IMPROVING THE ALPHABET-SIZE IN EXPANDER BASED CODE CONSTRUCTIONS

Abstract

Various code constructions use expander graphs to improve the error resilience. Often the use of expanding graphs comes at the expense of the alphabet size. This is the case, e.g., in [1], [9] and [7].

We show that by replacing the balanced expanding graphs used in the above constructions with unbalanced dispersers or extractors (depending on the actual construction) the alphabet size can be dramatically improved.

Contents

1. Introduction

A powerful technique for constructing high noise resilient codes uses a combination of codes with expanding graphs. The technique was first introduced by [1], and further developed in [9], [6] and [7]. The combination between a code and a bipartite expanding graph can be thought of as a concatenation with a repetition code, followed by mixing and regrouping the codes' coordinates. This composition can be formalized as follows:

Definition 1. *(graph encoding) Let* $G = ([N], [L], E)$ *be a regular bipartite graph, with regular right degree* T. *Let* Σ *be an alphabet. We define a function* $G : \Sigma^N \to (\Sigma^T)^L$ *as follows: Given* $x \in \Sigma^N$, *we let* $G(x) = G(x)_1, \ldots, G(x)_L$, *where* $G(x)_l = (x_{l_1}, \ldots, x_{l_T})$, *and* $l_1, \ldots, l_T \in [N]$ *are the neighbors of* $l \in [L]$ *in* G.

Figure 1 illustrates this graph encoding.

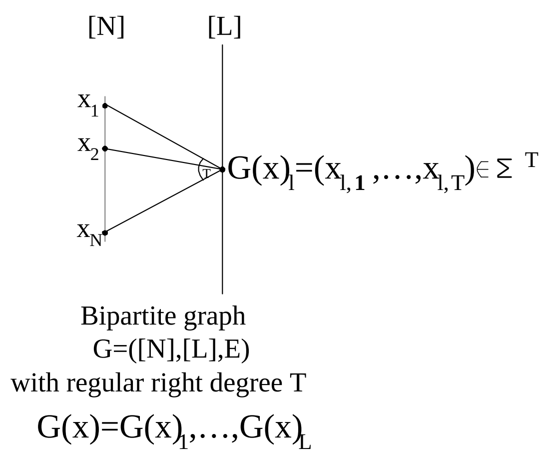

Figure 1. *Graph encoding.* $x_1, \ldots, x_N \in \Sigma$ *on the left are "put" along the graph's left side* $[N]$, *defining for each* $l \in [L]$ *an ordered vector of its neighbors:* $G(x)_l = (x_{l,1}, \ldots, x_{l,T}) \in \Sigma^T$, *where* $x_{l,t}$ *is the symbol that was matched to the* t^{th} *neighbor of* $l \in [L]$ *in* $[N]$. $G(x)$ *is defined to be* $G(x)_1, \ldots, G(x)_L$.

Definition 2. *(composition) Let* $C \subset \mathbb{F}_{q^N}$. *Let* G *be as above. We define the composition code* $G \circ C = \{G(c) | c \in C\}$

A trivial fact about the composition described above is that if C is linear then so is $G \circ C$.

To achieve the high error resilience, or large relative distance, G must have the following property: any small subset of $[L]$, say of size ϵL, sees almost all vertices in $[N]$, say at least $(1-\delta)N$. This property is the property of a disperser. Dispersers, extractors, error correcting codes, list decodable codes and other definitions needed for our purpose are fully detailed in section 3.

As we show next, this disperser's property assures that if C has relative distance δ, then $G \circ C$ has relative distance $(1-\epsilon)$.

We now define the entropy loss of a disperser, which plays a major role in our analysis. Having regular right degree T implies that any set of size ϵL on the right can see at most ϵLT vertices on the left. The disperser's expansion property assures that the set sees almost all vertices of $[N]$, and so ϵLT must be $\Omega(N)$. However, it can be much larger. Thus, a measurement for the quality of the expansion is $\Lambda_G = \frac{\epsilon LT}{N}$, called the entropy loss of the disperser.

The following lemma demonstrates how the composition above increases error resilience, and summarizes the parameters of the code composition $G \circ C$ as a function of the parameters of C and G.

Lemma 1. *If $G : [L] \times [T] \to [N]$ is a $(\epsilon L, \delta)$- disperser with entropy loss Λ, and if C is a $[N, rN, \delta N]_q$ code then $G \circ C$ is a $[L, \frac{r \cdot \epsilon}{\Lambda} L, (1-\epsilon)L]_{q^T}$ code*

We give the proof in section 5.1. This property translates the small relative distance δ to large relative distance $(1-\epsilon)$, while increasing the alphabet size.

1.1. **Previous Work and Our Improvement.** [1] take the graph G to be a balanced expander ($L = N$), and use the fact that it is a good disperser. As we saw this suffices for the error amplification. The cost of this, however, is enlarging the alphabet from Σ to Σ^T. Recall that the expansion property required is that any ϵN vertices on the left will see an least $(1-\delta)N \geq \frac{1}{2}N$ vertices on the right. This implies $\epsilon NT \geq \frac{1}{2}N$, yielding a degree T of order $(\frac{1}{\epsilon})$.

However, if we take an unbalanced disperser we can achieve the same error with a much smaller degree ($T = \log^{O(1)}(\frac{1}{\epsilon})$), yielding a much smaller alphabet size. One can worry what happens to the rate when taking an unbalanced disperser. However, as we saw before the new rate is $\frac{r \cdot \epsilon}{\Lambda_G}$. Thus, by taking a disperser with optimal entropy loss, we don't lose on the rate, while dramatically improve the alphabet size.

We apply this improvement to the constructions of [1], [9] and [7]. All these constructions are of the form $G \circ C$, and differ in the actual code C used, and the actual properties required from the expanding graph G used.

1.2. **On the Dispersers and Extractors that We Use.** We now survey the actual dispersers/extractors[1] G that we use in the various $G \circ C$ constructions. As suggested above we are looking for dispersers with optimal entropy loss and small degree. The first disperser we consider, denoted, G_{opt} has the best entropy loss and degree possible as follows from a lower bound and a matching upper bound for dispersers shown by [11]. This disperser, however, is only shown to exist using the probabilistic method, and there is no known explicit construction for it.

Lemma 1 implies that in order to achieve relative distance $(1-\epsilon)$ for $G \circ C$, we need a disperser which expands every set of constant fraction ϵ. In terms of expanding graphs we need a disperser/extaractor for the high min–entropy range. The recent extractor analogue of the zig-zag product due, to [13] gives good constructions for the high min–entropy range. We consider three constructions based on the zig-zag scheme. The first one is a disperser, denoted $G_{D_{opt}}$. $G_{D_{opt}}$ has an optimal entropy loss and near optimal degree. This construction uses two optimal sub components which need to be found by an exhaustive search, which takes $2^{\frac{1}{\epsilon}} polylog(N)$ time, where ϵ and N are the error and input length of the disperser $G_{D_{opt}}$. The second zig-zag based construction we use, denoted, $G_{E_{opt}}$ is an extractor with optimal entropy loss and near optimal degree. As with $G_{D_{opt}}$ the construction uses two optimal sub components which need to be found in time $2^{\frac{1}{\epsilon}} polylog(N)$. $G_{E_{opt}}$ and $G_{D_{opt}}$ are referred to semi-explicit for the exudative search they require. The last zig-zag based construction we use, denoted $G_{explicit}$, has optimal entropy loss but a bigger degree, however it is explicit and uses the extractors of [12] as sub components.

Finally, we consider $G_{balanced}$, the balanced disperser that appears in [1], [7], and in some of the constructions in [9]. This disperser, which is based on Ramanujan graphs has relatively large degree and sub optimal entropy loss. The exact parameters of these dispersers including construction times and computation times are given in section 3.6.

In all of our improvements we improve on a construction which uses either $G_{balanced}$ or a balanced extractor. Whenever $G_{balanced}$ is used, we examine what improvement we get replacing it with G_{opt}, $G_{D_{opt}}$, and $G_{explicit}$. Although $G_{explicit}$ is an extractor, having stronger properties

[1]The exact definition of extractros is given in section 3.4

than a disperser and thus larger degree and entropy loss, it is explicit and in most cases still achieves major improvement. Whenever a balanced extractor is used we find what happens when replacing it with $G_{E_{opt}}$, and with $G_{explicit}$.

2. The Improvement in Specific Applications

Our improvement can be demonstrated in the following constructions:

(1) The construction of [1] which composes Justesen code with a balanced disperserto give the first explicit constant rate code with arbitrarily large relative distance. [1] were also the first to define the composition of the form $G \circ C$.
(2) The construction of [1] leaves open the problem of decoding the code. [9] presents an error correcting code $G \circ C$ with explicit unique decoding, by taking C to be a list decodable code.
(3) List decodable codes with various range of parameters. We consider three constructions of list decodable codes with various parameters' tradeoffs.

2.1. **Error Correcting Codes.** We begin with the construction of asymptotically good error correcting codes of [1]. The construction of error correcting codes has the two combinatorially conflicting goals of simultaneously increasing the rate and the relative distance. A basic lower bound, known as the singleton bound, states that if C is a $(N, rN, \delta N)_q$ code then:

$$r \leq 1 - \delta + \frac{1}{N} \tag{1}$$

The only codes achieving the singleton bound are Reed-Solomon codes having:

$$r(\delta) \geq 1 - \delta \tag{2}$$

However, the alphabet size of these codes is at least the block length of the code. A probabilistic argument shows that for any alphabet size prime power q, there exist linear codes having:

$$r(\delta) \geq 1 - H_q(\delta) \tag{3}$$

This bound is known as the Gilbert-Varshamov bound. Thus, it is natural to concatenate Reed-Solomon codes with a code achieving the Gilbert-Varshamov bound, which as an inner code can be found in an exhaustive search in polynomial time. This gives the Zyablov bound:

$$R_{Zyablov}(\delta, q) \equiv max_{\delta \leq \mu \leq 1-\frac{1}{q}}(1 - H_q(\mu))(1 - \delta/\mu) \tag{4}$$

Although polynomial, the time needed for the exhaustive search above is of the form $N^{f(\mu)}$, where N is the concatenated code length and $lim_{\mu\rightarrow 1-\frac{1}{q}} f(\mu) = \infty$. Thus, the construction time of the above code is dependant on δ. [15] give a code with construction time which is independent of δ. The rate function of [15] satisfies:

$$R_{SKHN}(\delta) \geq max_{\delta\leq\mu\leq 1-\frac{1}{q}}(1 - H_q(\mu))\left(1 - \frac{\delta}{\mu}\left(1 + \ln\frac{\mu}{\delta}\right)\right) \tag{5}$$

[1] shows how to construct a code with arbitrarily large relative distance δ, rate $\Omega(1-\delta)$ and alphabet of size $2^{O(\frac{1}{1-\delta})}$. This construction is demonstrated in the following ways:

(1) Approaching the singleton bound. When q tends to infinity, the rate function of [1] has the form of $r(\delta) \geq \gamma_0(1-\delta)$. This resembles the singleton bound except the γ_0 factor.
(2) Beating the Zyablov bound for large alphabet size. It turns out that the rate function of [1] beats the Zyablov bound for large alphabets.
(3) Concatenation with codes of fixed alphabet size. When concatenating the construction of [1] with a code having some fixed alphabet size, the over all rate beats the rate function of the construction of [15] for very small rates.

We next show how replacing the balanced disperser with an unbalanced disperser can improve the alphabet size above to $2^{O(\log^2(\frac{1}{1-\delta}))}$. We also show how this improvement implies an improvement on the above three issues.

2.1.1. *Improving the Alphabet Size.* [1] give a construction of asymptotically good error correcting codes over large alphabets. We show how replacing the balanced expander used in their construction can be improved by using an unbalanced disperser:

Theorem 1. *For every relative distance $\delta < 1$, there exists an explicitly constructible family of codes of rate $\Omega(1-\delta)$ over alphabet of size:*

- $2^{O(\frac{1}{1-\delta})}$, *when using $G_{balanced}$ as in [1].*
- $2^{O(\log(\frac{1}{1-\delta}))}$, *when using G_{opt}.*
- $2^{O(\log^2(\frac{1}{1-\delta}))}$, *when using $G_{D_{opt}}$.*
- $2^{2^{polyloglog(\frac{1}{1-\delta})}}$, *when using $G_{explicit}$*

2.1.2. *Approaching the singleton bound.* The construction of [1] construction is of the form $G \circ C_{Jus}$, where C_{Jus} is a Justesen code[2] and G is the balanced disperser, $G_{balanced}$ above. They show:

Theorem 2. *There exists positive constants γ_0, γ_1, such that for every $\delta > \delta_{min}(\gamma_0)$, there exists $q_{min}(\delta)$, such that for every $q > q_{min}$ the rate function of the construction satisfies:*

$$R(\delta, q) > \gamma_0(1-\delta) - \frac{\gamma_1}{\log_2 q} \tag{6}$$

We show that by using an unbalanced disperser we get:

Theorem 3. *There exists a positive constant γ_0, such that for every $\gamma_1 > 0$, there is $\delta_{min}(\gamma_1)$ such that for every $\delta > \delta_{min}$, there exists $q_{min}(\delta)$, such that for every $q > q_{min}$ the rate function of the construction satisfies (6)*

For large alphabets, the rate function (6) resembles the singleton bound

$$R(\delta) \leq (1-\delta)$$

but with γ_0 in front of $(1-\delta)$ instead of 1. We show that the constant γ_0, can be improved when using an unbalanced disperser as shown below. Another difference relates to γ_1. While in theorem 3, γ_1 can be arbitrarily small, in the original construction it is a constant greater than 0.

Disperser used	γ_0	γ_1
$G_{balanced}$ [1]	≈ 0.021	> 0.58
G_{opt}	≈ 0.0605	$o(1)$
$G_{D_{opt}}$	≈ 0.0427	$o(1)$

Thus, we achieve rate which doubles or triples the rate achieved in [1].

2.1.3. *Beating the Zyablov bound for large alphabets.* [1] show that for large alphabet size (6) lies above the Zyablov bound (4). We show:

Theorem 4. *The rate function $R_{G\circ C_{Jus}}(\delta, q)$ lies above the Zyablov bound for alphabet size q of:*

- $2^{\Theta(\frac{1}{1-\delta})}$, *when using $G_{balanced}$ ([1]).*
- $poly(\frac{1}{1-\delta})$, *when using G_{opt}.*
- $2^{\Theta(\log^2(\frac{1}{1-\delta}))}$, *when using $G_{D_{opt}}$.*
- $\widetilde{plog}(\frac{1}{1-\delta})$, *when using $G_{explicit}$, where $\widetilde{plog}(x) = 2^{2^{polyloglog(x)}}$*

Thus, we beat the Zyablov bound for much smaller alphabet size.

[2] We elaborate on Justesen code in section 3.2

2.1.4. *Concatenation With Codes of Fixed Alphabet Size.* [1] show that concatenating $G \circ C$ with an inner code having fixed alphabet size, they get:

Theorem 5. *There exist constant γ_0, and $\delta_{min}(\gamma_0)$ such that for every $\delta > \delta_{min}$, and prime power q, there is an explicitly constructible family of codes, with rate function satisfying:*

$$R(\delta, q) \geq \max_{\delta \leq \mu \leq 1-\frac{1}{q}} \gamma_0 (1 - H_q(\mu))(1 - \frac{\delta}{\mu}) \tag{7}$$

We show that using an unbalanced disperser we have:

Theorem 6. *There exists constant γ_0 such that for every $0 < \delta < 1$, and prime power q, there is a code family with rate function satisfying (7).*

Using an unbalanced disperser improves on the γ_0 constant as shown below:

Disperser used	γ_0
$G_{balanced}$ [1]	0.0225
G_{opt}	0.0759
$G_{D_{opt}}$	0.0585

2.2. **Error Correcting Codes with Explicit Decoding Procedure.** The construction $G \circ C$ above of [1] gives codes with arbitrarily large relative distance δ, having rate $\Omega(1-\delta)$ for large alphabets. It is not clear, however, how to decode such codes. Denoting $\epsilon = 1 - \delta$, [9] give an efficient decoding procedure for an error correcting code $G \circ C$ of relative distance $1 - \epsilon$, having rate $\Omega(\epsilon)$. The decoding is achieved by taking C to be a list decodable code, and G to be a balanced expanding graph with strong mixing property. We show that replacing the balanced graph with an unbalanced extractor the alphabet size be improved. Specifically, we have:

Theorem 7. *For any $\frac{1}{8} > \epsilon > 0$ there is an explicitly specified code family with rate $\Omega(\epsilon)$, relative distance at least $(1-\epsilon)$ and alphabet size $|\Sigma|$ given by:*

Extractor Used	$\lvert\Sigma\rvert$	*Ref*
Balanced Ramanujan Graph	$2^{O(\frac{1}{\epsilon})}$	[9] *Theorem 8*
$G_{E_{opt}}$	$2^{O(\log^2(\frac{1}{\epsilon}))}$	*Section 7.1*
$G_{explicit}$	$\widetilde{plog}(\frac{1}{\epsilon})$	*Section 7.1*

Thus, we dramatically decrease the alphabet size with respect to [9]. We remark that if one could explicitly construct an optimal extractor the alphabet size could be improved to $(\frac{1}{\epsilon})^{O(1)}$. The full details of the proof, including encoding, decoding and construction times are given in the appendix (section 7.1).

2.3. **High Noise List Decodable Codes.** [9] and [7] give three different constructions of high noise list decodable codes with varying trade-off between rate and decoding list size, as described in table 1. All constructions are of the form $G \circ C$, differing only in the code C used.

In high noise list decoding we let the relative number of errors be $(1-\epsilon)$, where $\epsilon > 0$ is arbitrarily small, and present the other parameters: rate, alphabet size and decoding list size as a function of ϵ and the block length of the code N. Recall that for high noise list decodable codes $r = O(\epsilon)$, $L = \Omega(\frac{1}{\epsilon})$, and $q = \Omega(\frac{1}{\epsilon})$.

The first variant we consider has optimal rate of $\Omega(\epsilon)$, but suffers a sub exponential decoding list size. This construction is from [9] and takes C to be a list recoverable code with constant rate and sub exponential decoding list size. The exact parameters, including encoding, decoding and construction times are given in the appendix (section 7.2, theorem 11).

The second variant from [7], which takes C to be a list recoverable code from arbitrary size, has an almost optimal rate of $\Omega(\frac{\epsilon}{\log^{O(1)}(\frac{1}{\epsilon})})$ and still suffers sub exponential decoding list size. However, this construction shows that if one could construct better extractors for low min–entropies (or list recoverable codes from arbitrary size with short decoding lists), then one could get an almost optimal rate with small decoding list size. The exact parameters including construction times are given in the appendix (section 7.4, theorem 13).

The third variant has sub optimal rate, but has the merit of optimal decoding list size. In this construction from [9], C is taken to be a list recoverable code with rate $\Omega(\epsilon)$ and $O(\frac{1}{\epsilon})$ decoding list size, trading a shorter decoding list with a worse rate. The exact parameters, including encoding, decoding and construction times are given in the appendix (section 7.3, theorem 12).

In all cases, we show how replacing the balanced expander with various unbalanced dispersers improve on the alphabet size of these constructions, as shown in table 1.

2.4. **Summary and the Rest of This Work.** Amplification using expanding graphs is a widely used technique in both coding and complexity theory. Our technical contribution is noting that for the case of error amplification of codes the expanding graph needed is actually an unbalanced disperser and that its entropy loss is a key parameter in analyzing such codes constructions. The results above show that when using such unbalanced dispersers with optimal entropy loss, the resulting alphabet size, and sometimes the rate can be improved.

In section 3 we give the necessary coding and expanding graphs background, elaborating on the Zig-Zag graph construction, which we use for constructing good dispersers. In section 4 we explain the inherent loss in the rate of the almost optimal rate construction from table 1. We show that this loss with respect to the optimal rate construction stems from the fact that each construction uses a different flavor of a list recoverable code. In section 5 we give the general structure of the various $G \circ C$ constructions, and in sections 6, 7 we give the detailed parameter analysis of each construction.

3. Preliminaries

We give the necessary background on codes and expanding graphs we use.

3.1. **Codes.** Error correcting codes were built to deal with the task of correcting errors in transmission over noisy channels. Formally, an $(N, n, d)_q$ error correcting code over alphabet Σ, where $|\Sigma| = q$, is a subset $C \subseteq \Sigma^N$ of cardinality q^n in which every two elements are distinct in at least d coordinates. n is called the dimension of the code, N the block length of the code, and d the distance of the code. If C is a linear subspace of $[\mathbb{F}_q]^N$, where Σ is associated with some finite field $\mathbb{F}_q$ we say that C is a linear code, and denote it $[N, n, d]_q$ code. From the definition we see that one can uniquely identify a codeword in which at most $\frac{d-1}{2}$ errors occurred during transmission. Moreover, since two codewords from Σ^N can differ in at most N coordinates, the largest number of errors from which unique decoding is possible is $N/2$.

This motivates the list decoding problem, first defined in [4]. In list decoding we give up unique decoding, allowing potentially more than $N/2$ errors, and require that there are only few possible codewords having some modest agreement with any received word. Formally, we say that an $(N, n)_q$ code C is (p, K)-list decodable, if for every $r \in \Sigma^N$, $|\{c \in C | \Delta(r, c) \leq pN\}| \leq K$, where $\Delta(x, y)$ is the number of coordinates in which x and y differ. That is, the number of codewords which agree with r on at least $(1 - p)N$ coordinates is smaller than K. We call the ratio n/N the rate of the code, and p the error rate.

In the high noise regime we let $p = 1 - \epsilon$, for $\epsilon > 0$ being very small. A simple probabilistic argument shows that $(1 - \epsilon, O(\frac{1}{\epsilon}))$-list decodable

rate	Decoding list size	alphabet size	Ref
Lower bound			
ϵ	$\frac{1}{\epsilon}$	$\frac{1}{\epsilon}$	
Optimal rate list decodable codes			
ϵ	$2^{N^{\gamma} \log(\frac{1}{\epsilon})}$	$2^{\epsilon^{-1} \log(\frac{1}{\epsilon})}$ $2^{\log^2(\frac{1}{\epsilon})}$ $2^{\log^3(\frac{1}{\epsilon})}$ $\widetilde{plog}(\frac{1}{\epsilon})$	[9] Section 7.2 Section 7.2 Section 7.2
Almost optimal rate list decodable codes - Using explicit extractors			
$\frac{\epsilon}{\log^{O(1)}(\frac{1}{\epsilon})}$	$2^{\sqrt{g(\epsilon) \cdot N \log(g(\epsilon) \cdot N)}}$	$2^{\epsilon^{-1} \log(\frac{1}{\epsilon})}$ $2^{\log^2(\frac{1}{\epsilon})}$ $2^{\log^3(\frac{1}{\epsilon})}$ $\widetilde{plog}(\frac{1}{\epsilon})$	[7] Section 7.4 Section 7.4 Section 7.4
Almost optimal rate list decodable codes - Assuming optimal extractors			
$\frac{\epsilon}{\log(\frac{1}{\epsilon})}$	$\frac{1}{\epsilon}$	$2^{\epsilon^{-1} \log(\frac{1}{\epsilon})}$ $2^{\log^2(\frac{1}{\epsilon})}$ $2^{\log^3(\frac{1}{\epsilon})}$ $\widetilde{plog}(\frac{1}{\epsilon})$	[7] Section 7.4 Section 7.4 Section 7.4
Sub optimal rate list decodable codes			
ϵ^2	$\frac{1}{\epsilon}$	$2^{\epsilon^{-1} \log(\frac{1}{\epsilon})}$ $2^{\log^2(\frac{1}{\epsilon})}$ $2^{\log^3(\frac{1}{\epsilon})}$ $\widetilde{plog}(\frac{1}{\epsilon})$	[9] Section 7.3 Section 7.3 Section 7.3

TABLE 1. *The list decoding parameters and the alphabet size improvements. For each construction we list the improvements achieved whe using* G_{opt}, $G_{D_{opt}}$ *and* $G_{explicit}$. $O(\cdot)$, $\Omega(\cdot)$ *notations were omitted for readability. All codes have* $(1-\epsilon)$ *relative fraction of errors.* N *is the block length of the code and* $g(\epsilon)$ *is a function dependent only on* ϵ. *The value* γ *is in the interval* $(0, 1]$. $\widetilde{plog}(x)$ *stands for* $2^{2^{polyloglog(x)}}$.

codes with $rate = \Omega(\epsilon)$, and $|\Sigma| = O(\frac{1}{\epsilon^2})$ exist. Also the rate must be $O(\epsilon)$, and $|\Sigma| = \Omega(\frac{1}{\epsilon})$.

The notion of list decodable codes can be generalized to that of list recoverable codes, where for each coordinate $i \in [N]$ there is some subset of $|\Sigma|$ of possibilities for explaining the received symbol in the

i^{th} coordinate. Formally, we say that a code $C \subset [\Sigma]^N$, is $(\delta, \alpha|\Sigma|, L)$-list recoverable if for every $S_1, \ldots, S_N \subset \Sigma$ of size $\alpha|\Sigma|$ each, there are at most L codewords $w \in C$, having at least δN coordinates $w_i \in S_i$. List decoding is list recovering having $\alpha|\Sigma| = 1$.

After discussing extractors in section 3.4 we will further generalize the notion of list recovering to that of list recovering from arbitrary size. As we will see this notion is equivalent to extractors.

List decodable codes, list recoverable codes and list recoverable codes from arbitrary size (defined in section 3.5) are used as the code C in the constructions $G \circ C$ of error correcting codes with efficient list decoding, and of list decodable codes with various ranges of parameters.

Finally, we say that a code is explicit if a codeword of the code can be computed in time polynomial in the code length.

3.2. **Justesen Code.** The construction $G \circ C$ of [1] uses Justesen code as the code C. A Justesen code has the advantage of a good relationship between relative distance and rate, while still being explicit. This is achieved by concatenating Reed-Solomon of appropriate rate with a Wozencraft ensemble of codes. Before stating the parameters of Justesen code we need the definition of the entropy function:

Definition 3. *For every $0 \leq x \leq 1$, the binary entropy function, denoted $H_2(x)$, is defined as:*

$$H_2(x) = x \log_2(\frac{1}{x}) + (1-x)\log_2(\frac{1}{1-x})$$

Moreover, $H_2(0)$, $H_2(1)$ are defined to be 0 at these points as $\lim_{x\to 0} H_2(x) = \lim_{x \to 1} H_2(x) = 0$. For every $0 \leq x \leq 1 - \frac{1}{q}$, we define:

$$H_q(x) = x \log_q(\frac{1}{x}) + (1-x)\log_q(\frac{1}{1-x}) + x\log_q(q-1) \tag{8}$$

Again, $\lim_{x\to 0} H_q(x) = 0$, and so we define $H_q(0) = 0$. It can be easily verified that $H_q(1-\frac{1}{q}) = 1$, and is concave and monotonically increasing in $[0, 1-\frac{1}{q}]$.

Theorem 8. *For every $\delta_0 < \frac{1}{2}$, and alphabet size q_0, large enough such that $H_q^{-1}(\frac{1}{2}) > \delta_0$, there exists an explicit family of codes with relative distance δ_0 over alphabet of size q_0, and rate:*

$$R_{Jus}(\delta_0, q_0) = \frac{1}{2}\left(1 - \frac{\delta_0}{H_{q_0}^{-1}(1/2)}\right) \tag{9}$$

For further analysis we get rid of the inverse entropy function appearing in the rate above. We begin by bounding the inverse entropy function $H_q^{-1}(\frac{1}{2})$:

Claim 1.

$$H_q^{-1}(\frac{1}{2}) \geq \frac{1}{2} - \frac{1}{\log_2 q} \tag{10}$$

Proof. (8) can be rewritten as:

$$\begin{aligned} H_q(x) &= \frac{H_2(x)}{\log_2 q} + x\log_q(q-1) \leq \\ &\leq \frac{1}{\log_2 q} + x \end{aligned}$$

Letting $x = \frac{1}{2} - \frac{1}{\log_2 q}$, we thus have, $H_q(x) \leq \frac{1}{2}$. Since $H_q(x)$ is monotonically increasing the claim follows. We remark that this bound is tight. It can be easily shown that:

$$H_q^{-1}(\frac{1}{2}) \leq \frac{1}{2} - \frac{1}{4\log_2 q}$$

□

Substituting (10) in (9) we get:

Corollary 1. *If C_{Jus} is a Justesen code over alphabet of size q_0, and relative distance δ_0, then:*

$$R_{Jus}(\delta_0, q_0) > \frac{1}{2} - \delta_0 - \frac{2\delta_0}{\log_2 q_0 - 2} \tag{11}$$

3.3. **A General Decoding Scheme.** We now give a description of a decoding procedure for constructions of the form $G \circ C$, which is common to all the decoding procedures we consider later on. This decoding procedure was used in the various constructions of [9], [7]. The procedure interprets the i^{th} symbol of a received word - a symbol from the alphabet of $G \circ C$ - as a list of 'votes' saying what i thinks are the symbols from the smaller alphabet of C, in the coordinates neighboring to i in G.

Formally, let G be an expanding graph $G : [L] \times [T] \to [N]$ and $C \subset \Sigma^N$. The decoding procedure for $G \circ C$ takes a word $w \in (\Sigma^T)^L$, and constructs N subsets of Σ: $S_1, \ldots, S_N$ in the following way: for each $l \in [L]$, and each $t \in [T]$ we add to $S_{G(l,t)}$, the symbol $w_{l,t}$. See figure 2A. We refer to this procedure as a voting procedure, as every coordinate of w on the right votes for what it thinks are the symbols that should be in each of its neighbors on the left. A coordinate $i \in [N]$ having degree D, can get up to D different votes. Had the word w been a legitimate codeword of $G \circ C$, all votes were identical. See figure 2B. Different decoding strategies use the N sets in different ways, as described in section 5.

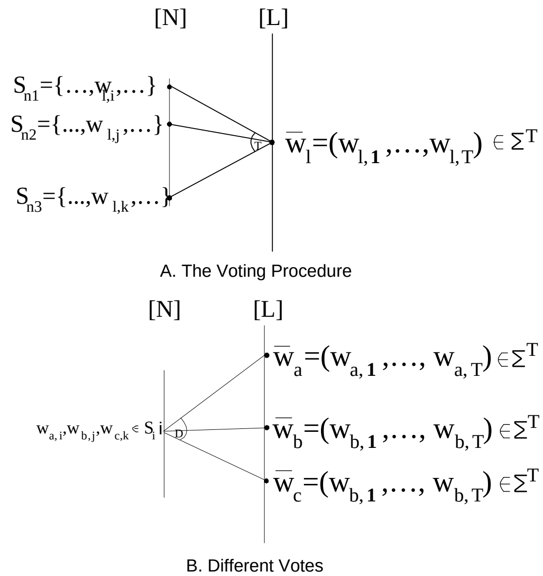

FIGURE 2. *A. The Voting Procedure.* $w_l \in \Sigma^T$*, is the* l^{th} *coordinate of some word* $w \in (\Sigma^T)^L$*. If* $n_1 \in [N]$ *is the* i^{th} *neighbor of* $l \in [L]$*,* S_{n1} *contains the symbol* $w_{l,i}$*. Similarly,* n_2*,* n_3 *are the* j^{th}*, and* k^{th} *neighbors of* l*, adding the 'votes'* $w_{l,j}$*, and* $w_{l,k}$ *to* S_{n2}*,* S_{n3} *accordingly. B. Different Votes.* $a, b, c \in [L]$ *are all neighbors of* i*, thus contributing their votes to* S_i*. Had* w_a, w_b*, and* w_c *been coordinates of a legitimate codeword of* $G \circ C$*, the votes were consistent, meaning* $w_{a,i} = w_{b,j}, w_{c,k}$*.*

We now analyze the complexity it takes to perform the encoding and decoding of the amplification procedure. Let G be a $[L] \times [T] \to [N]$ disperser. Assume that given $x \in [L]$, and $y \in [T]$, computing $G(x, y)$ takes time t. For the encoding procedure, we need to iterate over all elements in $[L]$ and for each element to find all its $[T]$ neighbors. Thus, the encoding time is $LT \cdot t$. For the decoding procedure described above, we need again $LT \cdot t$ time. We mention that in order to keep all sets $S_1, \ldots, S_N$ we need also $LT \log q$ space, where q is the alphabet size of the code C, used in $G \circ C$. The exact resources needed for the various dispersers we use are given below.

3.4. **Expanding Graphs.** Expanding graphs are highly connected graphs, but nevertheless sparse. There are two major ways to define the expansion property of these graphs. The weaker property of expansion states that every subset of the vertices X is expanded by some factor

$C > 1$, meaning the size of the neighbor set of X is at least $C|X|$. This property assures that if we start with a small subset X then after not too many expansion steps, we will visit almost all the vertices. This property is similar[3] to the property of dispersers defined below. The stronger property of mixing (see, [2], Chap 9) states that the number of edges between any two subsets of vertices is close to the relative number of edges leaving these subsets. This property assures that if we start with a small subset of vertices X then after not too many steps where in each step we proceed from X to its neighboring set, not only we visit almost all vertices, but each vertex is visited more or less the same number of times. This property is similar to the property of extractors defined below. Thinking of our graphs as bipartite graphs with regular left degree, we turn to the weaker definition of dispersers:

Definition 4. *(Dispersers) $G : [L] \times [T] \to [N]$ is a (K, ϵ)-disperser if for every $X \subseteq [L]$, $|X| \geq K$ we have $|\Gamma_G(X)| \geq (1 - \epsilon)N$. The entropy loss of the disperser is $\Lambda_G = \frac{KT}{N}$. The disperser is explicit if $G(x, y)$ can be computed in time polynomial in the input length, i.e., polynomial in $\log L + \log T$.*

Thus, the disperser assures that each small subset of $[L]$ sees almost all $[N]$. K is referred to as the min–entropy for which the disperser assures the required expansion. K vertices have at most KT neighbors, while the expansion property assures almost N neighbors. Thus, the entropy loss $\Lambda_G = \frac{KT}{N}$ gives some measurement of the quality of the disperser's expansion. It is useful to note that the expansion property of dispersers works for both sides, as demonstrated in the following lemma:

Lemma 2. *(Reverse expansion) If $G : [L] \times [T] \to [N]$ is a (K, ϵ)-disperser then for any subset $Y \subset [N]$, $|Y| \geq \epsilon N$, we have $|\Gamma_G(Y)| \geq L - K$.*

Proof. Any $X \subset [L]$, $|X| \geq K$ has $|\Gamma_G(X)| \geq (1 - \epsilon)N$. This implies that for any subset $Y \in [N]$, $|Y| \geq \epsilon N$ there can be a set of size at most K in $[L]$ missed by Y. Thus, $|\Gamma_G(Y)| \geq L - K$ $\square$

For the stronger definition of extractors, we need the following: A probability distribution D on Ω is a function $D : \Omega \to [0, 1]$, satisfying $\Sigma_{x\in\Omega} D(x) = 1$. For an integer M we define U_M as the uniform distribution over $[M]$, meaning $U_M(x) = \frac{1}{M}$ for every $x \in [M]$. The statistical

[3]Expanders assure the expansion of every small enough set whereas dispersers assure the expansion every large enough set.

distance between two distributions D_1, D_2, denoted $|D_1 - D_2|$ is:

$$\frac{1}{2}\Sigma_{x\in\Omega}|D_1(x) - D_2(x)| = \max_{S\subset\Omega}|D_1(S) - D_2(S)|$$

We say that D_1 and D_2 are ϵ-close if $|D_1 - D_2| < \epsilon$. We are now ready for the extractor definition:

Definition 5. *(Extractors) $E : [L] \times [T] \to [N]$ is a (K, ϵ)-extractor if for every $X \subseteq [L]$, $|X| \geq K$, the distribution of $E(x, y)$, is ϵ-close to U_N, where x is taken uniformly at random from X and y is taken uniformly at random from $[T]$. The entropy loss of the extractor is $\frac{KT}{N}$. ϵ is called the extractor error. An extractor is explicit if $E(x, y)$ can be computed in time polynomial in the input length, i.e., polynomial in $\log L + \log T$.*

As opposed to the definition of dispersers the condition $E(x, y)$ is ϵ-close to U_M states that not only every element in $[M]$ is sampled, but all elements in $[M]$ are sampled about the same number of times. Thus, any extractor is also a disperser having the exact same parameters. K is called the min–entropy of the extractor. A stronger definition of extractors demands that the output distribution stays close to uniform even if the random value of y is revealed.

Definition 6. *(Strong Extractors) $E : [L]\times[T] \to [N]$ is a (K, ϵ)-strong extractor if for every $X \subseteq [L]$, $|X| \geq K$, the distribution $y \circ E(x, y)$ is ϵ-close to $U_{[T]\times[N]}$, where x is taken uniformly at random from X and y is taken uniformly at random from $[T]$. The entropy loss of the strong extractor is $\frac{K}{N}$. The strong extractor is explicit if $E(x, y)$ can be computed in time polynomial in the input length, i.e., polynomial in $\log L + \log T$.*

As mentioned before the property of extractors is closely related to that of mixing. It is immediate from the definition of extractors that:

Fact 1. *(Extractors mixing property) If $E : [L] \times [T] \to [N]$ is a (K, ϵ)-extractor, then for every $S \subseteq [N]$, and every $X \subset [L]$, $|X| \geq K$, we have:*

$$\left|\frac{|\Gamma_E(X) \cap S|}{|X|T} - \frac{|S|}{N}\right| < \epsilon$$

where

$$\Gamma_E(X) = \{E(x, i)|x \in X, i \in [T]\}$$

If E above is strong we get for every $S \subseteq [T] \times [N]$, and every $X \subset [L]$, $|X| \geq K$

$$\left|\frac{|\Gamma_E(X) \cap S|}{|X|T} - \frac{|S|}{T \cdot N}\right| < \epsilon$$

where

$$\Gamma_E(X) = \{(i, E(x,i)|x \in X, i \in [T]\}$$

Another way to write the mixing property of strong extractors is: For every $S \subset [T] \times [N]$, there are at most K elements $x \in [L]$, for which:

$$\left|\frac{|\Gamma_E(x) \cap S|}{T} - \frac{|S|}{TN}\right| > \epsilon \tag{12}$$

Just as with the reverse expansion of dispersers, extractors have reverse mixing, as demonstrated in the next lemma:

Lemma 3. *(Reverse Mixing) if $E : [L]\times[T] \to [N]$ is a (K,ϵ)-extractor with regular right degree $Q = \frac{LT}{N}$ then for every $S \subset [N]$, and $X \subset [L]$, $|X| \geq K$, we have:*

$$\left|\frac{|\Gamma_E(S) \cap X|}{|S|Q} - \frac{|X|}{L}\right| < \epsilon \cdot \frac{N}{|S|} \cdot \frac{|X|}{L}$$

Proof. By the mixing property of extractors we have: $\forall S \subseteq [N]$, $\forall X \subseteq [L]$, $|X| \geq K$ it holds that:

$$\left|\frac{|\Gamma_E(X) \cap S|}{T|X|} - \frac{|S|}{N}\right| < \epsilon$$

Multiplying and dividing by $\frac{Q|S|}{T|X|}$, and noting that $|\Gamma_E(X) \cap S| = |\Gamma_E(S) \cap X|$ we get:

$$\frac{Q|S|}{T|X|} \cdot \left|\frac{|\Gamma_E(S) \cap X|}{Q|S|} - \frac{T|X|}{QN}\right| < \epsilon$$

Substituting $Q = \frac{LT}{N}$ the lemma follows. □

Finally, we mention that [11] give the following lower bounds, which have matching upper bound for extractors and strong extractors: if $E : [L] \times [T] \to [N]$ is a (K,ϵ)-(strong) extractor, then:

$$T = \Omega(\frac{1}{\epsilon^2} \log \frac{N}{K}) \tag{13}$$

entropy loss:

$$\Lambda_G = \Omega(\frac{1}{\epsilon^2}) \tag{14}$$

3.5. **Extractors and List Recoverability from Arbitrary Size.** We now further generalize the notion of list recovering to that of list recovering from arbitrary size. Recall that a code C of block length N is $(\delta, \alpha|\Sigma|, L)$-list recoverable if for every $S_1, \dots, S_N \subset \Sigma$ of size $\alpha|\Sigma|$ each, there are at most L codewords $w \in C$, having at least δN coordinates $w_i \in S_i$. We say that the i^{th} coordinate of a codeword $C(x)$ agrees with some $S_i \subset \Sigma$ of arbitrary size, if $C(x)_i \in S_i$. Denoting $S = \bigcup_i \{S_i, i\}$, where $\{S_i, i\} = \{(x,i)|x \in S_i\}$, we say that the agreement of $C(x)$ with S, is the number of coordinates i having agreement with S_i. The list recovering property can be now thought of as having a small number of codewords having some **fixed** agreement (δN) with a set $S \subset \Sigma \times [N]$.

In list recovering from arbitrary size we demand that for each $S \subset \Sigma \times [N]$ there is a small number of codewords having relative agreement with S which is slightly more than the **proportional** size of S. Formally, A code $C \subset [\Sigma]^N$ is (L, ϵ) list recoverable from arbitrary size if for every $S \subseteq \Sigma \times N$, there are at most L codewords $C(x)$, for which $A_S(x) > (\frac{|S|}{N|\Sigma|} + \epsilon)N$, where $A_S(x) = \{i|(x_i, i) \in S\}$.

[16] have shown that the notion of list recoverability from arbitrary size is equivalent to that of a strong extractor. Intuitively, and using the notations of extractors and codes above, the mixing property for strong extractors states that for every subset $S \subseteq [T] \times [M]$ there are few vertices having relative number of neighbors in S larger than the relative size of S. In list recovering from arbitrary size there are few codewords having relative agreement with $S \subseteq \Sigma \times [N]$ larger than the relative size of S. Formally, [16] show:

Theorem 9. *If $E : [N] \times [D] \to [M]$ is a (L, ϵ)-strong extractor, then the code $C_E : [N] \to [M]^D$ defined by $\forall x \in [N]$, $C(x) = (E(x,1), \dots, E(x,D))$ is (L, ϵ)-list recoverable from arbitrary size. Conversely, if C_E is (ϵ, L) list recoverable from arbitrary size then E is a $(\frac{L}{\epsilon}, 2\epsilon)$-strong extractor.*

We can thus derive an upper bound on the rate of a list recoverable code from arbitrary size from the degree lower bound of strong extractors:

Lemma 4. *For every $\epsilon > 0$ if $C_E : [N] \to [q]^D$ is a (ϵ, L)-list recoverable code from arbitrary size and L is constant in N, then the rate of the code r_{C_E} satisfies:*

$$r_{C_E} = \frac{\log_2 N}{D \log q} = O(\frac{\epsilon^2}{\log q})$$

Proof. Theorem 9 implies that C_E is a $(\frac{L}{\epsilon}, 2\epsilon)$-strong extractor $E : [N] \times [D] \to [M]$. By the degree lower bound of strong extractors (13) we have $D = \Omega(\frac{1}{\epsilon^2}\log(\frac{\epsilon N}{L}))$. Taking L independent of N, the lemma follows. $\square$

3.6. **The Dispersers and Extractors Parameters.** We now elaborate on the exact parameters of the dispersers and extractors surveyed in section 1.2. The extractors and dispersers we mention are used as G in the various $G \circ C$ constructions appearing in this paper. In all cases we consider a (K, ϵ)-disperser/extractor $G : [L] \times [T] \to [N]$.

We also summarize in table 2 the parameters of the relevant graphs below using slightly different notations, which comply with the notations of [1] for ease of presentation.

3.6.1. *The Optimal Disperser G_{opt}.* Ta-Shma and Radhakrishnan [11] show that any disperser with parameters as above must have degree:

$$T = \Omega(\frac{1}{\epsilon}\log\frac{N}{K}) \tag{15}$$

and entropy loss:

$$\frac{KT}{N} = \Omega(\log\frac{1}{\epsilon}) \tag{16}$$

Probabilistically, [11] show a disperser with:

$$T = \frac{2}{\epsilon}(\ln\frac{N}{K} + 1) \tag{17}$$

and with entropy loss:

$$\frac{KT}{N} = 2(\ln(\frac{1}{\epsilon}) + 1) \tag{18}$$

The disperser we refer to as G_{opt} has degree and entropy loss, as in (17), (18). G_{opt} is used in all constructions (except the explicit decoding of [1] in section 2.2, where an optimal extractor is needed).

3.6.2. *The Balanced Disperser $G_{balanced}$.* We compare all constructions to those using $G_{balanced}$, based on Ramanujan graphs, having the parameters:

$$T \geq \frac{4(\frac{1}{\epsilon} - 1)}{\frac{K}{L}} \tag{19}$$

$$\Lambda = \frac{KT}{L} = 4(\frac{1}{\epsilon} - 1) \tag{20}$$

3.6.3. *The Zig-Zag Based Constructions $G_{D_{opt}}$, $G_{E_{opt}}$, and $G_{explicit}$.* As mentioned above the graphs G in the $G \circ C$ constructions are dispersers/extractors for the high min–entropy range. Such graphs can be constructed using the recent zig-zag product scheme of [13] tailored for this range.

Zig-Zag preliminaries: We begin with the definition of min–entropy. A random variable X distributed over $\{0,1\}^n$ is said to have $k \leq n$ bits of min–entropy, denoted $H_\infty(X) = k$, if for every $x \in \{0,1\}^n$, $Pr[X = x] \leq 2^{-k}$. Min–entropy is thus a measurement of the amount of randomness in a weak source which is not uniformly distributed. An extractor is a function which takes a weak random source X having some min–entropy $k < n$ and transforms it to almost purely (ideally k) random bits. Formally (using min–entropy term):

Definition 7. *A function $Ext : \{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m$ is a (k,ϵ)-extractor if for every X distributed over $\{0,1\}^n$, having k min–entropy, $Ext(X, U_d)$ is ϵ-close to U_m.*

[14] have shown that no deterministic function can perform such an extraction. Thus, the extractors we consider take as input (apart form the weak source) an additional random seed of pure randomness to perform the extraction. Let us recall that for the application of codes, we consider agreement sets of size ϵL out of L, where $\epsilon > 0$ is some constant independent of L. In terms of min–entropy this is like having a source with $k = \log L - \log(\frac{1}{\epsilon})$ bits of min–entropy out of $n = \log L$. Defining $\Delta = n - k = \frac{1}{\epsilon}$, we say that the source has Δ min–entropy deficiency. Thus, for the error amplification of codes we need an extractor for sources with constant min–entropy deficiency. As pointed out by [3] any source X of length n having Δ deficiency can be thought of two 'almost independent' sources each having Δ deficiency. Formally, [3] show:

Lemma 5. *Let X be a random source distributed over $\{0,1\}^n$, having Δ deficiency. For every $\epsilon > 0$ and every n_1, n_2, such that $n_1 + n_2 = n$, X is ϵ-close to a block source $X_1 \circ X_2$ ($\circ$ denotes string concatenation), where X_1 has Δ deficiency, and conditioned on any value of X_1, X_2 has $\Delta + 2\log(\frac{1}{\epsilon})$ deficiency.*

[14] implies that no deterministic extraction is possible for such a block source, even if Δ is small. [10] give a simple extractor for block sources. The idea is to take a relatively short truly random seed, which is used to extract the randomness from X_2. The extracted randomness is then used to extract the randomness from X_1. The smaller Δ is, the smaller the truly random seed can be.

However, this idea looses Δ min–entropy 'by definition'. The zig-zag scheme of [13], overcomes this loss. We now sketch the zig-zag scheme, see figure 3.

Let X be a source distributed over $\{0,1\}^n$, having Δ min-entropy deficiency. Let $\epsilon > 0$, $n_1 + n_2 = n$, with X_1, X_2 as in the lemma above. Let $E_2 : \{0,1\}^{n_2} \times \{0,1\}^{d_2} \rightarrow \{0,1\}^{m_2}$, be a $(n_2 - \Delta - 2\log(\frac{1}{\epsilon}), \epsilon)$ extractor. E_2 uses d_2 truly random bits to extract m_2 random bits from X_2. Let $E_1 : \{0,1\}^{n_1} \times \{0,1\}^{m_2} \rightarrow \{0,1\}^{m_1}$, be a $(n_2 - \Delta, \epsilon)$ extractor, having the following property: E_1 can be extended to a pair of functions $\langle E_1, C_1 \rangle : \{0,1\}^{n_1} \times \{0,1\}^{m_2} \rightarrow \{0,1\}^{m_1} \times \{0,1\}^{n_1+m_2-m_1}$, such that $\langle E_1, C_1 \rangle$ is a $1 - to - 1$ mapping.

E_1 uses the output bits of E_2 to extract m_1 random bits from X_1. The point is to note that:

$$\langle E_1, C_1 \rangle (X_1, E_2(X_2, Y)) \circ X_2 \circ Y = Z_1 \circ Z_2 \circ X_2 \circ Y \tag{21}$$

where $Y = U_{d_2}$ and Z_1, Z_2 denote the output distribution of $\langle E_1, C_1 \rangle$, is a $1 - to - 1$ mapping. Now, since Z_1 is close to uniform, we expect that given the m_1 random bits of Z_1 extracted from X_1, there are still almost $n - \Delta + d_2 - m_1$ random bits in $Z_2 \circ X_2 \circ Y$. Thus, we apply a third extractor using fresh random bits on $Z_2 \circ X_2 \circ Y$. This extractor needs to be a $((n - \Delta + d_2 - m_1), \epsilon)$-extractor $E_3 : \{0,1\}^{(n_1+m_2-m_1)+n_2+d_2} \times \{0,1\}^{d_3} \rightarrow \{0,1\}^{m_3}$. Thus, the total entropy loss of the process is only the entropy loss of E_3 and the total amount of random bits used is $d_2 + d_3$.

The Zig-Zag disperser: For most of our applications we only need a disperser with optimal entropy loss. Seemingly, the natural thing to do would be taking E_1, E_2 and E_3 to be dispersers. Doing so will reduce both the amount of randomness needed and the entropy loss which is smaller for a disperser. However, if for example E_2 is a disperser the argument above fails. To see why we note that $X_1 \circ X_2$ is a block source in which X_1 conditioned on X_2 might have very small or even no min–entropy. Thus, if E_2 is a disperser and $E_2(X_2, Y)$ is far from uniform, X_1 conditioned on $E_2(X_2, Y)$ might have very small or even no min–entropy. This in turn implies that although $\langle E_1, C_1 \rangle$ is a $1 - to - 1$ mapping we cannot argue about the amount of min–entropy in $Z_2 \circ X_2 \circ Y$ conditioned on Z_1 as we did before. Similarly, if we take E_1 to be a disperser, Z_1 might be far from uniform and again we cannot argue about the amount of min–entropy in Z_2, given Z_1.

We thus take only E_3 to be a disperser. The above argument is identical, only we 'extract' the $n - \Delta + d_2 - m_1$ bit of min–entropy from $Z_2 \circ X_2 \circ Y$ in a 'disperser manner'. This yields Z_3 which is not close to uniform but close to have full support. This is exactly what

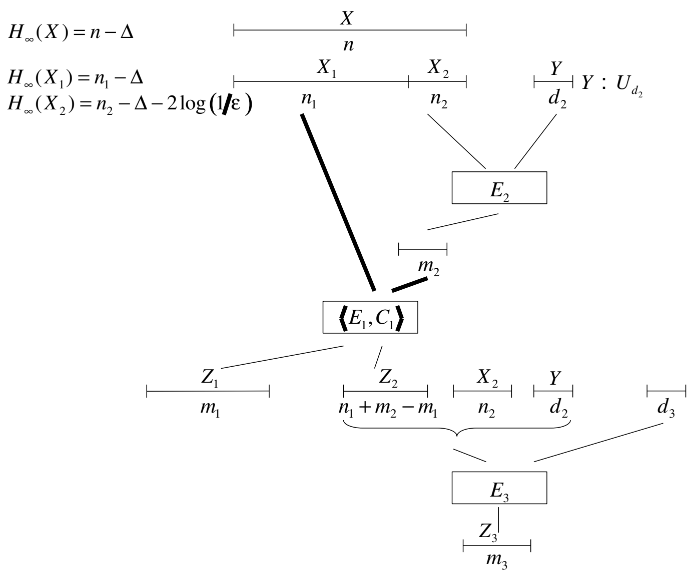

FIGURE 3. *The zig-zag scheme*

we need as we want a disperser and not an extractor. Taking E_3 to be a disperser, the entropy loss of the scheme above will be the entropy loss of a disperser which is much better than that of an extractor. Also d_3 can be much smaller for a disperser.
The parameters: We now give the exact parameters of the zig-zag based constructions which we use. In all three constructions E_1 is the extractor of [5] based on an expander random walk:

Theorem 10. *For any $\epsilon > 0$ and $0 < k < n$ there exists an explicit $(n - \Delta, \epsilon)$-extractor $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$, where $d = \Delta + 2\log(\frac{1}{\epsilon}) + 2$*

Using an appropriate expander for the construction of E_1 (e.g. a Caley graph), it can be easily extended to be a $1 - to - 1$ mapping $\langle E_1, C_1 \rangle : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n \times \{0,1\}^d$.

For $G_{D_{opt}}$ we take E_2 to be an optimal extractor and E_3 to be an optimal disperser. $G_{D_{opt}}$ is used in the error correcting codes construction from section 2.1 and all constructions of list decodable codes from section 2.3. This construction actually appears in [13] lemma 6.13 only with E_3 being an optimal extractor.

Lemma 6. *([13] corollary 6.13, replacing E_3 with an optimal disperser) For any $1 \le K \le N$ and $\epsilon > 0$, there exists a (K, ϵ)-Disperser G :*

$[L]\times[T]\to[N]$ with

$$T = 512(\frac{1}{\epsilon})^3(\log(\frac{L}{K}) + \log(\frac{1}{\epsilon}) + 2)^2 \tag{22}$$

$$\Lambda = 2(\ln(\frac{4}{\epsilon}) + 1) \tag{23}$$

Given $x\in[L]$, and $y\in[T]$ computing $G(x,y)$ takes $25\log^2 L$ time. The construction time of the disperser is $2^{(\frac{L}{K})^{O(1)}}\cdot poly\log L$, and it can be represented in $O((\frac{L}{K}+\log(\frac{1}{\epsilon}))^2)$ space.

For $G_{E_{opt}}$ we take E_2 and E_3 to be optimal extractors, this construction is given in [13] lemma 6.13. $G_{E_{opt}}$ is used in the construction of error correcting codes having efficient decoding from section 2.2.

Lemma 7. *([13] corollary 6.13) For any $1\le K\le N$ and $\epsilon>0$, there exists a (K,ϵ)-Extractor $E:[L]\times[T]\to[N]$ with*

$$T = 1024(\frac{1}{\epsilon})^4(\log(\frac{L}{K}) + \log(\frac{1}{\epsilon}) + 2)^2 \tag{24}$$

$$\Lambda = 16(\frac{1}{\epsilon})^2\ln 2 \tag{25}$$

Given $x\in[L]$, and $y\in[T]$ computing $G(x,y)$ takes $25\log^2 L$ time. The construction time of the disperser is $2^{(\frac{L}{K})^{O(1)}}\cdot poly\log L$, and it can be represented in $O((\frac{L}{K}+\log(\frac{1}{\epsilon}))^2)$ space.

We emphasize that although $G_{D_{opt}}$ and $G_{E_{opt}}$ use optimal subcomponents these subcomponents are small enough so that the construction time is exponential in $\frac{L}{K}$. Recall that for our applications $\frac{L}{K}=\frac{1}{\epsilon}$, where $1-\epsilon$ is a constant representing the decoding radius/minimum distance of the codes.

For $G_{explicit}$ we take E_2 and E_3 to be the optimal entropy loss extractors of [12]. This construction is explicit, however its entropy loss and degree are inferior to previous constructions. $G_{explicit}$ is used throughout all the constructions we consider.

Lemma 8. *([13] Theorem 6.12 using the explicit extractors of [12] Theorem 4) For any $1\le K\le L$ and $\epsilon>0$, there exists a (K,ϵ)-Extractor $E:[L]\times[T]\to[N]$ with degree:*

$$T = 2^{O(log^3(\frac{1}{\epsilon}\log(\frac{L}{K}))} \tag{26}$$

and entropy loss:

$$\Lambda = 128(\frac{1}{\epsilon^2}) \tag{27}$$

Graph	Λ	T	Ref
G_{opt}	$2(\ln(\frac{1}{\delta_0})+1)$	$\frac{2}{\delta_0}(\ln(\frac{1}{1-\delta})+1)$	[11]
$G_{D_{opt}}$	$2(\ln(\frac{4}{\delta_0})+1)$	$512(\frac{1}{\delta_0})^3(\log(\frac{1}{1-\delta})+\log(\frac{1}{\delta_0})+2)^2$	[13]
$G_{explicit}$	$128(\frac{1}{\delta_0})^2$	$2^{O(\log^3(\frac{1}{\delta_0}\log(\frac{1}{1-\delta})))}$	[13]
$G_{balanced}$	$4(\frac{1}{\delta_0}-1)$	$\frac{4(\frac{1}{\delta_0}-1)}{1-\delta}$	[1]

TABLE 2. *The dispersers' and extractor's parameters we use to improve on* [1]. *The parameters are quoted for $G : [L] \times [T] \to [N]$, which is a $((1-\delta)L, \delta_0)$-disperser/extractor. δ is the relative distance of $G \circ C$. δ_0 is relative distance of C and N is the block length of C.*

Given $x \in [L]$, and $y \in [T]$ computing $E(x, y)$ takes $25\log^2 L + O((\log(\frac{L}{K}) + \log(\frac{1}{\epsilon}))^3)$ *time. The construction time of the extractor is poly*$((\frac{1}{\epsilon})(\frac{L}{K}))$

4. COMPARING THE LIST DECODABILITY VARIANTS

Comparing the optimal rate and almost optimal rate list decodable codes in table 1, we see that both have sub exponential decoding list size, and both have the same alphabet size[4]. To see why we loose on the rate, we observe that each construction uses a different variant of a list recoverable code. The optimal rate construction uses a list recoverable code whereas the almost optimal rate construction uses a list recoverable code from arbitrary size. We now compare the parameters of these codes.

type of the code	$\lvert\Sigma\rvert$	Decoding list size	error	rate
L.R.C.	$O(\frac{1}{\epsilon^2})$	sub exponential in N	$O(1)$	$\Omega(1)$
L.R.C. from arbitrary size	$O(\frac{1}{\epsilon})$	sub exponential in N	$O(1)$	$\Omega(\frac{1}{\log^{O(1)}(\frac{1}{\epsilon})})$

Where $\epsilon > 0$ is the agreement fraction of the resulting list decodable code. Thus, we loose on the rate when using list recoverable code from arbitrary size.

Furthermore, if we think of ϵ above as a parameter and compare the rate upper bound for list recoverable codes from arbitrary size with the probabilistic bound for list recoverable codes, both having decoding list of size $O(\frac{1}{\epsilon})$, we have:

[4]The benefit of the almost optimal rate construction is that when using optimal extractors for the left list recoverable from arbitrary size code, one can get optimal decoding list size.

type of the code	$\lvert\Sigma\rvert$	Decoding list size	error	rate
L.R.C.	$O(\frac{1}{\epsilon^2})$	$O(\frac{1}{\epsilon})$	$O(1)$	$\Omega(1)$
L.R.C. from arbitrary size	$O(\frac{1}{\epsilon})$	$O(\frac{1}{\epsilon})$	$O(1)$	$O(\frac{1}{\log(\frac{1}{\epsilon})})$

The mentioned rate upper bound of list recoverable codes from arbitrary size is immediate from the connection with extractors (as shown in section 3.5). The probabilistic bound for list recoverable codes is shown in [8] Corollary 9.3.

5. The Internal Structure of the Different Constructions

In this section we summarize the various ways in which we use the $G \circ C$ construction. In section 1 we give the example where C is a linear code, as in [1]. For the constructions in sections 2.2, 2.3 we need to take C either as a list decodable code or as a list recoverable code or as a list recoverable code from arbitrary size. As shown below, the choice of C determines the internal structure of the proof regarding the list decodability of the overall construction.

5.1. **Taking C to be a Linear Code.** For the asymptotically good error correcting codes [1] take C to be a linear code. The parameters of $G \circ C$ where C is a linear code are given in lemma 1. We now give the proof.

Lemma 1. *If $G : [L] \times [T] \longrightarrow [N]$ is a $(\epsilon L, \delta)$- disperser with entropy loss Λ, and if C is a $[N, rN, \delta N]_q$ code then $G \circ C$ is a $[L, \frac{r \cdot \epsilon}{\Lambda} L, (1-\epsilon)L]_{q^T}$ code*

Proof. The alphabet size of $G \circ C$ is immediate from the composition definition. By the composition definition the rate of $G \circ C$ is:

$$r \cdot \frac{N \log q}{L \log(q^T)} = r \cdot \frac{N}{LT} = \frac{r \cdot \epsilon}{\Lambda} \tag{28}$$

For the relative distance, let $C(x)$ be a codeword of C. C is linear with relative distance δ, and so there are at least δN coordinates which are not zero in $C(x)$. By the reverse expansion of dispersers (Lemma 2), these δN coordinates have at least $(1 - \epsilon)L$ neighbors in G, yielding $(1 - \epsilon)L$ coordinates different from zero in $G \circ C(x)$. Thus, from the linearity of $G \circ C$ it has relative distance $(1 - \epsilon)$. $\square$

5.2. **Taking C to be a List Decodable Code.** For the unique decoding of asymptotically good error correcting codes mentioned in section 2.2, [9] take C to be a list decodable code, and give a unique decoding procedure for a linear code. The next lemma describes the decoding scheme and shows how the parameters behave.

Lemma 9. *Assuming:*

- *for every $\alpha > 0$, there exists $[N, rN, \frac{1}{2}N]_{q(\alpha)}$ code C, which can be list decoded from $(1-\alpha)$ fraction of errors.*
- *for every $\epsilon > 0$, there exists $(\epsilon L, \frac{1}{4})$ extractor $G : [L] \times [T] \to [N]$, with regular right degree $Q = \frac{LT}{N}$, and entropy loss Λ.*

Then for every $0 < \delta \leq \frac{1}{2}$, $\epsilon < \delta$, the code $G \circ C$ is $[L, \frac{r\epsilon}{\Lambda}L, (1-\epsilon)L]_{q(\frac{\delta}{4})^T}$ code, for which there is a list decoding procedure from a fraction of $(1-\delta)$ errors, implying a unique decoding procedure from $\frac{1-\epsilon}{2}$ fraction of errors.

The proof follows exactly the lines of [9]:

Proof. Let $\delta > 0$, $\epsilon < \delta$. Let C be the code from the first assumption using $\alpha = \frac{\delta}{4}$, and G the extractor from the second assumption. Exactly as in lemma 1, $G \circ C$ has the stated rate, relative distance, and alphabet size. We now show the decoding procedure for $G \circ C$. Let $w \in [q^T]^L$, be a word which agree with some codeword $G \circ C(x)$ on at least δL coordinates. Denote by $X \subseteq [L]$ the coordinates in agreement. $|X| \geq \delta L > \epsilon L$. We now perform the decoding procedure described in section 3.3, only, instead of taking all votes to the sets $S_1, \ldots, S_N$, we take only the t most popular votes, for t to be determined later. We now claim:

Claim 2. *If for some $i \in [N]$, $C(x)_i \notin S_i$, then in G there are at most $\frac{Q}{t+1}$ edges between X and i*

Proof. All edges from X to i vote for the same symbol, as X contain coordinates of a legitimate codeword. Thus, if this symbol didn't make it to the t most popular votes, it means that there are t other symbols, each having more than $\frac{Q}{t+1}$ votes. Assuming by contradiction that the claim is false imply that the degree of i is bigger than $t \cdot \frac{Q}{t+1} + \frac{Q}{t+1} = Q$. □

Thus, denoting by $Y \in [N]$ all coordinates for which $C(x)_i \notin S_i$, we conclude that

$$|\Gamma_G(X) \cap Y| < |Y|\frac{Q}{t+1}$$

Choosing t such that $t+1 = \ulcorner \frac{2}{\delta} \urcorner$ we have:

$$\frac{|\Gamma_G(X) \cap Y|}{|Y|Q} < \frac{\delta}{2} \tag{29}$$

The reverse mixing lemma (Lemma 3), implies that:

$$\frac{|\Gamma_G(X) \cap Y|}{|Y|Q} > \delta(1 - \frac{1}{4}\frac{N}{|Y|})$$

And so for every Y such that $|Y| \geq \frac{N}{2}$:

$$\frac{|\Gamma_G(X) \cap Y|}{|Y|Q} > \frac{\delta}{2} \tag{30}$$

Thus, it must be that $|Y| < \frac{N}{2}$, otherwise (30) contradicts (29). We conclude that there are more than $\frac{1}{2}N$ sets S_i, which contain $C(x)_i$. We now use the sets $S_1, \dots, S_N$ to construct t strings $w_1, \dots, w_t$. For each $1 \leq j \leq t$, and $1 \leq i \leq N$ define $(w_j)_i$ be the j^{th} symbol of S_i.

Claim 3. *At least one of the words $w_1, \dots, w_t$ has αN agreement with $C(x)$.*

Proof. More than half of the sets S_i contain the symbol $C(x)_i$. Averaging over the t words w_j, there is at least one such word with at least $\frac{N}{2t}$ coordinates from $C(x)$. By the choice of t, $\frac{N}{2t} \geq \frac{\delta}{4}N = \alpha N$. □

Thus, using the decoding procedure of C on $w_1, \dots, w_t$, gives a list which contains x. Now, for the unique decoding, let $w \in [q^T]^L$ have at most $\frac{1-\epsilon}{2} < (1-\delta)$ fraction of errors. We apply the decoding procedure above, to get a decoding list L. Encoding all words in L we find the single word within distance at most $\frac{1-\epsilon}{2}$ from w. □

Remark 1.

(1) *One could argue that we don't need the t popular votes, and we can actually do with all Q votes. However, this might have a time cost, especially if $\epsilon << \delta$. To see why, recall that $t \approx \frac{1}{\delta} << \frac{1}{\epsilon}$. $Q = \frac{L \cdot T}{N} = \frac{\Lambda}{\epsilon}$. Even if we take an extractor with optimal entropy loss and constant error, we still have $Q = \theta(\frac{1}{\epsilon}) >> \frac{1}{\delta}$.*

(2) *One could worry what happens if the t we choose is larger than Q. This cannot happen as $Q = \frac{LT}{N} = \frac{\Lambda}{\epsilon}$, where $\epsilon < \delta$, and in all cases $\Lambda > 2$, meaning $Q > \frac{2}{\delta} \approx t$.*

5.3. **Taking C to be a List Recoverable Code.** For the optimal rate list decodable code construction and the optimal decoding list size list decodable code construction mentioned in 2.3, [9] take C to be a list recoverable code. The next lemma gives the parameters and decoding scheme for composing a list recoverable code with a disperser.

Lemma 10. *Assuming that for every $\epsilon > 0$:*

- *There exists $(\epsilon L, \frac{1}{2})$ disperser $G : [L] \times [T] \to [N]$ with entropy loss Λ, and regular right degree $Q = \frac{LT}{N} = \frac{\Lambda}{\epsilon}$*
- *There exists $(N, rN)_{q=O((\frac{1}{\epsilon})^2)}$ code C which is $(\frac{1}{2}, \frac{\Lambda}{\epsilon}, M)$-list recoverable code*

Then for every $\epsilon < 0$, $G \circ C$ is a $(L, \frac{r\epsilon}{\Lambda} L)_{O((\frac{1}{\epsilon})^2)^T}$, $(1-\epsilon, M)$-list decodable code

We follow the lines of [8] "Reduction of list decoding to list recoverability using expanders".

Proof. Let $\epsilon > 0$. Let C, and G be as above. The block length, rate and alphabet size of $G \circ C$ follow exactly as in lemma 1. We now show the list decodability parameters. Let $w \in [q^T]^L$, be a word which agree with some codeword $G \circ C(x)$ on at least ϵL coordinates. Denote by $X \subseteq [L]$ the coordinates in agreement. $|X| \geq \epsilon L$. We now perform the decoding procedure described in section 3.3, yielding $S_1, \ldots, S_N$ of size $\frac{\Lambda}{\epsilon}$ each. By the expansion property of G, there are at least $\frac{1}{2}N$ sets S_i, for which $C(x)_i \in S_i$. Thus, performing the decoding procedure of C we get the decoding list of size M. □

5.4. **Taking C to be a List Recoverable Code from Arbitrary Size.** For the almost optimal rate list decodable code construction mentioned in 2.3, [7] takes C to be a list recoverable code from arbitrary size[5] to construct an almost optimal rate list decodable code. The next lemma analyzes the parameters and decoding scheme.

Lemma 11. *Let $C \subset [M]^D$ be a code of rate r_C, which is (L, ζ_C) list recoverable code from arbitrary size. Let $G : [N] \times [T] \to [D]$ be a $(\epsilon N, \zeta_G)$-disperser, with entropy loss $\Lambda_G = \frac{\epsilon NT}{D}$. if $M \cdot D \geq \frac{N \cdot T}{1-\zeta_C-\zeta_G}$, then $G \circ C$ has the following properties:*

1. *It has rate $r_C \cdot \frac{\epsilon}{\Lambda_G}$, and is defined over an alphabet of size M^T.*
2. *It is a $(1-\epsilon, L)$-list decodable code.*

Proof. Let C and G be as above. The rate and alphabet size follow immediately as in lemma 1. We now show the list decodability parameters. Let $w \in [M^T]^N$, be a word which agree with some codeword $G \circ C(x)$ on at least ϵN coordinates. Denote by $X \subseteq [N]$ the coordinates in agreement. $|X| \geq \epsilon N$. We now perform the decoding procedure described in section 3.3, yielding $S_1, \ldots, S_D$. We think of each element $s \in S_i$, as an ordered pair $(s, i) \in [M] \times [D]$. Thus, $S = \bigcup_i S_i$ can be thought of a subset of $[M] \times [D]$. Since X is the set

[5]In the terminology of [7] C is a strong extractor.

of coordinates in agreement, then for all the neighbors $G(x,j) \in [D]$ ($j \in [T]$) of $x \in X$, we have:

$$(w_x)_j = C(x)_{G(x,j)}$$

More specifically we have:

$$((w_x)_j, G(x,j)) = (C(x)_{G(x,j)}, G(x,j)) \tag{31}$$

By the expansion property of G, there are at least $(1-\zeta_G)D$ indices $G(x,j) \in [D]$ for which (31) holds. Thus, denoting $A_S(x) = \{i|(C(x)_i, i) \in S\}$, we have that $|A_S(x)| \geq (1-\zeta_G)D$. Now, $|S| \leq NT$, and by the assumption $M \cdot D \geq \frac{N \cdot T}{1-\zeta_C-\zeta_G}$, and so:

$$(\frac{|S|}{MD} + \zeta_C)D \leq (\frac{NT}{MD} + \zeta_C)D \leq (1-\zeta_G)D \leq A_S(x).$$

Thus, by the list recoverability from arbitrary size property of C, there are at most L codewords having $|A_s(x)|$ agreement with $C(x)$, or in other words at most L codewords having ϵN agreement with $G \circ C(x)$. □

6. Asymptotically Good Error Correcting Codes Over Large Alphabets

Section 1.1 shows that when taking the [1] construction of $C_{Jus} \circ G$, where C_{Jus} is a Justesen code and G is a balanced disperser, the alphabet size, as well as the rate can be improved by replacing the balanced expander with an unbalanced one, while keeping all other parameters the same[6]. In this section we formally prove this (Theorems 1-6).

6.1. Improving the Alphabet Size (Theorem 1).

The proof is straight forward from lemma 1:

Proof. Let $\delta < 1$. Take C to be a $[N, r_{Jus}N, \delta_{Jus}N]_{q_{Jus}}$ Justesen code having constant rate, constant relative distance and constant alphabet size. Take $G : [L] \times [T] \to [N]$ to be a $((1-\delta)L, \delta_{Jus})$-disperser. By the above lemma the resulting code $G \circ C$ is a $[L, \frac{r_{Jus} \cdot 1-\delta}{\Lambda} L, \delta L]_{q_{Jus}^T}$ code. Plugging in the degree and entropy loss of the dispersers $G_{balanced}$, G_{opt}, $G_{D_{opt}}$ and $G_{explicit}$ gives the claimed rate and alphabet size. □

[6]This is true when using G_{opt} or $G_{D_{opt}}$. For the explicit extractor $G_{explicit}$ the alphabet size is improved, but the rate is inferior to that of [1], due to its larger entropy loss.

6.2. **Approaching the Singleton Bound (Theorems 2, 3).** We first see how the rate function of $G \circ C_{Jus}$ behaves for prescribed alphabet size q and relative distance $\delta < 1$. The analysis below follows that of [1], only we represent the rate function using the entropy loss of the disperser G as implied by (28):

$$Rate_{G \circ C_{Jus}}(\delta, q) = R_{Jus}(\delta_0, q_0) \cdot \frac{(1-\delta)}{\Lambda} \tag{32}$$

Where, $q, \delta < 1$ are the prescribed alphabet size and relative distance of the construction and δ_0, $q_0 = q^{\frac{1}{T}}$ are the relative distance and alphabet size of C_{Jus}. Writing the rate function as above it is immediate to see the improvement in the rate when using an unbalanced disperser with optimal entropy loss:

(1) The smaller the entropy loss is, the larger the rate is.
(2) The alphabet size of the Justesen code is $q^{\frac{1}{T}}$, where T is the degree of the disperser used. Since unbalanced dispersers have smaller degree the alphabet size of the Justesen code can be larger. The rate function of Justesen code (9) is increasing in the alphabet size and so we get a better rate for the Justesen code, and thus a better rate for the overall code.

We now turn to the analysis. Substituting the Justesen code lower bound (11) in (32) we have that:

$$Rate_{G \circ C_{Jus}}(\delta, q) \geq E_0(1-\delta) - \frac{E_1 \cdot T \cdot (1-\delta)}{2\log_2 q - 2T} \tag{33}$$

where:

$$E_0 = \frac{1 - 2\delta 0}{2\Lambda} \tag{34}$$

$$E_1 = \frac{2\delta_0}{\Lambda} \tag{35}$$

We now split the analysis for the balanced and unbalanced cases:

Claim 4. *For the balanced disperser $G_{balanced}$, the following holds:*

(1) *E_0 is of the form $\frac{f(\delta_0)}{\mu(\delta)}$, where $\mu(\delta) > 1$ and $\lim_{\delta \to 1} \mu(\delta) = 1$.*
(2) *$E_1 \cdot T \cdot (1-\delta)$ is a constant which depends only on δ_0.*

Thus, for the balanced case (33) can be rewritten as:

$$Rate_{G \circ C_{Jus}}(\delta, q) \geq \frac{f(\delta_0)}{\mu(\delta)}(1-\delta) - \frac{g(\delta_0)}{2\log_2 q - 2T}$$

Let $\gamma_1 > g(\delta_0)$. Let $\gamma_0 < f(\delta_0)$. By the property of $\mu(\delta)$ above there exists $\delta_{min}(\gamma_0)$ such that for any $\delta > \delta_{min}$, $\gamma_0 > \frac{f(\delta_0)}{\mu(\delta)}$. Since T is increasing in δ, for every $\delta > \delta_{min}$, there exists q_{min} such that for every $q > q_{min}$ $\log_2 q > 2T$. Altogether, there exists positive constants γ_0, γ_1, such that for every $\delta > \delta_{min}(\gamma_0)$, there exists $q_{min}(\delta)$, such that for every $q > q_{min}$ the rate function of the construction satisfies (6) proving theorem 2.

Claim 5. *For G_{opt} and $G_{D_{opt}}$ the following holds:*

(1) *Λ depends only on δ_0 and $\lim_{\delta\to 1} T \cdot (1-\delta) = 0$.*
(2) *E_0 depends only on δ_0.*

Thus, (33) can be rewritten as:

$$Rate_{G\circ C_{Jus}}(\delta, q) \geq f'(\delta_0)(1-\delta) - \frac{g'(\delta)}{2\log_2 q - 2T}$$

where $\lim_{\delta\to 1} g'(\delta) = 0$. Let $\gamma_0 = f'(\delta_0)$. Let $\gamma_1 > 0$. Since $\lim_{\delta\to 1} g'(\delta) = 0$ there is $\delta_{min}(\gamma_1)$, such that for every $\delta > \delta_{min}$, $\gamma_1 > g'(\delta)$. Again, since T is increasing in δ, for every $\delta > \delta_{min}$, there exists q_{min} such that for every $q > q_{min}$ $\log_2 q > 2T$. Altogether, there exists a positive constant γ_0, such that for every $\gamma_1 > 0$, there is $\delta_{min}(\gamma_1)$ such that for every $\delta > \delta_{min}$, there exists $q_{min}(\delta)$, such that for every $q > q_{min}$ the rate function of the construction satisfies (6). This proves theorem 3.

We now turn to prove claims 4, 5 and estimate the exact values γ_0 can attain in each case. We note that by lemma 1, the disperser we need is a $((1-\delta)L, \delta_0)$-disperser $G : [L] \times [T] \to [N]$, where N, δ_0 are the block length and relative distance of the Justesen code used in $G \circ C_{Jus}$, and δ is the relative distance of $G \circ C_{Jus}$. We refer the reader to table 2 for the disperser graphs parameters T and Λ used in the proofs below.

Proof. (Claim 4) The degree T of $G_{balanced}$ satisfies:

$$\mu(\delta)\frac{4(\frac{1}{\delta_0}-1)}{(1-\delta)} \geq T \geq \frac{4(\frac{1}{\delta_0}-1)}{(1-\delta)} \tag{36}$$

where $\lim_{\delta\to 1}\mu(\delta) = 1$, $\mu > 1$ (see [1] section 3). The entropy loss of $G_{balanced}$ satisfies:

$$\Lambda = (1-\delta)T$$

Substituting the above in (34) we get:

$$E_0 = \frac{(1-2\delta_0)}{8\mu(\frac{1}{\delta_0}-1)}$$

Obviously, E_0 is of the form $\frac{f(\delta_0)}{\mu(\delta)}$ as claimed above. Substituting the degree and entropy loss above in (35) we have:

$$E_1 \cdot T \cdot (1-\delta) \geq 2\delta_0.$$

Thus, we can take $\gamma_0 < \frac{(1-2\delta_0)}{8(\frac{1}{\delta_0}-1)}$, and $\gamma_1 > 2\delta_0$. The maximum value of γ_0 is attained at $\delta_0 \approx 0.29$, and is ≈ 0.021. □

Proof. (Claim 5) For G_{opt} substituting its entropy loss in (34) we get:

$$E_0 = \frac{(1-2\delta_0)}{2(\ln(\frac{1}{\delta_0})+1)}$$

Obviously, E_0 depends only on δ_0. Also Λ is dependent only on δ_0, and by the degree of G_{opt} we have $\lim_{\delta\to 1} T \cdot (1-\delta) = 0$ as claimed above. By E_0 above, we can take $\gamma_0 = \frac{(1-2\delta_0)}{2(\ln(\frac{1}{\delta_0})+1)}$. γ_0 attains its maximum value of 0.0605 at $\delta_0 \approx 0.1$.

For $G_{D_{opt}}$ substituting its entropy loss in (34) we get:

$$E_0 = \frac{(1-2\delta_0)}{2(\ln(\frac{4}{\delta_0})+1)}$$

Again, E_0 and Λ depend only on δ_0. Also $\lim_{\delta\to 1} T \cdot (1-\delta) = 0$ as claimed above. By E_0 above, we can take $\gamma_0 = \frac{(1-2\delta_0)}{2(\ln(\frac{4}{\delta_0})+1)}$. γ_0 attains its maximum value of 0.0427, at $\delta_0 \approx 0.0855$. □

6.3. **Beating The Zyablov Bound (Theorem 4).** [1] show that for large enough alphabet size q, the rate function of $G \circ C_{Jus}$ beats the Zyablov bound. We show that the alphabet size needed to beat this bound can be much smaller when using an unbalanced disperser G. For the analysis we need the following lemma implicit in [1].

Lemma 12. *Let $R_{Zyablov}(\delta)$ be the Zyablov rate function given in (4). Let $G \circ C_{Jus}$ be a code with rate function of the form (33). If q is large enough such that:*

$$\log_2 q > \frac{E_1 T}{2(E_0 - 1)} + T \tag{37}$$

where E_0, E_1 are as in (34),(35), then $Rate_{G\circ C_{Jus}}(\delta, q) > R_{Zyablov}(\delta, q)$

Proof. By the Gilbert-Varshmov bound $R_{GV}(\delta) \geq (1 - H_q(\delta))$. By the Singleton bound for every q, $R(\delta) < (1-\delta)$. Thus, we have:

$$(1-\delta) > (1 - H_q(\delta)) \tag{38}$$

Substituting (38) in (4), we get:

$$R_{Zyablov}(\delta) < \max_{\mu \geq 0}(1-\mu)(1-\frac{\delta}{\mu})$$

The maximum is achieved when $\mu = \sqrt{\delta}$, giving:

$$R_{Zyablov}(\delta) < (1-\sqrt{\delta})^2$$

Using (33) we need q satisfying:

$$E_0(1-\delta) - \frac{E_1 \cdot T \cdot (1-\delta)}{2\log_2 q - 2T} > (1-\sqrt{\delta})^2$$

Rearranging the above, we get:

$$\log_2 q > \frac{E_1 T}{2(E_0 - \frac{1-\sqrt{\delta}}{1+\sqrt{\delta}})} + T$$

Noting that $\frac{1-\sqrt{\delta}}{1+\sqrt{\delta}} < 1$, the lemma follows. □

Corollary 2. *For $G_{balanced}$, G_{opt}, $G_{D_{opt}}$, and $G_{explicit}$ in order to beat the Zyablov bound it is enough to take q, satisfying*

$$\log_2 q = \Theta(T) \tag{39}$$

where T is the disperser's degree.

Proof. Looking at (34) and (35), and using the fact that the entropy loss of $G_{balanced}$, G_{opt}, $G_{D_{opt}}$ and $G_{explicit}$ is dependent only on δ_0, (37) implies that we need q satisfying $\log_2 q > f(\delta_0) \cdot T$, where f is some function dependent only on δ_0. Since δ_0 is a constant the corollary follows. □

Plugging the degree of $G_{balanced}$, G_{opt}, $G_{D_{opt}}$, and $G_{explicit}$ in (39) Theorem 4 follows.

6.4. **Concatenation With Codes of Fixed Alphabet Size (Theorem 6).** We outline the proof of the theorem:

Proof. Let q be the prescribed fixed alphabet size, and $\delta < 1$ the required relative minimum distance. Following the lines of [1] section 4, we concatenate a construction of the form $G \circ C$ with a Wozencraft ensemble of codes. The code C is a $[N_C, r_C N_C, \delta_C N_C]_{q_C}$ code, which is a concatenation of two Reed-Solomon codes, having a rate function:

$$r_C(\delta_C) \;\; \geq \;\; (1-\sqrt{\delta_C})^2 \tag{40}$$

This code was used as the outer code in [15]. Denoting by μ the relative distance of the Wozencraft ensemble used as the inner code, the relative distance of $G \circ C$ should be $(\frac{\delta}{\mu})$ such that the final relative distance is δ. Lemma 1 implies that we need a $(\frac{\delta}{\mu}L, \delta_C)$-disperser $G : [L] \times [T] \rightarrow [N_C]$. Now, concatenating $G \circ C$ with a Wozencraft ensemble of codes having relative minimum distance μ, (28) implies that the overall rate satisfies:

$$Rate(\delta) \geq r_C(\delta_C)\frac{1-\frac{\delta}{\mu}}{\Lambda}(1-H_q(\mu)) = (1-\sqrt{\delta_C})^2\frac{1-\frac{\delta}{\mu}}{\Lambda}(1-H_q(\mu)) \tag{41}$$

where Λ is the entropy loss of G, and $(1 - H_q(\mu))$ is the lower bound on rate of the Wozencraft ensemble. We remark that although the code length of C cannot take arbitrarily large values (q is fixed), it is large enough to perform the concatenation with all the ensemble (see [1] section 4, [15]). Letting:

$$\gamma_0 = \frac{(1 - \sqrt{\delta_0})^2}{\Lambda} \tag{42}$$

and taking the maximum over μ, the rate function above has the form of (7). Plugging the entropy loss of G_{opt}, $G_{D_{opt}}$ (18), (23) in (42) we get the γ_0 values appearing in theorem 6. $\square$

7. Appendix

We now give the proof of theorem 7 which deals with error correcting codes with efficient decoding, and state the exact theorems and proofs dealing with list decodable codes summarized in table 1.

7.1. **Theorem 7, Asymptotically Good Error Correcting Codes with Explicit Decoding Procedure.**

Theorem 7. *For any $\beta > 0$,$\frac{1}{2} \geq \delta > 0$ there is a constant $B > 1$ such that for all $\frac{\delta}{4} > \epsilon > 0$ there is an explicitly specified code family with rate $(\frac{\epsilon}{B})$, relative distance at least $(1 - \epsilon)$ and alphabet size $f(\epsilon)$. A code of block length N in the family can be list decoded in time $d(\epsilon, N)$ from up to a $(1 - \delta)$ errors, and can be encoded in $e(\epsilon, N)$ time. where:*

- *$f(\epsilon)$ is given by:*

Extractor Used	$f(\epsilon)$	*Ref*
Balanced Ramanujan Graph	$2^{O(\frac{1}{\epsilon})}$	[9] *Theorem 8*
$G_{E_{opt}}$	$2^{O(\log^2(\frac{1}{\epsilon}))}$	*Section 7.1*
$G_{explicit}$	$\widetilde{plog}(\frac{1}{\epsilon})$	*Section 7.1*

- *For the balanced graph,*([9] *Theorem 8*)*:*
  $e_{balanced}(\epsilon, N) = O(N \log^{O(1)} N)$
  $d_{balanced}(\epsilon, N) = O(N^{1+\beta})$
- *For $G_{E_{opt}}$:*
  $e(\epsilon, N) = e_{balanced}(\epsilon, O(\epsilon \log^2(\frac{1}{\epsilon}))N) + O(\log^2(\frac{1}{\epsilon})N \log^2 N)$
  $d(\epsilon, N) = d_{balanced}(\epsilon, O(\epsilon \log^2(\frac{1}{\epsilon}))N) + O(\log^2(\frac{1}{\epsilon})N \log^2 N)$
  there is an overhead of $2^{(\frac{1}{\epsilon})^{O(1)}} \cdot poly \log N$ time to construct $G_{E_{opt}}$.
- *For $G_{explicit}$:*
  $e(\epsilon, N) = e_{balanced}(\epsilon, O(\epsilon 2^{polyloglog(\frac{1}{\epsilon})} N) + O(2^{polyloglog(\frac{1}{\epsilon})} N \log^2 N)$

$$d(\epsilon, N) = d_{balanced}(\epsilon, O(\epsilon 2^{polyloglog(\frac{1}{\epsilon})}N) + O(2^{polyloglog(\frac{1}{\epsilon})}N\log^2 N)$$

The encoding and decoding times are as in the original construction except:

(1) Replacing N with $\frac{\epsilon NT}{\Lambda}$, where Λ, T are the entropy loss and degree of the extractor used. This is because we use an unbalanced extractor, implying that the block length of the code $G \circ C$, is longer than block length of the code C used.
(2) Adding $NT \cdot t_G$ time, where t_G is the time it takes to find a neighbor in G.

Recall that in section 5.2 we gave a general lemma stating the parameters and decoding scheme for a construction of the form $G \circ C$, where C is a list decodable code and G is an extractor. For theorem 7, [9] use the list decodable code from lemma 13 below, with a balanced Ramajuan graph having a large degree to assure the reverse mixing property.

Lemma 13. *([8] Lemma 11.1) For every $\alpha > 0$ there exists a prime power $q = q_\alpha$ of order $O(\frac{1}{\alpha^2})$, which may be assumed to be a power of 2, such that for all $\beta > 0$, the following holds. There is an explicitly specified code family with constant rate $r_{\alpha,\beta} > 0$ and relative distance at least $\frac{1}{2}$ over an alphabet of size q with the property that a code of block length N in the family can be list decoded from up to $(1-\alpha)$ fraction of errors in $O(N^{1+\beta})$ time, and can be encoded in $O(N \log^{O(1)} N)$*

We now prove the theorem:

Proof. Let $\delta = \frac{1}{4}$, $\beta > 0$, $\frac{1}{4} > \epsilon > 0$, and $\alpha = \frac{\delta}{4} = \frac{1}{16}$. Let C, be the $(N, r_{\alpha,\beta}N, \frac{1}{2}N)_{q(\frac{1}{16})}$ code from lemma 13. Taking a $(\epsilon L, \frac{1}{4})$-extractor $G : [L] \times [T] \to [N]$ lemma 9 implies that the code $G_{E_{opt}} \circ C$ is $(L, \frac{r\epsilon}{\Lambda}L, (1-\epsilon)L)_{O(1)^T}$, and is list decodable from $\frac{3}{4}L$ errors. We now split the analysis to three: using balanced expanding graph with the required mixing property, $G_{E_{opt}}$ and $G_{explicit}$.

For the balanced graph we have:

$$\begin{aligned} \Lambda &= O(1) \\ T &= O(\frac{1}{\epsilon}) \end{aligned}$$

By (25), (24), the degree and entropy loss of $G_{E_{opt}}$ we have:

$$\begin{aligned} \Lambda &= O(1) \\ T &= 2^{O(log^2(\frac{1}{\epsilon}))} \end{aligned}$$

and by (26), (27), the degree and entropy loss of $G_{explicit}$ we have:

$$\begin{aligned}\Lambda &= O(1)\\ T &= 2^{O(log^3 log(\frac{1}{\epsilon}))}\end{aligned}$$

giving the stated rate and alphabet size. For the encoding and decoding times we note that:

(1) The block length of $G \circ C$ is L, whereas the block length of C is $N = \frac{\epsilon LT}{\Lambda_G}$. Thus, the times appearing in lemma 13 should be taken accordingly.
(2) To encode we first need to encode using C $(O(N \log^{O(1)} N))$, and then perform the amplification $(LT \cdot t_G)$, where t_G is the time for computing a neighbor in G.
(3) To decode, we first need to perform the decoding scheme $(LT \cdot t_G)$, and then perform the decoding of C for $\frac{1}{\delta}$ strings $(O(N^{1+\beta}))$.
(4) For the balanced expanding graph, the encoding/decoding time overhead of $LT \cdot t_G$ needed for the composition is dominated by the encoding and decoding times of the code C. Thus, the times $e_{balanced}$ and $d_{balanced}$ are similar to the encoding and decoding times of C.

Substituting the degree, entropy loss and t_G for $G_{E_{opt}}$ and $G_{explicit}$ in the above, the theorem follows. We mention that in the case of $G_{E_{opt}}$ there is an additional overhead of $2^{(\frac{1}{\epsilon})^{O(1)}} \cdot poly \log N$ construction time as implied from lemma 7. □

We remark that had we known to explicitly construct an optimal extractor with degree $T = O(\log(\frac{1}{\epsilon}))$, the resulting code would have alphabet of size $(\frac{1}{\epsilon})^{O(1)}$.

7.2. **List Decodable Codes with Optimal Rate.**

Theorem 11. *For every $\epsilon > 0$, every constant $\gamma > 0$ there exists a code family with rate $\Omega(2^{-O(\gamma^2)}\epsilon)$, which can be list decoded from a fraction of $(1-\epsilon)$ errors, and have alphabet size $f(\epsilon)$. A code of block length N in the family can be found with high probability in time $cp(N, \epsilon, \gamma)$ or deterministically in time $cd(N, \epsilon, \gamma)$. Moreover, the code can be encoded in $e(N, \epsilon, \gamma)$ time, and list decoded in $d(N, \epsilon, \gamma)$. Where,*

- *$f(\epsilon)$ is given by:*

Disperser used	$f(\epsilon)$	*Ref*
$G_{balanced}$	$2^{O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})}$	[9] *Theorem 6*
G_{opt}	$2^{O(\log^2(\frac{1}{\epsilon}))}$	*This paper*
$G_{D_{opt}}$	$2^{O(\log^3(\frac{1}{\epsilon}))}$	*This paper*
$G_{explicit}$	$\widetilde{plog}(\frac{1}{\epsilon})$	*This paper*

- $cp_{balanced}(N,\epsilon,\gamma) = O(N^{2(1-\gamma)}\log(\frac{1}{\epsilon})$,
  $cd_{balanced}(N,\epsilon,\gamma) = 2^{O(N^{(1-\gamma)}(\frac{1}{\epsilon})\log(\frac{1}{\epsilon}))}$,
  $e_{balanced}(N,\epsilon,\gamma) = O(N^{2(1-\gamma)}\log^2 N \log^{O(1)}(\frac{1}{\epsilon}))$,
  $d_{balanced}(N,\epsilon,\gamma) = 2^{O(N^{\gamma}\log(\frac{1}{\epsilon}))}$
  are the construction, encoding and decoding times achieved in [9].
- $cp_{D_{opt}}(N,\epsilon,\gamma) = cp_{balanced}(\epsilon\log^2(\frac{1}{\epsilon})\cdot N,\epsilon,\gamma) + 2^{(\frac{1}{\epsilon})^{O(1)}} plog(N)$,
  $cd_{D_{opt}}(N,\epsilon,\gamma) = cd_{balanced}(\epsilon\log^2(\frac{1}{\epsilon})\cdot N,\epsilon,\gamma) + 2^{(\frac{1}{\epsilon})^{O(1)}} plog(N)$,
  $e_{D_{opt}}(N,\epsilon,\gamma) = e_{balanced}(\epsilon\log^2(\frac{1}{\epsilon})\cdot N,\epsilon,\gamma) + O(\log^2(\frac{1}{\epsilon})N\log^2 N)$,
  $d_{D_{opt}}(N,\epsilon,\gamma) = d_{balanced}(\epsilon\log^2(\frac{1}{\epsilon})\cdot N,\epsilon,\gamma) + O(\log^2(\frac{1}{\epsilon})N\log^2 N)$
  are the construction, encoding and decoding times when using the disperser $G_{D_{opt}}$.
- $cp_{zig-zag-ext}(N,\epsilon,\gamma) = cp_{balanced}(\epsilon\log^3(\frac{1}{\epsilon})\cdot N,\epsilon,\gamma)$,
  $cd_{zig-zag-ext}(N,\epsilon,\gamma) = cd_{balanced}(\epsilon\log^3(\frac{1}{\epsilon})\cdot N,\epsilon,\gamma)$,
  $G_{E_{opt}}(N,\epsilon,\gamma) = e_{balanced}(\epsilon\log^3(\frac{1}{\epsilon})\cdot N,\epsilon,\gamma) + N2^{polyloglog(\frac{1}{\epsilon})}(\log^2 N + \log^3(\frac{1}{\epsilon}))$,
  $d_{zig-zag-ext}(N,\epsilon,\gamma) = d_{balanced}(\epsilon\log^3(\frac{1}{\epsilon})\cdot N,\epsilon,\gamma) + N2^{polyloglog(\frac{1}{\epsilon})}(\log^2 N + \log^3(\frac{1}{\epsilon}))$
  are the construction, encoding and decoding times when using the extractor $G_{explicit}$.

The encoding and decoding times are as in the original construction except:

(1) Replacing N with $\frac{\epsilon NT}{\Lambda}$, where Λ, T are the entropy loss and degree of the disperser used. Using an unbalanced disperser implies that the block length of the code $G \circ C$ is longer than block length of the code C used.
(2) Adding $NT \cdot t_G$ time, where t_G is the time it takes to find a neighbor in G. This is the time it takes to perform the composition/decoding scheme.

Recall that in section 5.3 we gave a general lemma stating the parameters and decoding scheme for a construction of the form $G \circ C$, where C is a list recoverable code and G is a disperser. For theorem 11, [9] use the list recoverable code from lemma 14 below, with a balanced Ramajuan graph.

Lemma 14. *(implicit in [8] Theorem 9.16) For every $0 < \gamma \leq \frac{1}{2}$ and every $\epsilon > 0$, there exist a code family with the following properties:*

(1) *The family has rate $2^{-O(\frac{1}{\gamma^2})}$ and is defined over an alphabet of size $O(\frac{1}{\epsilon^2})$.*
(2) *Any code of block length N in the family is $(\frac{1}{2}, O(\frac{1}{\epsilon}), 2^{O(N^\gamma \log(\frac{1}{\epsilon}))})$-list recoverable. Such list recovering can be accomplished in $2^{O(N^\gamma \log(\frac{1}{\epsilon}))}$ time.*
(3) *A code of block length N in the family can be constructed in deterministic $2^{O(N^{1-\gamma}\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))}$ time, or probabilistically in $O(N^{2(1-\gamma)}\log(\frac{1}{\epsilon}))$ time. Also, encoding can be performed in $O(N^{2(1-\gamma)}\log^2 N \log^{O(1)}(\frac{1}{\epsilon}))$ time .*

Remark 2. *As implied by lemma 10 from section 5.3, the size of the voting sets is $(\frac{\Lambda}{\epsilon})$. The list recoverable code above can deal with sets of size $O(\frac{1}{\epsilon})$. The constant in the $O(\cdot)$ can be adjusted by picking appropriate γ. This will only affect the constants in the rate and the decoding list size of the code C. The exact details can be found in* [8] *Lemma 9.15 and Theorem 9.16.*

Proof. The proof of the theorem follows immediately from plugging in lemma 10 the code from lemma 14, together with G_{opt}, $G_{D_{opt}}$, and $G_{explicit}$ which are taken to be $(\epsilon L, \frac{1}{2})$-dispersers $G : [L] \times [T] \to [N]$. For completeness we recall the following:

(1) The above G_{opt}, has $O(1)$ entropy loss and degree $O(\log(\frac{1}{\epsilon}))$, giving the required rate and alphabet size.
(2) The above $G_{D_{opt}}$, has $O(1)$ entropy loss and degree $O(\log^2(\frac{1}{\epsilon}))$, giving the required rate and alphabet size. With these degree and entropy loss we have that $N = \frac{\epsilon LT}{\Lambda} = O(\epsilon \log^2(\frac{1}{\epsilon})L)$, and it takes $O(\log^2 L)$ to compute a neighbor in $G_{D_{opt}}$. The construction time of $G_{D_{opt}}$ is $2^{(\frac{1}{\epsilon})} polylog L$.
(3) The above $G_{explicit}$, has $O(1)$ entropy loss and degree $O(\log^3(\frac{1}{\epsilon}))$, giving the required rate and alphabet size. With these degree and entropy loss we have that $N = O(\epsilon \log^3(\frac{1}{\epsilon})L)$, and it takes $O(\log^2 L + \log^3(\frac{1}{\epsilon}))$ to compute a neighbor in $G_{explicit}$. The construction time of $G_{explicit}$ is $poly(\frac{1}{\epsilon})$.

□

7.3. **List Decodable Codes with Optimal List Size.**

Theorem 12. *For every $\epsilon > 0$, there exists a code family with rate $\Omega(\epsilon^2)$, which can be list decoded from a fraction of $(1-\epsilon)$ errors, and have alphabet size $f(\epsilon)$. A code of block length N in the family can be found with high probability in time $cp(N,\epsilon)$ or deterministically in time $cd(N,\epsilon)$. Moreover the code can be encoded in $e(N,\epsilon)$ time, and list decoded in $d(N,\epsilon)$. Where,*

- *$f(\epsilon)$ is exactly as in Theorem 11.*
- *$cp_{balanced}(N,\epsilon) = O((\frac{1}{\epsilon})\log(\frac{1}{\epsilon})\log^2 N)$,*
  $cd_{balanced}(N,\epsilon) = N^{O((\frac{1}{\epsilon})\log(\frac{1}{\epsilon}))}$,
  $e_{balanced}(N,\epsilon) = O(N\log N)$,
  $d_{balanced}(N,\epsilon) = O((\frac{1}{\epsilon})^{O(1)}N^2\log N)$
  are the construction, encoding and decoding times achieved in [9].
- *The construction, encoding and decoding times of $G_{D_{opt}}$, $G_{explicit}$ are computed from the times above in the exact manner described in 7.2.*

The construction used to obtain these codes is exactly as the one used in 7.2, the only change is the list recoverable code used:

Lemma 15. *(Implicit in [8] Theorem 9.14) For every $\epsilon > 0$ there exists a code family with the following properties:*

1. *It has rate $\Theta(\epsilon)$, and is defined over an alphabet of size $q = O(\frac{1}{\epsilon^2})$.*
2. *each code in the family is $(\frac{1}{2}, \frac{1}{\epsilon}, O(\frac{1}{\epsilon}))$-list recoverable.Such list recovering can be accomplished in $O((\frac{1}{\epsilon})^{O(1)}N^2\log N)$ time.*
3. *A code if block length N in the family can be constructed in deterministic $N^{O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))}$ time, or probabilistically in $O((\frac{1}{\epsilon})\log(\frac{1}{\epsilon})\log^2 N)$ time. Also, encoding can be performed in $O(N\log N)$ time.*

More details regarding the above code construction can be found in [8] section 9.3 on Pseudolinear codes.

Theorem 12 now follows by plugging the above code with G_{opt}, $G_{D_{opt}}$, and $G_{explicit}$ in lemma 10.

7.4. **Almost Optimal Rate List Decodable Codes.**

Theorem 13. *For every $\epsilon > 0$*

1. *There exists a family of codes constructible in time $t(N,\epsilon)$ having rate $\Omega(\frac{\epsilon}{log^{O(1)}(\frac{1}{\epsilon})})$, which can be list decoded from a fraction of $(1-\epsilon)$ errors, and have alphabet size $f(\epsilon)$. A code of block length N in the family has a decoding list size of $2^{\sqrt{g(\epsilon)\cdot N\log(g(\epsilon)\cdot N)}}$,*

where $f(\epsilon)$ is as in Theorem 11 and $t(\epsilon, N)$, $g(\epsilon, N)$ are given by:

Disperser used	$g(\epsilon)$	$t(\epsilon, N)$	*Ref*
$G_{balanced}$	$\frac{1}{\log^{O(1)}(\frac{1}{\epsilon})}$	$poly(N, \frac{1}{\epsilon})$	[7]
$G_{D_{opt}}$	$\frac{\epsilon}{\log^{O(1)}(\frac{1}{\epsilon})}$	$2^{\frac{1}{\epsilon}} polylog(N)$	*This paper*
$G_{explicit}$	$\frac{\epsilon 2^{polyloglog(\frac{1}{\epsilon})}}{\log^{O(1)}(\frac{1}{\epsilon})}$	$poly(N, \frac{1}{\epsilon})$	*This paper*

(2) *There exists a family of codes having rate $\Omega(\frac{\epsilon}{\log(\frac{1}{\epsilon})})$. Each code in the family is a $(1-\epsilon, O(\frac{1}{\epsilon}))$-list decodable code.*

In section 5.4 we gave a general lemma giving the parameters and decoding scheme for a construction of the form $G \circ C$, where C is a list recoverable code from arbitrary size and G is a disperser. For theorem 13, [7] use the following list recoverable code from arbitrary size, which is based on strong extractors from Reed-Muller codes from [16] Theorem 1.

Lemma 16. *For every $\epsilon > 0$ and every $\beta \geq 2$ there is an explicit family of codes having rate $\Omega(\frac{1}{\log^{O(1)}\beta \cdot \log^{O(1)}(\frac{1}{\epsilon})})$ over an alphabet of size $\beta\frac{1}{\epsilon}$. A code of block length D is the family is a $(L, \frac{1}{4})$-list recoverable from arbitrary size, where $L = 2^{O(\sqrt{D \cdot g'(\epsilon)\log(D \cdot g'(\epsilon))})}$ and $g'(\epsilon) = \Theta(\frac{1}{\log^{O(1)}\beta \cdot \log^{O(1)}(\frac{1}{\epsilon})})$*

Also, the existence of optimal strong extractors imply the following in terms of list recoverability from arbitrary size.

Lemma 17. *For every $\epsilon > 0$ and every $\beta \geq 2$, there exists a family of codes having rate $\Omega(\frac{1}{\log\beta + \log(\frac{1}{\epsilon})})$ over an alphabet of size $\beta\frac{1}{\epsilon}$. A code of block length D is the family is a $(O(\beta\frac{1}{\epsilon}), \frac{1}{4})$-list recoverable from arbitrary size.*

We now prove Theorem 13. Let $\epsilon > 0$. Let $G : [N] \times [T] \to [D]$ be a $(\epsilon N, \frac{1}{4})$-disperser with entropy loss Λ_G. We let $\beta = 2\Lambda_G$.

For the explicit part of the theorem, we pick the code C to be as in lemma 16, of block length D, with the above ϵ, and alphabet size $M = \beta \cdot (\frac{1}{\epsilon})$. By the choice of β, we have:

$$M \cdot D = \beta(\frac{1}{\epsilon})D = 2\frac{\Lambda_G}{\epsilon}D \geq 2NT$$

and so by lemma 11, $G \circ C$ has rate $r_C \cdot \frac{\epsilon}{\Lambda_G}$, alphabet size M^T, and is $(1-\epsilon, L)$. The degree T of the various dispersers give the alphabet size as given by $f(\epsilon)$ in the Theorem. Having G with a constant error of $\frac{1}{4}$, implies that for all of the dispersers used $\Lambda_G = O(1)$, thus $\beta = O(1)$, and for all dispersers we get the stated rate of $\Omega(\frac{\epsilon}{log^{O(1)}(\frac{1}{\epsilon})})$. By lemma 16 the decoding list size $L = 2^{O(\sqrt{D\cdot g'(\epsilon)\log(D\cdot g'(\epsilon))})}$ and $g'(\epsilon) = \Theta(\frac{1}{\log^{O(1)}(\frac{1}{\epsilon})})$. Having $O(1)$ entropy loss, we have that $D = \Theta(\epsilon NT)$, yielding

$$L = 2^{O(\sqrt{\epsilon NT\cdot g'(\epsilon)\log(\epsilon NT\cdot g'(\epsilon))})} \tag{43}$$

Substituting the degree T of $G_{balanced}$, $G_{D_{opt}}$ and $G_{explicit}$ in the above, gives L, and $g(\epsilon)$ as stated in the theorem.

Finally, the construction time of the various dispersers gives $t(N, \epsilon)$ as in the Theorem.

For the second part of the theorem, we note that the existence of optimal strong extractors, which implies lemma 17 plugged in lemma 11 together with G_{opt} yields the stated parameters.

References

1. N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth, *Construction of asymptotically good, low-rate error-correcting codes through pseudo-random graphs*, IEEE Transactions on Information Theory **38** (1992), 509–516.
2. N. Alon, J. H. Spencer, and P. Erdős, *The Probabilistic Method*, Wiley–Interscience Series, John Wiley & Sons, Inc., New York, 1992.
3. B. Chor and O. Goldreich, *Unbiased bits from sources of weak randomness and probabilistic communication complexity*, SIAM Journal on Computing **17** (1988), no. 2, 230–261.
4. P. Elias, *List decoding for noisy channels*, 1957-IRE WESCON Convention Record, Pt. 2, 1957, pp. 94–104.
5. O. Goldreich and A. Wigderson, *Tiny families of families with random properties: A quality-size trade-off for hashing*, Random Structures and Algorithms **11** (1997), 315–343.
6. V. Guruswami and P. Indyk, *Near-optimal linear-time codes for unique decoding and new list decodable codes over smaller alphabets*, Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002.
7. Venkatesan Guruswami, *Better extractors for better codes?*, Proceedings of the 36th Annual ACM Symposium on Theory of Computing, ACM Press, 2004, pp. 436–444.
8. ———, *List decoding of error-correcting codes*, Ph.D. thesis, Massachusetts Institute of Technology, August 2001.
9. Venkatesan Guruswami and Piotr Indyk, *Expander-based constructions of efficiently decodable codes*, Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 658–667.
10. N. Nisan and D. Zuckerman, *Randomness is linear in space*, Journal of Computer and System Sciences **52** (1996), no. 1, 43–52.

11. J. Radhakrishnan and A. Ta-Shma, *Bounds for dispersers, extractors, and depth-two superconcentrators*, SIAM Journal on Discrete Mathematics **13** (2000), no. 1, 2–24.
12. R. Raz, O. Reingold, and S. Vadhan, *Extracting all the randomness and reducing the error in Trevisan's extractors*, Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 149–158.
13. O. Reingold, S. Vadhan, and A. Wigderson, *Entropy waves, the zig-zag product, and new constant-degree expanders and extractors*, Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000.
14. M. Santha and U. V. Vazirani, *Generating quasi-random sequences from semi-random sources*, Journal of Computer and System Sciences **33** (1986), 75–87.
15. Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, *A modification of the constructive asymptotically good codes of Justesen for low rates*, Inform. Control **25** (1974).
16. A. Ta-Shma and D. Zuckerman, *Extractor codes*, Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 193–199.