Application of Expander Graphs to Error-Correcting Codes

Brice Huang

May 17, 2017

Abstract

Error-correcting codes are a method of sending messages over a noisy channel, which may corrupt a small fraction of the transmitted bits. Good error-correcting codes are efficient, can resolve many bit-errors, and allow fast (ideally, linear in the message length) encoding and decoding.

Expander graphs are sparse graphs with strong connectivity properties, in the sense that the neighbor set of all sufficiently small vertex sets is large. In this paper, we study error-correcting codes arising from bipartite expanders. Using this approach, we will construct asymptotically good codes that allow linear-time encoding and decoding. We survey results from [5] and [6].

1 Introduction

We consider the problem of sending messages over a noisy channel, which corrupts some fraction of the bits we send. In this model, we send n-bit codewords through the channel, which may adversarially flip up to t bits of each codeword. Our goal is to design a protocol that maps each message to a codeword, such that given an output from the channel the receiver can unambiguously recover the original message.

Because much of modern communication occurs over lossy channels with low but nontrivial probability for error, error-correcting codes have many applications. For example, these codes are used for reading and writing to a CD, communication to satellites, and transmitting over the Internet.

The problem of sending digital information over a noise-contaminated channel was first considered by Shannon [4], in the context of information theory. Shannon established, for a fixed amount of noise contamination, an upper bound (now known as the Shannon capacity) on the rate of error-free data transmission through the channel. Hamming [3] formulated the modern definition of an error-correcting code and discovered the first such code, the Hamming $(4, 7)$ linear code. Golay discovered the Binary Golay Code [2], a 23-bit perfect linear code correcting 3 errors, one of the key results of early coding theory.

An important problem in coding theory is the construction of asymptotically good families of codes – families of arbitrarily large codes with lower-bounded normalized distance and rate. A related problem is the construction of an asymptotically-good family that also allows efficient encoding and decoding.

These problems were solved using codes based on expander graphs. The first work in this direction was by Gallager [1], who constructed so-called Low Density Parity-Check Codes based on random, sparse regular bipartite graphs. Zybalov-Pinsker [9] proved that with high probability, Gallager's codes are asymptotically good and have, in the circuit model of computation, decoding circuits of size $O(n \log n)$. However, due to randomness this construction is not explicit. Sipser-Spielman [5] replaced Gallager's random graphs with semi-regular bipartite expanders and showed that the resulting codes are asymptotically good and decodable in $O(n)$ time. Spielman [6] builds upon this construction, using error-reducing codes as an intermediate step, to get asymptotically good codes that can be encoded and decoded in $O(n)$ time.

In this paper, we survey the last two results. In Section 2 we formally define error-correcting codes and survey background results in coding theory. In Section 3 we introduce the expander graphs used in our construction. In Section 4 we construct Sipser-Spielman's expander code and prove that it is asymptotically good and linear-time decodable. In Section 5 we extend this construction to get Spielman's construction; we prove that this code is asymptotically good and linear-time encodable and decodable. Finally, in Section 6 we discuss these two results in the context of coding theory and expander graphs.

2 Error-Correcting Codes

In this section, we give a brief introduction to error-correcting codes. For a more thorough exposition, see e.g. [8].

2.1 Definitions

We introduce error-correcting codes in full generality. Let $\mathcal{M}$ be our message space. Then, a code is a set $C \subset \{0,1\}^n$ with $|C| = |\mathcal{M}|$, and an encoding is given by a bijective map $\psi : \mathcal{M} \to C$.

Two important metrics of a code are its *distance* and *rate*. The distance of a code C, denoted $d(C)$, is defined by

$$d(C) = \min_{\substack{x,y \in C \\ x \neq y}} d_H(x,y),$$

where d_H denotes the Hamming distance. A code's distance measures its ability to resolve corrupted bits; indeed, a code with distance $d(C)$ can resolve up to $t = \lfloor \frac{d(C)-1}{2} \rfloor$ errors by associating an output w from the channel with the (unique) codeword $x \in C$ within a Hamming distance t of w. A code's normalized distance $\delta(C)$ is defined by $\delta(C) = \frac{d(C)}{n}$.

The rate of a code C is defined by

$$r(C) = \frac{\log_2 |C|}{n}$$

and measures the number of information-bits carried by each message-bit. As a code gets sparser, its distance increases, allowing it to resolve more errors, but its rate, and thus its information-density, decreases.

Balancing the distance-rate tradeoff is one of the main considerations when designing a code. At the very least, we want our codes to resolve a constant-fraction number of errors while maintaining a constant rate. This is formalized as follows.

Definition 1. A family $\mathcal{C}$ of codes $C_n \subset \{0,1\}^n$, in which n grows arbitrarily large, is *asymptotically good* if there exist constants $\delta, r > 0$ such that for all $C_n \in \mathcal{C}$, $\delta(C_n) > \delta$ and $r(C_n) > r$.

Another consideration when designing a code is efficiency of encoding and decoding. In order for our codes to be useful, we require encoding and decoding to be done in $poly(n)$ time, and ideally in $O(n)$ time.

2.2 The Hamming Bound

The Hamming (or sphere-packing) bound places an upper limit on how much distance and rate we can simultaneously attain, thus quantifying the distance-rate tradeoff.

Theorem 2. *[3] Let $C \subset \{0,1\}^n$ be a code, and let $t = \lfloor \frac{d(C)-1}{2} \rfloor$. Then,*

$$\frac{|C|}{2^n} \leq \frac{1}{\sum_{i=0}^{t} \binom{n}{i}}.$$

Proof. For $x \in \{0,1\}^n$, define

$$B(x,\epsilon) = \{y \in \{0,1\}^n | d_H(x,y) \leq \epsilon\}.$$

For any codewords $x, y \in C$, the sets $B(x,t)$ and $B(y,t)$ cannot intersect, because if they intersect, then $d_H(x,y) \leq 2t < d(C)$. Thus, all the sets $B(x,t)$ are disjoint, and each has size $\sum_{i=0}^{t} \binom{n}{i}$. Their union is contained in $\{0,1\}^n$, so

$$|C| \sum_{i=0}^{t} \binom{n}{i} \leq 2^n.$$

□

We say a code is *perfect* if it achieves the Hamming bound.

2.3 Linear Codes

For the rest of this paper, we focus on linear codes.

Definition 3. A linear code C is a code where $\mathcal{M} = \{0,1\}^k$ for some $k < n$, and encoding is done by a linear operator (called a *generating matrix*) $G \in \mathbb{F}_2^{k \times n}$:

$$\psi(v) = v^T G.$$

In this case, C is a linear subspace of $\{0,1\}^n$ of dimension k, whose basis is given by the rows of G.

The orthogonal complement of the rows of G describes linear relations that hold for any codeword $w \in C$. Specifically, if P is a $(n-k) \times n$ matrix whose rows span the orthogonal complement of the rows of G, then $Pw = 0$ if and only if $w \in C$. The matrix P is called a *parity check matrix*, and is another way of describing the code C.

We make the following observation on the distance of linear codes. For $v \in \mathbb{F}_2^n$, let $\operatorname{supp}(v)$ be the number of nonzero entries in v.

Proposition 4. *If C is a linear code, then*

$$d(C) = \min_{\substack{v \in C \\ v \neq 0}} |\operatorname{supp}(v)|.$$

Proof. For a linear code, if $x, y \in C$, then $x \oplus y \in C$. Then,

$$d_H(x,y) = d_H(0, x \oplus y) = |\operatorname{supp}(x \oplus y)|.$$

□

Linear codes trivially allow $O(n^2)$ encoding. Decoding is harder, however; we will discuss efficient decoding later.

2.3.1 Parity Check Codes

By a change of basis of $\mathbb{F}_2^k$, we can always write generating matrices G in the form $G = \begin{pmatrix} I_k & A \end{pmatrix}$.

A linear code whose generating matrix is of this form is called a *parity check code*. In these codes, an encoding consists of the original k-bit message, followed by $n-k$ bits, called *parity check bits*, each of which is a sum of a pre-specified set of message bits.

Moreover, if $G = \begin{pmatrix} I_k & A \end{pmatrix}$ is the generating matrix for a code, then $P = \begin{pmatrix} A^T & I_{n-k} \end{pmatrix}$ is the corresponding parity check matrix.

2.3.2 Example: the Hamming $(4,7)$ Code

This example is from [3]. Consider the linear code given by the generating matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

We can check that $|\operatorname{supp}(v)| \geq 3$ for all nonzero v, so $d(C) = 3$. Thus C can correct one bit. Moreover, this code achieves the Hamming bound because

$$\frac{|C|}{2^n} = \frac{1}{8} = \frac{1}{\binom{n}{0} + \binom{n}{1}}.$$

Thus, no 7-bit code correcting one bit has a larger number of codewords.

Larger examples of perfect codes exist. For example, the Binary Golay Code [2] has $k = 12$, $n = 23$, $d(C) = 7$, which achieves the Hamming bound

$$\frac{|C|}{2^n} = \frac{1}{2048} = \frac{1}{\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \binom{n}{3}}.$$

For more on perfect codes, see e.g. [8].

3 Bipartite Expander Graphs

Our construction uses bipartite, semiregular expander graphs. In this section, we define these graphs and state their relevant properties.

We say a graph G is (c, d)-semiregular if it is bipartite with vertex partition S, T, such that all vertices in S have degree c, and all vertices in T have degree d, with $c < d$.

Definition 5. G is a (c, d, ϵ, δ)-*expander* if it is (c, d)-semiregular, and for all sets $R \subset S$ with $|R| \leq \epsilon|S|$, $|\Gamma(R)| > \delta|R|$.

Note that unlike the definition for expander graphs, which demands expansion of all sufficiently small vertex sets, this definition only considers expanding sets in S, the c-regular side.

Like for expander graphs, the expansion properties of bipartite semiregular expanders are closely related to the second-largest eigenvalue. The following result quantifies this relation.

Theorem 6. *[7] Suppose G is (c, d)-semiregular and $|\theta| \leq \lambda$ for all eigenvalues $\theta \neq \pm\sqrt{cd}$ of G. For a set $R \subset S$, define $\rho = \frac{|R|}{|S|}$. Then,*

$$\frac{|\Gamma(R)|}{|R|} \geq \frac{c^2}{\rho(cd - \lambda^2) + \lambda^2}.$$

Although explicit constructions of bipartite expander graphs are not known, a randomly chosen (c, d)-semiregular graph is a good expander with high probability, in the following sense:

Theorem 7. *[5] Let B be a randomly chosen (c, d)-semiregular bipartite graph with n c-regular vertices. For any $a \in (0, 1)$, with high probability, all sets of αn c-regular vertices have at least*

$$n\left(\frac{c}{d}\left(1 - (1 - \alpha)^d\right) - \sqrt{\frac{2c\alpha H(\alpha)}{\log_2 e}}\right)$$

neighbors, where $H(\cdot)$ is the binary entropy function.

4 The Expander Code

This construction is from [5]. We will construct an asymptotically good family of expander codes that allow linear-time decoding, though not linear-time encoding. Using a (c, d)-semiregular expander graph B whose c-degree side has n vertices, we will extend a d-bit linear code S to an n-bit linear code $C(B, S)$.

Let $S \subset \mathbb{F}_2^d$ be a linear code with parity check matrix P_S.

Let B be a (c, d)-regular graph, with $c < d$. Say the c-degree side has n vertices, which we label $\{v_1, \dots, v_n\}$; by edge-counting, the d-degree side has $\frac{c}{d}n$ vertices, which we label $\{C_1, \dots, C_{cn/d}\}$. Let b be a function such that for $i = 1, \dots, \frac{c}{d}n$, the neighbors of C_i are $v_{b(i,1)}, \dots, v_{b(i,d)}$.

Definition 8. The code $C(B, S) \in \mathbb{F}_2^n$ consists of the words $(x_1, \dots, x_n)$ such that for all $i = 1, \dots, \frac{c}{d}n$, $(x_{b(i,1)}, \dots, x_{b(i,d)}) \in S$.

We claim this is a linear code. Indeed, let $M_{B,i}$ be the $0-1$ matrix that maps $(x_1, \dots, x_n)$ to $(x_{b(i,1)}, \dots, x_{b(i,d)})$. Then, the parity check matrix P of $C(B, S)$ is the matrix whose rows are the union of the rows of the matrices $P_S M_{B,i}$, for $i = 1, \dots, \frac{c}{d}n$.

Theorem 9. *[5] Suppose B is a $(c, d, \alpha, \frac{c}{d\epsilon})$-expander, and S has rate $r > 1 - \frac{1}{c}$ and normalized distance ϵ. Then $C(B, S)$ has rate at least $1 - c(1 - r)$ and normalized distance at least α.*

This theorem implies that, given a family of $(c, d, \alpha, \frac{c}{d\epsilon})$-expanders with increasing n, we can construct arbitrarily large asymptotically good error-correcting codes.

Proof of Theorem 9. Each matrix $P_S M_{B,i}$ has $(1 - r)d$ rows, so the parity check matrix of $C(B, S)$ has

$$\left(\frac{c}{d}n\right)((1 - r)d) = cn(1 - r)$$

rows. The code $C(B, S)$ is generated by the orthocomplement of these rows, so it has dimension at least $n - cn(1 - r)$, yielding a rate of at least

$$\frac{n - cn(1 - r)}{n} = 1 - c(1 - r).$$

Next, we bound the normalized distance. Suppose for contradiction that some word $w \in C(B, S)$ such that $|\operatorname{supp}(w)| \le \alpha n$. Let R be the vertices in G corresponding to the coordinates of $\operatorname{supp}(w)$. By the expansion property, $|\Gamma(R)| > \frac{c}{d\epsilon}|R|$. There are $c|R|$ edges from R to $\Gamma(R)$, so some constraint $C_i \in \Gamma(R)$ has fewer than $d\epsilon$ neighbors in R. Then $(x_{b(i,1)}, \dots, x_{b(i,d)}) \in S$ has fewer than $d\epsilon$ 1s, contradicting that the normalized distance of S is ϵ. □

Let $S = \mathcal{P} \subset \mathbb{F}_2^d$ be the code consisting of all even-weight words, so $P_{\mathcal{P}} = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}$. Note that $\mathcal{P}$ has normalized distance $\frac{2}{n}$ and rate $r = 1 - \frac{1}{d}$. Thus:

Corollary 10. *If B is a $(c, d, \alpha, \frac{1}{2}c)$-expander, then $C(B, \mathcal{P})$ has normalized distance at least α and rate at least $1 - \frac{c}{d}$.*

4.1 Linear-Time Decoding

Assuming the existence of expander graphs with sufficiently good expansion properties, we will give an asymptotically good family of expander graphs that allow linear-time decoding. The main result of this section is the following.

Theorem 11. *If B is a $(c, d, \alpha, \frac{3}{4}c)$-expander, then the code $C(B, \mathcal{P})$ permits an $\frac{1}{2}\alpha$ fraction of errors to be corrected in linear time.*

By Corollary 10, $C(B, \mathcal{P})$ has normalized distance at least α and rate at least $1 - \frac{c}{d}$.

Theorem 11 implies that, assuming the existence of a family of $(c, d, \alpha, \frac{3}{4}c)$-expanders $\{B_n\}_{n=1}^{\infty}$, the codes $C_n = C(B_n, \mathcal{P})$ form an asymptotically good family that can be decoded in linear time.

Recall that the code $C(B, \mathcal{P})$ has n variables and $\frac{c}{d}n$ constraints. Given a word $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, we say a constraint C_i is *satisfied* if $(x_{b(i,1)}, \ldots, x_{b(i,n)}) \in \mathcal{P}$, i.e.

$$\sum_{k=1}^{d} x_{b(i,k)} \equiv 0 \pmod 2,$$

and otherwise *unsatisfied*.

We will show that the following algorithm decodes an $\frac{1}{2}\alpha$ fraction of errors of $C(B, \mathcal{P})$.

Algorithm 12. While not all constraints are satisfied, find a variable x_i in more unsatisfied than satisfied constraints, and toggle x_i.

Let $w = (w_1, \ldots, w_n) \in C(B, \mathcal{P})$ be the codeword closest to x at the beginning of the algorithm. We say a variable x_i is *corrupt* if it differs from the corresponding w_i, and *valid* otherwise.

At each stage of the algorithm, we say the algorithm is in state (u, v) if there are u unsatisfied constraints and v corrupt variables. We let A be the set of corrupt variables, so $|A| = v$.

Lemma 13. *Over the progress of the algorithm, u is a decreasing monovariant.*

Proof. When we toggle a variable x_i, all the unsatisfied constraints involving x_i become satisfied and vice versa. Because x_i was in more unsatisfied than satisfied constraints, the number of unsatisfied constraints decreases. □

Lemma 14. *If the algorithm is in state (u, v) with $0 < v \le \alpha n$, then $u > \frac{1}{2}cv$.*

Proof. Clearly all the unsatisfied constraints are in $\Gamma(A)$. Let there be s satisfied constraints in $\Gamma(A)$.

By expansion properties of B, we have

$$u + s = |\Gamma(A)| > \left(\frac{3}{4}c\right)|A| = \frac{3}{4}cv.$$

All unsatisfied constraints in $\Gamma(A)$ must have at least one edge to A, while all satisfied constraints in $\Gamma(A)$ must have at least two edges to A. Since there are cv edges from A to $\Gamma(A)$,

$$cv \ge u + 2s.$$

Therefore

$$u + (u + 2s) = 2(u + s) > 2\left(\frac{3}{4}cv\right) = cv + \frac{1}{2}cv \ge (u + 2s) + \frac{1}{2}cv \Rightarrow u > \frac{1}{2}cv.$$

□

Lemma 15. *If the algorithm is in state (u, v) with $v \le \alpha n$, then there exists a variable x_i in more unsatisfied than satisfied constraints.*

Proof. By Lemma 14, $u > \frac{1}{2}cv$. Since all the unsatisfied constraints are in $\Gamma(A)$, and there are cv edges from A to $\Gamma(A)$, more than half of these edges are from A to an unsatisfied constraint. By the Pigeonhole Principle, some variable $x_i \in A$ is in more unsatisfied than satisfied constraints. □

Proposition 16. *If Algorithm 12 begins in state $(u, v) = (u_0, v_0)$ with $v_0 \le \frac{1}{2}\alpha n$, then it will correctly decode its input.*

Proof. We will show that:

- The algorithm is well-defined, i.e. when not all constraints are satisfied, there is a valid variable to toggle;
- The algorithm terminates;
- When the algorithm terminates, the value of x is the codeword w.

By Lemma 15, for the algorithm to not be well-defined it must reach a state (u, v) with $v > \alpha n$. Since each toggle changes the value of v by 1, there must have been a state with $v = \alpha n$. By Lemma 14, at this state $u > \frac{1}{2}c\alpha n$. But, in the original state, we must have $u_0 \leq \frac{1}{2}c\alpha n$ because the corrupt variables are collectively in at most $v_0 c \leq \frac{1}{2}c\alpha n$ constraints. This contradicts Lemma 13 that u is decreasing.

Since u is a decreasing monovariant, the algorithm will terminate.

Moreover, by definition the algorithm terminates if and only if $u = 0$. We claim that $x = w$ when the algorithm terminates, i.e. $v = 0$. Suppose for contradiction that $v > 0$. We showed above that $v \leq \alpha n$, so by Lemma 14, $u > \frac{1}{2}cv > 0$, contradiction. Therefore, $x = w$ when the algorithm terminates. □

Finally, we will prove that an implementation of Algorithm 12 terminates in linear time.

Lemma 17. *Algorithm 12 terminates after $O(n)$ toggles.*

Proof. Say Algorithm 12 begins in state $(u, v) = (u_0, v_0)$. Since there are $\frac{c}{d}n$ constraints, $u_0 \leq \frac{c}{d}n$. By Lemma 13, u decreases by at least 1 with every toggle. Therefore, the algorithm terminates in at most $\frac{c}{d}n = O(n)$ steps. □

Proposition 18. *Algorithm 12 can be implemented in $O(n)$ time.*

Proof. We claim the following implementation suffices:

```
1   Initialize S to a set of pointers, initially empty
2   For i = 1,...,cn/d:
3     Initialize boolean variable C[i]; set C[i] true iff constraint C_i is satisfied
4   For j = 1,...,n:
5     Initialize integer variables B[j]; set B[j] equal to the number of unsatisfied
        constraints including variable x_i
6     Add a pointer to x_i to S iff B[j] > cn/2d
7   While S is nonempty:
8     Choose a variable x_k in S
9     Toggle x_k
10    For all constraints C containing x_k:
11      Toggle C[i]
12      For all variables x_m in constraint C:
13        Increment (resp. decrement) B[m] if C[i] was toggled from false to true (resp.
            true to false) in line 9
14        Recompute whether B[m] > cn/2d and add or remove a pointer to x_m from S if
            necessary, such that x_m is in S iff B[m] > cn/2d
```

We first show the initialization steps (before line 7) take $O(n)$ time.

Initializing the variables $C[i]$ require looping over the d neighbors of each of the $\frac{c}{d}n$ constraints; this takes $\left(\frac{c}{d}n\right)d = cn = O(n)$ time; initializing the variables $B[j]$ and the set S require looping over the c neighbors of each of the n variables; this also takes $cn = O(n)$ time.

By Lemma 17, the loop beginning in line 7 is run $O(n)$ times. Within this loop, there is a loop over the c constraints containing x_k, and within that a loop over the d variables in each of these constraints. The contents of the inner loop take $O(1)$ time.

Therefore, each iteration of the line 7 loop takes $cdO(1) = O(1)$ time. The line 7 loop is run for $O(n)$ iterations, so running this loop takes $O(n)$ time. □

Finally, we can prove Theorem 11.

Proof of Theorem 11. This theroem follows from Propositions 16 and 18. □

5 Error Reduction Codes

The constructions in this section are from [6]. We will define error reduction codes, a class of linear codes that allow encoding and partial decoding in linear time. From these codes we will construct error-correcting codes that can be encoded and fully decoded in linear time. We will then construct error reduction codes from expander graphs.

Definition 19. Let C be an n-bit parity-check code with rate r; by definition this has rn message bits and $(1-r)n$ check bits. C is an *error reduction code of error reduction ϵ and reducible distance δ* if there exists an algorithm that, on input x, differing from a word $w \in C$ by at $v \le \delta n$ message bits and $t \le \delta n$ check bits, outputs a word differing from w by at most ϵt message bits.

If C is a parity-check code, let $C(m)$ denote the string of parity-check bits produced by the code C on message m.

5.1 Error Correcting Codes from Error Reduction Codes

For this construction, we assume we have an n_0-bit error-correcting code $\mathcal{C}_0$ of rate $\frac{1}{4}$ that can resolve a $\frac{\delta}{4}$ fraction of error. We also assume we have a family of error correcting codes $\{\mathcal{R}_k\}_{k=-1}^{\infty}$; $\mathcal{R}_k$ is a $\frac{3}{2}n_0 2^k$-bit parity-check code with rate $\frac{2}{3}$ (i.e. $n_0 2^k$ message bits and $n_0 2^{k-1}$ check bits), reducible distance δ, and error reduction $\frac{1}{2}$.

We will construct a family $\{\mathcal{C}_k\}_{k=0}^{\infty}$ of error-correcting codes, where $\mathcal{C}_k$ is a $n_0 2^k$-bit error-correcting code of rate $\frac{1}{4}$ that can resolve a $\frac{\delta}{4}$ fraction of error.

The construction is recursive. Given a message M of length $|M| = n_0 2^{k-2}$, $\mathcal{C}_k$ generates the parity check bits

$$\mathcal{C}_k(M) = ABC,$$

where $A = \mathcal{R}_{k-2}(M)$, $B = \mathcal{C}_{k-1}(A)$, and $C = \mathcal{R}_{k-1}(AB)$. Note that the constituent subwords of the resulting codeword, $MABC$ have lengths

$$|M| = \frac{1}{4}n_0 2^k, \qquad |A| = \frac{1}{8}n_0 2^k, \qquad |B| = \frac{3}{8}n_0 2^k, \qquad |C| = \frac{1}{4}n_0 2^k,$$

with total length $n_0 2^k$, and that exactly $\frac{1}{4}$ of these bits are message bits, as claimed.

Lemma 20. *The code $\mathcal{C}_k$ is an $n_0 2^k$-bit error-correcting code of rate $\frac{1}{4}$ that can resolve a $\frac{\delta}{4}$ fraction of error. Moreover, if the codes $\mathcal{R}_k$ can be encoded by linear-time algorithms E_k, and error-reduced by linear-time algorithms D_k that:*

- *On input x, differing from a word $w \in \mathcal{R}_k$ by at $v \le \delta n$ message bits and $t \le \delta n$ check bits, outputs a word differing from w by at most $\max(\frac{1}{2}v, \frac{1}{2}t)$ message bits, [1] and*
- *On input x, differing from a word $w \in \mathcal{R}_k$ by at $v \le \delta n$ message bits and $t = 0$ check bits, outputs w,*

[1] Note that this is slightly weaker than the error-reduction condition for $\epsilon = \frac{1}{2}$, which requires that it output a word differing from w by at most $\frac{1}{2}t$ message bits.

then $\mathcal{C}_k$ can be encoded and decoded by linear-time algorithms E'_k and D'_k.

Proof. Encoding is trivial; the prescribed operations define E'_k and clearly take linear time.

We will exhibit an algorithm D'_k and prove its correctness; the fact that it runs in linear time should be clear.

We define D'_k recursively. On input a word $x = M_0A_0B_0C_0$, where $|M_0| = \frac{1}{4}n_0 2^k, |A_0| = \frac{1}{8}n_0 2^k, |B_0| = \frac{3}{8}n_0 2^k, |C_0| = \frac{1}{4}n_0 2^k$, D'_k:

- Runs $D_{k-1}(A_0B_0C_0)$, which outputs $A_1B_1C_1$;
- Runs $D'_{k-1}(A_1B_1)$, which outputs A_2B_2;
- Runs $D_{k-2}(M_0A_2)$, which outputs M_1A_3;
- Returns $E'_k(M_1)$.

Suppose $x = M_0A_0B_0C_0$ differs from a codeword $w = MABC$ in at most $\frac{1}{4}\delta n_0 2^k$ bits. Then, there are at most this many errors in A_0B_0 and in C_0. By the error-reducing properties of D_{k-1}, A_1B_1 differs from AB in at most $\frac{1}{4}\delta n_0 2^{k-1}$ bits. By the error-correcting properties of D'_{k-1}, all the errors in A_1B_1 get corrected, so $A_2B_2 = AB$. Since $A_2 = A$, the error-reducing properties of D'_{k-2} imply that $M_1 = M$. Thus $E'_k(M_1)$ is the codeword $MABC$. □

5.2 Error Reduction Codes from Expander Graphs

We use the same notation as Section 4. Let B be a $(d, 2d)$-regular graph. Say the d-degree side has n vertices, which we label $\{v_1, \dots, v_n\}$; we label the $\frac{n}{2}$ vertices on the $2d$-degree side $\{C_1, \dots, C_{n/2}\}$. Let b be a function such that for $i = 1, \dots, \frac{1}{2}n$, the neighbors of C_i are $v_{b(i,1)}, \dots, v_{b(i,2d)}$.

Definition 21. The code $\mathcal{R}(B) \in \mathbb{F}_2^{3n/2}$ consists of the words $(x_1, \dots, x_n, c_1, \dots, c_{n/2})$, with message bits $x_1, \dots, x_n$ and parity-check bits $c_1, \dots, c_{n/2}$, where

$$c_i = \sum_{k=1}^{2d} x_{b(i,k)} \pmod 2.$$

The main result of this section is the following.

Theorem 22. *$\mathcal{R}(B)$ is an error reduction code with rate $\frac{2}{3}$, error reduction $\frac{1}{2}$, and reducible distance $\frac{1}{3}\alpha$. Moreover, there exists a reduction algorithm for $\mathcal{R}(B)$ that takes linear time.*

Note the relation between this code and the code $C(B, \mathcal{P})$. If the code $C(B, \mathcal{P})$ has parity check matrix P, the code $\mathcal{R}(B)$ has generating matrix $G = \begin{pmatrix} I & P \end{pmatrix}$. In fact, the code $C(B, \mathcal{P})$ can be identified as the linear subspace of the code $\mathcal{R}(B)$ where all the check bits are 0.

Since each check bit is computed from $2d$ message bits, it is clear that $\mathcal{R}(B)$ can be encoded in linear time.

Given a word $x = (x_1, \dots, x_n, c_1, \dots, c_{n/2}) \in \mathbb{F}_2^{3n/2}$, we say that a check bit c_i is *satisfied* if $c_i = \sum_{k=1}^{2d} x_{b(i,1)} \pmod 2$, and otherwise *unsatisfied*. If w is the nearest codeword to u, say that a message bit or check bit in u is *corrupt* if it differs from the corresponding bit in w, and otherwise *valid*.

We will use the following algorithm to reduce errors in $\mathcal{R}(B)$:

Algorithm 23. If there exists at least one message bit x_i that contributes to more unsatisfied than satisfied check bits, pick any such x_i and toggle it.

Note the similarity of this algorithm to Algorithm 12. In fact, if all of the check bits c_i are valid, the proof of Theorem 9 directly applies and shows that Algorithm 23 corrects all errors.

We will prove the following proposition, which shows that when the number of check-bit errors is small, Algorithm 23 returns a word in which the number of message-bit errors is small.

Proposition 24. *If B is a $(d, 2d, \alpha, \frac{3}{4}d+2)$-expander, and $x = (x_1, \dots, x_n, c_1, \dots, c_{n/2}) \in \mathbb{F}_2^{3n/2}$ differs from a codeword $w \in \mathcal{R}(B)$ in $v \le \frac{1}{2}\alpha n$ message bits and $t \le \frac{1}{2}\alpha n$ check bits, then Algorithm 23 on x returns a word x' that differs from w in at most $\frac{1}{2}t$ message bits.*

Like before, we let u be the number of unsatisfied check bits and consider the state of the algorithm to be (u, v). We let A be the set of corrupt message bits, so $|A| = v$.

The proof of Lemma 13 still applies, so u is a decreasing monovariant.

Lemma 25. *If the algorithm is in a state (u, v) with $\frac{1}{2}t \le v \le \alpha n$, then $u > \left(\frac{1}{2}d + 4\right)v - t$.*

Proof. All the unsatisfied check bits are in $\Gamma(A)$. Let there be s satisfied check bits in $\Gamma(A)$.

By expansion properties of B, we have

$$u + s = |\Gamma(A)| > \left(\frac{3}{4}d + 2\right)v.$$

All unsatisfied check bits in $\Gamma(A)$ must have at least one edge to A or be corrupt; all satisfied check bits in $\Gamma(A)$ must have at least two edges to A, or have at least one edge to A and be corrupt. Since there are cv edges from A to $\Gamma(A)$, and t corrupt check bits,

$$dv + t \ge u + 2s.$$

Therefore,

$$u + (u + 2s) = 2(u + s) > 2\left(\frac{3}{4}d + 2\right)v = \frac{1}{2}dv + dv + 4v \ge \frac{1}{2}dv + (u + 2s) - t + 4v \Rightarrow u > \left(\frac{1}{2}d + 4\right)v - t.$$

□

Lemma 26. *If the algorithm is in state (u, v) with $\frac{1}{2}t \le v \le \alpha n$, then there exists a message bit x_i contributing to more unsatisfied than satisfied check bits.*

Proof. By Lemma 25, $u > \left(\frac{1}{2}d + 4\right)v - t$. Since $\frac{1}{2}t \le v$, we have

$$u > \frac{1}{2}dv + 4v - t \ge \frac{1}{2}dv + t.$$

Since all the unsatisfied check bits are in $\Gamma(A)$, and there are dv edges from A to $\Gamma(A)$, more than half of these edges are from A to an unsatisfied check bit. By the Pigeonhole Principle, some message bit $x_i \in A$ contributes to more unsatisfied than satisfied check bits. □

Proof of Proposition 24. We will show that:

- The algorithm terminates;
- When the algorithm terminates, $v < \frac{1}{2}t$.

Because u is a monovariant, the algorithm terminates.

By Lemma 26, for the algorithm to not terminate with $v < \frac{1}{2}t$, it must terminate with $v > \alpha n$. Since each toggle changes the value of v by 1, at some point we must have had $v = \alpha n$. By Lemma 25, at that point

$$u > \left(\frac{1}{2}d + 4\right) v - t \geq \frac{1}{2}dv + t \geq \frac{1}{2}d\alpha n.$$

where we have used the inequality $\frac{1}{2}t \leq v$.

But, in the original state, we had $u_0 \leq \frac{1}{2}d\alpha n$ because the corrupt message bits collectively contribute to at most $v_0 c \leq \frac{1}{2}c\alpha n$ check bits. This contradicts that u is decreasing. □

Proof of Theorem 22. The proof of Proposition 18 can be repeated for Algorithm 23 to show that it terminates in linear time. The result then follows from Proposition 24. □

6 Discussion

In this paper, we constructed Sipser-Spielman's expander codes, an asymptotically-good family of codes allowing linear-time decoding. We then extended this construction to get Spielman's codes, which are an asymptotically-good family supporting linear-time encoding and decoding. In doing so, we have achieved a long-standing goal of coding theory.

The similarity of the methods used to prove our two results should not be surprising, because of the two codes' shared algebraic properties; indeed, each of Spielman's error-reducing codes $\mathcal{R}(B)$ contains an expander code $C(B, \mathcal{P})$ as a linear subspace. The algebraic properties that give rise to asymptotic goodness and efficient encoding and decoding are themselves rooted in the expansion of the underlying graphs, demonstrating the power of expander graphs.

References

[1] R. G. Gallager. *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA, 1963.

[2] M. J. E. Golay. Notes on digital coding. *Proceedings of the IRE*, 37:657, 1949.

[3] R. Hamming. Error detecting and error correcting codes. *Bell System Tech. Journal*, 29:147–160, 1950.

[4] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. Journal*, 27:372–423, 623–656, 1948.

[5] M. Sipser and D. A. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42(6, part 1):1710–1722, 1996.

[6] D. A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inform. Theory*, 42(6, part 1):1723–1731, 1996.

[7] R. M. Tanner. Explicit concentrators from generalized n-gons. *SIAM Journal on Algebric Discrete Methods*, 5:287–293, 1984.

[8] J. H. van Lint. *Introduction to Coding Theory*, volume 86 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 1999.

[9] V. V. Zyablov and M. S. Pinsker. Estimation of the error-correction complexity of gallager low-density codes. *Problems of Information Transmission*, 11(1):18–28, 1975.