtaining an explicit code which meets the bound is de-randomization. The following procedure provides such a scheme [7]:

1. Start with an binary code $\mathcal{C}_0 : \{0,1\}^k \to \{0,1\}^n$. One way to view the code-word $c \in \mathcal{C}_0$ is as a function $f : [n] \to \{0,1\}$.
2. Let $G = (V, E)$ be an expander graph with $|V| = n$. Map the set of coordinates $[n]$ with vertices such that the i^{th} bit of a code-word is associated with the vertex $v_i \in V$
3. Let $W \subset G^t$ be a subset of the set of all t- length walks in A. Define $g : W \to \{0,1\}$ by $g(v_1, v_2, \cdots, v_t) = f(v_1) \oplus f(v_2) \oplus \cdots \oplus f(v_t)$. We output $g \in \{0,1\}^{|W|}$.

The above process effectively takes each code-word $c \in \mathcal{C}_0$ and deterministically "randomizes" it with a random walk to create a new code. We can alter the parameters (rate and distance) of the code by cleverly choosing the subset W of t- length random walks which we base our code off of. Indeed, the core of Ta-Shma's construction is to choose W by utilizing the s- wide replacement product. Before we discuss this, we first motivate the replacement product.

The replacement product arises from our desire to construct expander graphs with a large number of vertices. By taking the replacement product of two expander graphs, we can construct a new expander where we have significant control over the degree, number of vertices, and connectivity (λ). Formally, the replacement product is defined as follows:

Definition 3.4.1 ***(Replacement Product)***. Let $G = (V_1, E_1)$ be a D-regular graph, and $H = (V_2, E_2)$ be a d-regular graph with $V_2 = [D]$. For every vertex $v \in V_1$, assign each of its D neighbors a unique index between 1 and D. The replacement product $G \circledR H$ is a graph with $V = V_1 \times [D]$ that is constructed as follows:

1. **(Vertices)** Replace every vertex $v \in V_1$, with a copy of H denoted H_v. Denote the nodes of H_v as $\{(v,i) : i \in [D]\}$.
2. **(Edges)** For every $(u,v) \in E_1$, we will connect H_u and H_v by a single edge between (u,i) and (v,j) such that $N_i(u) = v$ and $N_j(v) = u$.

Typically, we call G the outer graph and H the inner graph. We provide the following 2 examples of the replacement product:

Example 3.4.1 Let $G = K_3$ and $H = K_2$, then $G \textcircled{R} H$ is a 6-cycle.
Example 3.4.2 Let G be a n-cycle and $H = K_2$, then $G \textcircled{R} H$ is a $2n$-cycle.

The s-wide replacement product is a natural extension of the zig-zag product of two graphs (which can be viewed as a random 3-step walk on $G \textcircled{R} H$). This enables us to model a random walk on the outer graph, G in terms of a walk on the inner graph H. Indeed, a 1-wide s- replacement product is actually just the replacement product itself. We now specify our inner and outer graphs which we will utilize for the remainder of the construction.

Theorem 3.4.2 The following constructions exist and can be constructed explicitly [7].

- **Outer Graph** We can explicitly construct an undirected d- regular $\frac{8}{\sqrt{d}}$- spectral expanding Cayley graph with $n \cdot (1 + o_n(1))$ vertices for any $n, d \in \mathbb{N}$
- **Inner Graph**We can explicitly construct an undirected 2^{2l}- regular, $(r-1)2^{-l}$- expanding Cayley graph over $\mathbb{F}_2^r$ for any $r, l \in \mathbb{N}$ with $l \leq \frac{r}{2}$

Utilizing Theorem 3.4.2, we can now build a d- regular graph with n vertices which we denote $A = (V_A, E_A)$ (outer graph) and a Cayley graph over $\mathbb{F}_2^{ms}$ B where $2^m = d$ which we denote $B = (V_B, E_B)$ (inner graph). Note that theorem 3.4.2 also gives us their expansions. Furthermore, because B is a Cayley Graph over the finite field$\mathbb{F}_2^{ms}$, we can group together the ms such that we have s-tuples of m bit strings. Each m bit string can be written in base d, meaning that we can identify each vertex of B with s- tuples of elements in $[d]$. We denote one such tuple (vertex) as $b = (b[1], b[2], \cdots, b[s])$. Furthermore, because A is d- regular, we know that $|N(a)| = d \, \forall a \in A$, so we can number each of its neighbors with $i \in [d]$. This means that if we were to examine the first coordinate of some vertex $b \in B$, then we have completely identified a neighbor of some vertex $v \in V_A$.

This means that we can define a rotation map, which takes as input a vertex a and some position i and then outputs the vertex $a' = N_i(v)$ and the position of a with regards to the numbering of $N(a')$. Because of the connection between A and B, we can define a rotation map $\phi : A \times B \to A$ via $\phi(a, b) = a'$ where a' is the $b[1]$-th neighbor of a. Because ϕ only depends on the first coordinate of b, we can equivalently express our rotation map with $\phi(a, b[1])$. We now have enough information to define the s- wide replacement product on our graphs A and B.

Definition 3.4.2 ***(s-wide Replacement Product)*** For any $k \geq 1$, we denote $sRW_{A,B}^k$ to be the the k-length s- wide replacement walk distribution. $sRW_{A,B}^k$ is obtained by first drawing $a \sim A$ and $(b_1, \cdots, b_{k-1}) \sim RW_B^{k-1}$ and then outputting the walk $(a_1, \cdots a_k) \in A^k$ where $a_1 = a$ and $a_{i+1} = \phi(a_i, b_i[1])$ for $1 \leq i \leq k-1$

Intuitively, Definition 3.4.2 provides us with a mechanism to take a random walk on A by utilizing a random walk on B. An alternative viewpoint is to view the s- wide replacement product as a random walk on the the graph $A \textcircled{R} B$ where some steps are constrained to be intra-cloud (within some H_v) or inter-cloud steps (which are on the edges already in A). These constraints ensure that we select a correct subset of A^t which can be utilized to construct a code with good distance and rate. We now put together all the pieces are describe the construction of Ta-Shma codes.

As an input, we take 3 things: a base code $\mathcal{C}_0 : \{0,1\}^k \to \{0,1\}^{n_0}$ with bias ϵ_0 and rate $R_0 = O(\epsilon_0^3)$, the outer graph promised by Theorem 3.4.2 (A), and the inner graph promised by Theorem 3.4.2 (B). Note that these are the same two graphs we have been discussing throughout this section.

Next, we connect the three inputs by adding the parameters s and m such that the following holds: $r = ms$, $d_A = 2^m$, and $m = s = 5l$. Again, the parameters r, l and $d_A = d$ are outlined in Theorem 3.4.2. We now execute the process for code construction via de-randomization which we began Section 3.4 with.

1. Define an embedding $\iota : [n_0] \hookrightarrow A$. Now, we compute a function f to represent our base code $\mathcal{C}_\prime$ by defining f such that when given a message $x \in \{0,1\}^k$, we have

$$f(a) = \begin{cases} \mathcal{C}_0(x)[i] & a = \iota(i) \\ 0 & \text{else} \end{cases}$$

2. Denote sRW^t obtained from the s- wide replacement product of A and B. Now, define $g : sRW^t \to \{0,1\}$ by $g(a_0, a_1, \cdots, a_t) = f(a_0) \oplus f(a_1) \oplus \cdots \oplus f(a_t)$. We output $g \in \{0,1\}^{|sRW^t|}$.

The code obtained from the above process has near-optimal parameters.

Theorem 3.4.1 ***(Ta-Shma Code)*** $\forall \epsilon > 0$, we can explicitly compute a linear code $\mathcal{C}$ with relative distance $\geq \frac{1}{2} - \epsilon$ and rate $\Omega(\epsilon^{2+o(1)})$ [8]

References

[1] Venkatesan Guruswami. *Notes 1: Introduction, linear codes.* CMU, January 2010.

[2] Max Hopkins and Shachar Lovett. *CSE 291g: Expanders and HDX Course Notes.* UCSD, February 2021.

[3] Bhrushundi Abhishek and Conway Alex. *Lecture 6: Expander Codes.* Rutgers University, Spring 2016.

[4] Venkatesan Guruswami. *Tanner codes and Expander Codes. Advances in Error-Correcting Codes.* UC Berkeley, Fall 2022.

[5] Venkatesan Guruswami and Ankit Sharma. *Expander Codes and their decoding.* CMU, March 2010.

[6] Joseph Naor Moni Naor Noga Alon, Jehoshua Bruck and Ron M. Roth. *Construction of asymptotically good low- rate error-correcting codes through pseudo-random graphs.* IEEE Transactions on Information Theory, 38:509–516, 1992.

[7] Silas Richelson and Sourya Roy. *Analyzing Ta-Shma's Code via the Expander Mixing Lemma.* arxiv, 26 Jan 2022.

[8] Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 238–251, 2017.