# Unique Decoding of Explicit $\varepsilon$-balanced Codes Near the Gilbert–Varshamov Bound

Fernando Granha Jeronimo*, Dylan Quintana*, Shashank Srivastava†, and Madhur Tulsiani†
*University of Chicago, Chicago, USA
granha@uchicago.edu, dquintana@uchicago.edu
†Toyota Technological Institute, Chicago, USA
shashanks@ttic.edu, madhurt@ttic.edu

***Abstract*—The Gilbert–Varshamov bound (non-constructively) establishes the existence of binary codes of distance $1/2-\varepsilon$ and rate $\Omega(\varepsilon^2)$ (where an upper bound of $O(\varepsilon^2\log(1/\varepsilon))$ is known). Ta-Shma [STOC 2017] gave an explicit construction of $\varepsilon$-balanced binary codes, where any two distinct codewords are at a distance between $1/2-\varepsilon/2$ and $1/2+\varepsilon/2$, achieving a near optimal rate of $\Omega(\varepsilon^{2+\beta})$, where $\beta\to 0$ as $\varepsilon\to 0$.**

**We develop unique and list decoding algorithms for (a slight modification of) the family of codes constructed by Ta-Shma, in the adversarial error model. We prove the following results for $\varepsilon$-balanced codes with block length $N$ and rate $\Omega(\varepsilon^{2+\beta})$ in this family:**

**– For all $\varepsilon,\beta>0$, there are explicit codes which can be uniquely decoded up to an error of half the minimum distance in time $N^{O_{\varepsilon,\beta}(1)}$.**

**– For any fixed constant $\beta$ independent of $\varepsilon$, there is an explicit construction of codes which can be uniquely decoded up to an error of half the minimum distance in time $(\log(1/\varepsilon))^{O(1)}\cdot N^{O_\beta(1)}$.**

**– For any $\varepsilon>0$, there are explicit $\varepsilon$-balanced codes with rate $\Omega(\varepsilon^{2+\beta})$ which can be list decoded up to error $1/2-\varepsilon'$ in time $N^{O_{\varepsilon,\varepsilon',\beta}(1)}$, where $\varepsilon',\beta\to 0$ as $\varepsilon\to 0$.**

**The starting point of our algorithms is the framework for list decoding direct-sum codes develop in Alev et al. [SODA 2020], which uses the Sum-of-Squares SDP hierarchy. The rates obtained there were quasipolynomial in $\varepsilon$. Here, we show how to overcome the far from optimal rates of this framework obtaining *unique decoding* algorithms for explicit binary codes of near optimal rate. These codes are based on simple modifications of Ta-Shma's construction.**

## I. Introduction

Binary error correcting codes have pervasive applications [Gur10], [GRS19] and yet we are far from understanding some of their basic properties [Gur09]. For instance, until very recently no explicit binary code achieving distance $1/2-\varepsilon/2$ with rate near $\Omega(\varepsilon^2)$ was known, even though the existence of such codes was (non-constructively) established long ago [Gil52], [Var57] in what is now referred as the Gilbert–Varshamov (GV) bound. On the impossibility side, a rate upper bound of $O(\varepsilon^2\log(1/\varepsilon))$ is known for binary codes of distance $1/2-\varepsilon/2$ (e.g., [Del75], [MRRW77], [NS09]).

In a breakthrough result [TS17], Ta-Shma gave an explicit construction of binary codes achieving nearly optimal distance versus rate trade-off, namely, binary codes of distance $1/2-\varepsilon/2$ with rate $\Omega(\varepsilon^{2+\beta})$ where $\beta$ vanishes as $\varepsilon$ vanishes [1]. Actually, Ta-Shma obtained $\varepsilon$-balanced binary linear codes, that is, linear binary codes with the additional property that non-zero codewords have Hamming weight bounded not only below by $1/2-\varepsilon/2$ but also above by $1/2+\varepsilon/2$, and this is a fundamental property in the study of pseudorandomness [NN90], [AGHP92].

While the codes constructed by Ta-Shma are explicit, they were not known to admit efficient decoding algorithms, while such results are known for codes with smaller rates. In particular, an explicit binary code due to Guruswami and Rudra [GR06] is known to be even list decodable at an error radius $1/2-\varepsilon$ with rate $\Omega(\varepsilon^3)$. We consider the following question:

*Do explicit binary codes near the GV bound admit an efficient decoding algorithm?*

Here, we answer this question in the af-

[1] In fact, Ta-Shma obtained $\beta=\beta(\varepsilon)=\Theta(((\log\log 1/\varepsilon)/\log 1/\varepsilon)^{1/3})$ and thus $\lim_{\varepsilon\to 0}\beta(\varepsilon)=0$.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00048

firmative by providing an efficient [2] unique decoding algorithm for (essentially) Ta-Shma's code construction, which we refer as Ta-Shma codes. More precisely, by building on Ta-Shma's construction and using our unique decoding algorithm we have the following result.

**Theorem I.1** (Unique Decoding). *For every $\varepsilon > 0$ sufficiently small, there are explicit binary linear Ta-Shma codes $\mathcal{C}_{N,\varepsilon,\beta} \subseteq \mathbb{F}_2^N$ for infinitely many values $N \in \mathbb{N}$ with*

*(i) distance at least $1/2 - \varepsilon/2$ (actually $\varepsilon$-balanced),*

*(ii) rate $\Omega(\varepsilon^{2+\beta})$ where $\beta = O(1/(\log_2(1/\varepsilon))^{1/6})$, and*

*(iii) a unique decoding algorithm with running time $N^{O_{\varepsilon,\beta}(1)}$.*

*Furthermore, if instead we take $\beta > 0$ to be an arbitrary constant, the running time becomes $(\log(1/\varepsilon))^{O(1)} \cdot N^{O_\beta(1)}$ (fixed polynomial time).*

We can also perform "gentle" list decoding in the following sense (note that this partially implies Theorem I.1).

**Theorem I.2** (Gentle List Decoding). *For every $\varepsilon > 0$ sufficiently small, there are explicit binary linear Ta-Shma codes $\mathcal{C}_{N,\varepsilon,\beta} \subseteq \mathbb{F}_2^N$ for infinitely many values $N \in \mathbb{N}$ with*

*(i) distance at least $1/2 - \varepsilon/2$ (actually $\varepsilon$-balanced),*

*(ii) rate $\Omega(\varepsilon^{2+\beta})$ where $\beta = O(1/(\log_2(1/\varepsilon))^{1/6})$, and*

*(iii) a list decoding algorithm that decodes within radius $1/2 - 2^{-\Theta((\log_2(1/\varepsilon))^{1/6})}$ in time $N^{O_{\varepsilon,\beta}(1)}$.*

We observe that the exponent in the running time $N^{O_{\varepsilon,\beta}(1)}$ appearing in Theorem I.1 and Theorem I.2 depends on $\varepsilon$. This dependence is no worse than $O(\log\log(1/\varepsilon))$, and if $\beta > 0$ is taken to be an arbitrarily constant (independent of $\varepsilon$), the running time becomes $(\log(1/\varepsilon))^{O(1)} \cdot N^{O_\beta(1)}$. Avoiding this dependence in the exponent when $\beta = \beta(\varepsilon)$ is an interesting open problem. Furthermore, obtaining a list decoding radius of $1/2 - \varepsilon/2$ in Theorem I.2 with the same rate (or even $\Omega(\varepsilon^2)$) is another very interesting open problem and related to a central open question in the adversarial error regime [Gur09].

[2] By "efficient", we mean polynomial time. Given the fundamental nature of the problem of decoding nearly optimal binary codes, it is an interesting open problem to make these techniques viable in practice.

**Direct sum codes.** Our work can be viewed within the broader context of developing algorithms for the decoding of direct sum codes. Given a (say linear) code $\mathcal{C} \subseteq \mathbb{F}_2^n$ and a collection of tuples $W \subseteq [n]^t$, the code $\mathrm{dsum}_W(\mathcal{C})$ with block length $|W|$ is defined as

$$\mathrm{dsum}_W(\mathcal{C}) = \{(z_{w_1} + z_{w_2} + \cdots + z_{w_t})_{w \in W} \mid z \in \mathcal{C}\}.$$

The direct sum operation has been used for several applications in coding and complexity theory [ABN+92], [IW97], [GI01], [IKW09], [DS14], [DDG+15], [Cha16], [DK17], [Aro02]. It is easy to see that if $\mathcal{C}$ is $\varepsilon_0$-balanced for a constant $\varepsilon_0$, then for any $\varepsilon > 0$, choosing $W$ to be a random collection of tuples of size $O(n/\varepsilon^2)$ results in $\mathrm{dsum}_W(\mathcal{C})$ being an $\varepsilon$-balanced code. The challenge in trying to construct good codes using this approach is to find explicit constructions of (sparse) collections $W$ which are "pseudorandom" enough to yield a similar distance amplification as above. On the other hand, the challenge in decoding such codes is to identify notions of "structure" in such collections $W$, which can be exploited by decoding algorithms.

In Ta-Shma's construction [TS17], such a pseudorandom collection $W$ was constructed by considering an expanding graph $G$ over the vertex set $[n]$, and generating $t$-tuples using sufficiently long walks of length $t-1$ over the so-called $s$-wide replacement product of $G$ with another (small) expanding graph $H$. Roughly speaking, this graph product is a generalization of the celebrated zig-zag product [RVW00] but with $s$ different steps of the zig-zag product instead of a single one. Ta-Shma's construction can also be viewed as a clever way of selecting a *sub-collection* of all walks in $G$, which refines an earlier construction suggested by Rozenman and Wigderson [Bog12] (and also analyzed by Ta-Shma) using *all* walks of length $t-1$.

**Identifying structures to facilitate decoding.** For the closely related direct product construction (where the entry corresponding to $w \in W$ is the entire $t$-tuple $(z_{w_1}, \ldots, z_{w_t})$) which amplifies distance but increases the alphabet size, it was proved by Alon et al. [ABN+92] that the resulting code admits a unique decoding algorithm if the incidence graph corresponding to the collection $W$ is a good sampler. Very recently, it was proved by Dinur et al. [DHK+19]

that such a direct product construction admits list decoding if the incidence graph is a "double sampler". However, these results do not apply to direct sum, which achieves distance amplification while preserving the alphabet size.

For the case of direct sum codes, the decoding task can be phrased as a maximum $t$-XOR problem with the additional constraint that the solution must lie in $\mathcal{C}$. More precisely, given $\tilde{y} \in \mathbb{F}_2^W$ within the unique decoding radius of $\mathrm{dsum}_W(\mathcal{C})$, we consider the following optimization problem

$$\operatorname*{argmin}_{z \in \mathcal{C}} \ \Delta(\tilde{y}, \mathrm{dsum}_W(z)),$$

where $\Delta(\cdot,\cdot)$ is the (normalized) Hamming distance. While maximum $t$-XOR is in general hard to solve to even any non-trivial degree of approximation [Hås97], previous work by the authors [AJQ+20] identified a structural condition on $W$ called "splittability" under which the above constraint satisfaction problem can be solved (approximately) resulting in efficient unique and list decoding algorithms. However, by itself the splittability condition is too crude to be applicable to codes such as the ones in Ta-Shma's construction. The requirements it places on the expansion of $G$ are too strong and the framework in [AJQ+20] is only able to obtain algorithms for direct sum codes with rate $2^{-(\log(1/\varepsilon))^2} \ll \varepsilon^{2+\beta}$.

The conceptual contribution of this work can be viewed as identifying a different recursive structure in direct sums generated by expander walks, which allows us to view the construction as giving a sequence of codes $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_\ell$. Here, $\mathcal{C}_0$ is the starting code $\mathcal{C}$ and $\mathcal{C}_\ell$ is the final desired code, and each element in the sequence can be viewed as being obtained via a direct sum operation on the preceding code. Instead of considering a "one-shot" decoding task of finding an element of $\mathcal{C}_0$, this facilitates an iterative approach where at each step we reduce the task of decoding the code $\mathcal{C}_i$ to decoding for $\mathcal{C}_{i-1}$, using the above framework from [AJQ+20]. Such an iterative approach with a sequence of codes was also used (in a very different setting) in a work of Guruswami and Indyk [GI03] constructing codes over a large alphabet which are list decodable in linear time via spectral algorithms.

Another simple and well-known (see e.g., [GI04]) observation, which is very helpful in our setting, is the use of list decoding algorithms for unique decoding. For a code with distance $1/2 - \varepsilon/2$, unique decoding can be obtained by list decoding at a much smaller error radius of (say) $1/2 - 1/8$. This permits a much more efficient application of the framework from [AJQ+20], with a milder dependence on the expansion of the graphs $G$ and $H$ in Ta-Shma's construction, resulting in higher rates. We give a more detailed overview of our approach in Section III.

**Known results for random ensembles.** While the focus in this work is on explicit constructions, there are several known (non-explicit) constructions of random ensembles of binary codes near or achieving the Gilbert–Varshamov bound. Although it is usually straightforward to ensure the desired rate in such constructions, the distance only holds with high probability. Given a sample code from such ensembles, certifying the minimum distance is usually not known to be polynomial time in the block length. Derandomizing such constructions is also a possible avenue for obtaining optimal codes, although such results remain elusive to this date (to the best of our knowledge).

One of the simplest constructions is that of random binary linear codes in which the generator matrix is sampled uniformly. This random ensemble achieves the GV bound with high probability, but its decoding is believed to be computationally hard [MMT11].

Much progress has been made on binary codes by using results for larger alphabet codes [Gur09]. Codes over non-binary alphabets with optimal (or nearly optimal) parameters are available [vL99], [Sti08], [GR06] and thanks to this availability a popular approach to constructing binary codes has been to concatenate such large alphabet codes with binary ones. Thommesen [Tho83] showed that by concatenating Reed–Solomon (RS) codes with random binary codes (one random binary code for each position of the outer RS code) it is possible to achieve the GV bound. Note that Thommesen codes arise from a more structured ensemble than random binary linear codes. This additional structure enabled Guruswami and Indyk [GI04] to obtain efficient decoding algorithms for the non-explicit

Thommesen codes (whose minimum distance is not known to admit efficient certification). This kind of concatenation starting from a large alphabet code and using random binary codes, which we refer as Thommesen-like, has been an important technique in tackling binary code constructions with a variety of properties near or at the GV bound. An important drawback in several such Thommesen-like code constructions is that they end up being non-explicit (unless efficient derandomization or brute-force is viable).

Using a Thommesen-like construction, Gopi et al. [GKO+17] showed non-explicit constructions of locally testable and locally correctable binary codes approaching the GV bound. More recently, again with a Thommesen-like construction, Hemenway et al. [HRW17] obtained non-explicit near linear time unique decodable codes at the GV bound improving the running time of Guruswami and Indyk [GI04] (and also the decoding rates).

There are also non-explicit constructions known to achieve list decoding capacity [GR08], [MRRZ+19] (being concatenated or LDPC/Gallager [Gal62] is not an obstruction to achieve capacity). Contrary to the other results in this subsection, Guruswami and Rudra [Gur05], [GR06], [Gur09], also using a Thommesen-like construction, obtained explicit codes that are efficiently list decodable from radius $1/2 - \varepsilon$ with rate $\Omega(\varepsilon^3)$. This was done by concatenating the so-called folded Reed–Solomon codes with a derandomization of a binary ensemble of random codes.

**Results for non-adversarial error models.** All the results mentioned above are for the adversarial error model of Hamming [Ham50], [Gur10]. In the setting of random corruptions (Shannon model), the situation seems to be better understood thanks to the seminal result on explicit polar codes of Arikan [Ari09]. More recently, in another breakthrough Guruswami et al. [GRY19] showed that polar codes can achieve almost linear time decoding with near optimal convergence to capacity for the binary symmetric channel. This result gives an explicit code construction achieving parameter trade-offs similar to Shannon's randomized construction [Sha48] while also admitting very efficient encoding and decoding. Explicit capacity-achieving constructions are also known for bounded memory channels [SKS19] which restrict the power of the adversary and thus interpolate between the Shannon and Hamming models.

## II. Preliminaries and Notation

### A. Codes

We briefly recall some standard code terminology. Given $z, z' \in \mathbb{F}_2^n$, recall that the relative Hamming distance between $z$ and $z'$ is $\Delta(z, z') := \left|\{i \mid z_i \neq z'_i\}\right| / n$. A binary code is any subset $\mathcal{C} \subseteq \mathbb{F}_2^n$. The distance of $\mathcal{C}$ is defined as $\Delta(\mathcal{C}) := \min_{z \neq z'} \Delta(z, z')$ where $z, z' \in \mathcal{C}$. We say that $\mathcal{C}$ is a linear code if $\mathcal{C}$ is a linear subspace of $\mathbb{F}_2^n$. The rate of $\mathcal{C}$ is $\log_2(|\mathcal{C}|)/n$.

Instead of discussing the distance of a binary code, it will often be more natural to phrase results in terms of its bias.

**Definition II.1** (Bias)**.** *The* bias *of a word $z \in \mathbb{F}_2^n$ is defined as* $\mathrm{bias}(z) := \left|\mathbb{E}_{i \in [n]} (-1)^{z_i}\right|$. *The bias of a code $\mathcal{C}$ is the maximum bias of any non-zero codeword in $\mathcal{C}$.*

**Definition II.2** ($\varepsilon$-balanced Code)**.** *A code $\mathcal{C}$ with* $\mathrm{bias}(\mathcal{C}) \leq \varepsilon$ *is known as an* $\varepsilon$-balanced code.

### B. Direct Sum Lifts

Starting from a code $\mathcal{C} \subseteq \mathbb{F}_2^n$, we amplify its distance by considering the *direct sum lifting* operation based on a collection $W(k) \subseteq [n]^k$. The direct sum lifting maps each codeword of $\mathcal{C}$ to a new word in $\mathbb{F}_2^{|W(k)|}$ by taking the $k$-XOR of its entries on each element of $W(k)$.

**Definition II.3** (Direct Sum Lifting)**.** *Let $W(k) \subseteq [n]^k$. For $z \in \mathbb{F}_2^n$, we define the* direct sum lifting *as* $\mathrm{dsum}_{W(k)}(z) = y$ *such that $y_{\mathfrak{s}} = \sum_{i \in \mathfrak{s}} z_i$ for all $\mathfrak{s} \in W(k)$. The direct sum lifting of a code $\mathcal{C} \subseteq \mathbb{F}_2^n$ is*

$$\mathrm{dsum}_{W(k)}(\mathcal{C}) = \{\mathrm{dsum}_{W(k)}(z) \mid z \in \mathcal{C}\}.$$

*We will omit $W(k)$ from this notation when it is clear from context.*

**Remark II.4.** *We will be concerned with collections $W(k) \subseteq [n]^k$ arising from length-$(k-1)$ walks on expanding structures (mostly on the s-wide replacement product of two expander graphs).*

We will be interested in cases where the direct sum lifting reduces the bias of the base code; in [TS17], structures with such a property are called *parity samplers*, as they emulate the

reduction in bias that occurs by taking the parity of random samples.

**Definition II.5** (Parity Sampler)**.** *A collection* $W(k) \subseteq [n]^k$ *is called an* $(\varepsilon_0, \varepsilon)$-parity sampler *if for all* $z \in \mathbb{F}_2^n$ *with* $\mathrm{bias}(z) \leq \varepsilon_0$*, we have* $\mathrm{bias}(\mathrm{dsum}_{W(k)}(z)) \leq \varepsilon$*.*

### C. Linear Algebra Conventions

All vectors considered in this paper are taken to be column vectors, and are multiplied on the left with any matrices or operators acting on them. Consequently, given an indexed sequence of operators $\mathsf{G}_{k_1}, \ldots, \mathsf{G}_{k_2}$ (with $k_1 \leq k_2$) corresponding to steps $k_1$ through $k_2$ of a walk, we expand the product $\prod_{i=k_1}^{k_2} G_i$ as

$$\prod_{i=k_1}^{k_2} G_i \;:=\; G_{k_2} \cdots G_{k_1}.$$

Unless otherwise stated, all inner products for vectors in coordinate spaces are taken to be with respect to the (uniform) probability measure on the coordinates. Similarly, all inner products for functions are taken to be with respect to the uniform measure on the inputs. All operators considered in this paper are normalized to have singular values at most 1.

### D. Overview of the s-wide Replacement Product

Ta-Shma's code construction is based on the so-called $s$-wide replacement product [TS17]. This is a derandomization of random walks on a graph $G$ that will be defined via a product operation of $G$ with another graph $H$ (see the full version for a formal definition). We will refer to $G$ as the *outer* graph and $H$ as the *inner* graph in this construction.

Let $G$ be a $d_1$-regular graph on vertex set $[n]$ and $H$ be a $d_2$-regular graph on vertex set $[d_1]^s$, where $s$ is any positive integer. Suppose the neighbors of each vertex of $G$ are labeled 1, 2, …, $d_1$. For $v \in V(G)$, let $v_G[j]$ be the $j$-th neighbor of $v$. The $s$-wide replacement product is defined by replacing each vertex of $G$ with a copy of $H$, called a "cloud". While the edges within each cloud are determined by $H$, the edges between clouds are based on the edges of $G$, which we will define via operators $\mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_{s-1}$. The $i$-th operator $\mathsf{G}_i$ specifies one inter-cloud edge for each vertex $(v, (a_0, \ldots, a_{s-1})) \in V(G) \times V(H)$, which goes to the cloud whose $G$ component is $v_G[a_i]$, the neighbor of $v$ in $G$ indexed by the $i$-th coordinate of the $H$ component. (We will resolve the question of what happens to the $H$ component after taking such a step momentarily.)

Walks on the $s$-wide replacement product consist of steps with two different parts: an intra-cloud part followed by an inter-cloud part. All of the intra-cloud substeps simply move to a random neighbor in the current cloud, which corresponds to applying the operator $\mathsf{I} \otimes \mathsf{A}_H$, where $\mathsf{A}_H$ is the normalized adjacency matrix of $H$. The inter-cloud substeps are all deterministic, with the first moving according to $\mathsf{G}_0$, the second according to $\mathsf{G}_1$, and so on, returning to $\mathsf{G}_0$ for step number $s+1$. The operator for such a walk taking $t-1$ steps on the $s$-wide replacement product is

$$\prod_{i=0}^{t-2} \mathsf{G}_{i \bmod s}(\mathsf{I} \otimes \mathsf{A}_H).$$

Observe that a walk on the $s$-wide replacement product yields a walk on the outer graph $G$ by recording the $G$ component after each step of the walk. The number of $(t-1)$-step walks on the $s$-wide replacement product is

$$|V(G)| \cdot |V(H)| \cdot d_2^{t-1} = n \cdot d_1^s \cdot d_2^{t-1},$$

since a walk is completely determined by its intra-cloud steps. If $d_2$ is much smaller than $d_1$ and $t$ is large compared to $s$, this is less than $n d_1^{t-1}$, the number of $(t-1)$-step walks on $G$ itself. Thus the $s$-wide replacement product will be used to simulate random walks on $G$ while requiring a reduced amount of randomness (of course this simulation is only possible under special conditions, namely, when we are uniformly distributed on each cloud).

## III. Proof Overview

The starting point for our work is the framework developed in [AJQ+20] for decoding direct sum codes, obtained by starting from a code $\mathcal{C} \subseteq \mathbb{F}_2^n$ and considering all parities corresponding to a set of $t$-tuples $W(t) \subseteq [n]^t$. Ta-Shma's near optimal $\varepsilon$-balanced codes are constructed by starting from a code with constant rate and constant distance and considering such a direct sum lifting. The set of tuples $W(t)$ in his construction corresponds to a set of walks of length $t-1$ on the $s$-wide replacement product of an expanding graph $G$ with vertex set $[n]$ and a smaller expanding

graph $H$. The $s$-wide replacement product can be thought of here as a way of constructing a much smaller pseudorandom subset of the set of all walks of length $t-1$ on $G$, which yields a similar distance amplification for the lifted code.

**The simplified construction with expander walks.** While we analyze Ta-Shma's construction later in the paper, it is instructive to first consider a $W(t)$ simply consisting of all walks of length $t-1$ on an expander. This construction, based on a suggestion of Rozenman and Wigderson [Bog12], was also analyzed by Ta-Shma [TS17] and can be used to obtain $\varepsilon$-balanced codes with rate $\Omega(\varepsilon^{4+o(1)})$. It helps to illustrate many of the conceptual ideas involved in our proof, while avoiding some technical issues.

Let $G$ be a $d$-regular expanding graph with vertex set $[n]$ and the (normalized) second singular value of the adjacency operator $\mathsf{A}_G$ being $\lambda$. Let $W(t) \subseteq [n]^t$ denote the set of $t$-tuples corresponding to all walks of length $t-1$, with $N = |W(t)| = n \cdot d^{t-1}$. Ta-Shma proves that for all $z \in \mathbb{F}_2^n$, $W(t)$ satisfies

$$\mathrm{bias}(z) \leq \varepsilon_0 \Rightarrow \mathrm{bias}(\mathrm{dsum}_{W(t)}(z)) \leq (\varepsilon_0 + 2\lambda)^{\lfloor (t-1)/2 \rfloor},$$

i.e., $W(t)$ is an $(\varepsilon_0, \varepsilon)$-*parity sampler* for $\varepsilon = (\varepsilon_0 + 2\lambda)^{\lfloor (t-1)/2 \rfloor}$. Choosing $\varepsilon_0 = 0.1$ and $\lambda = 0.05$ (say), we can choose $d = O(1)$ and obtain the $\varepsilon$-balanced code $\mathcal{C}' = \mathrm{dsum}_{W(t)}(\mathcal{C})$ with rate $d^{-(t-1)} = \varepsilon^{O(1)}$ (although the right constants matter a lot for optimal rates).

**Decoding as constraint satisfaction.** The starting point for our work is the framework in [AJQ+20] which views the task of decoding $\tilde{y}$ with $\Delta(\mathcal{C}', \tilde{y}) < (1-\varepsilon)/4 - \delta$ (where the distance of $\mathcal{C}'$ is $(1-\varepsilon)/2$) as an instance of the MAX t-XOR problem (see Fig. 1). The goal is to find

$$\operatorname*{argmin}_{z \in \mathcal{C}} \Delta\left(\mathrm{dsum}_{W(t)}(z), \tilde{y}\right),$$

which can be rephrased as

$$\operatorname*{argmax}_{z \in \mathcal{C}} \mathop{\mathbb{E}}_{w=(i_1,\ldots,i_t) \in W(t)} \left[\mathbb{1}_{\{z_{i_1} + \cdots + z_{i_t} = \tilde{y}_w\}}\right].$$

It is possible to ignore the condition that $z \in \mathcal{C}$ if the collection $W(t)$ is a slightly stronger parity sampler. For any solution $\tilde{z} \in \mathbb{F}_2^n$ (not necessarily in $\mathcal{C}$) such that

$$\Delta(\mathrm{dsum}_{W(t)}(\tilde{z}), \tilde{y}) < \frac{1-\varepsilon}{4} + \delta,$$

we have

$$\Delta(\mathrm{dsum}_{W(t)}(\tilde{z}), \mathrm{dsum}_{W(t)}(z)) < \frac{1-\varepsilon}{2}$$

by the triangle inequality, and thus $\mathrm{bias}(\mathrm{dsum}_{W(t)}(z - \tilde{z})) > \varepsilon$. If $W(t)$ is not just an $(\varepsilon_0, \varepsilon)$-parity sampler, but in fact a $((1+\varepsilon_0)/2, \varepsilon)$-parity sampler, this would imply $\mathrm{bias}(z - \tilde{z}) > (1+\varepsilon_0)/2$. Thus, $\Delta(z, \tilde{z}) < (1-\varepsilon_0)/4$ (or $\Delta(z, \bar{\tilde{z}}) < (1-\varepsilon_0)/4$) and we can use a unique decoding algorithm for $\mathcal{C}$ to find $z$ given $\tilde{z}$.

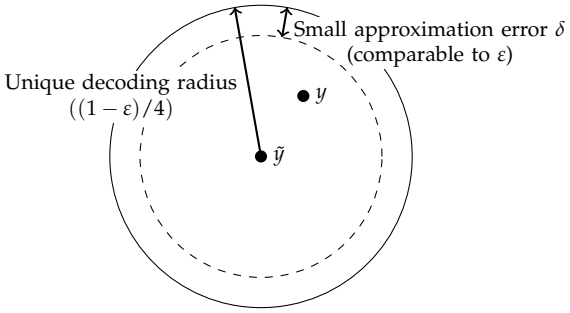


Figure 1. Unique decoding ball along with error from approximation.

The task of finding such a $z \in \mathcal{C}$ boils down to finding a solution $\tilde{z} \in \mathbb{F}_2^n$ to a MAX t-XOR instance, up to a an additive loss of $O(\delta)$ in the fraction of constraints satisfied by the optimal solution. While this is hard to do in general [Hås01], [Gri01], [AJQ+20] (building on [AJT19]) show that this can be done if the instance satisfies a special property called *splittability*. To define this, we let $W[t_1, t_2] \subset [n]^{t_2 - t_1 + 1}$ denote the collection of $(t_2 - t_1 + 1)$-tuples obtained by considering the indices between $t_1$ and $t_2$ for all tuples in $W(t)$. We also assume that all $w \in W[t_1, t_2]$ can be extended to the same number of tuples in $W(t)$ (which is true for walks).

**Definition III.1** (Splittability (informal))**.** *A collection $W(t) \subseteq [n]^t$ is said to be $\tau$-splittable, if $t = 1$ (base case) or there exists $t' \in [t-1]$ such that:*

1) *The matrix $\mathsf{S} \in \mathbb{R}^{W[1,t'] \times W[t'+1,t]}$ defined by $\mathsf{S}(w, w') = \mathbb{1}_{\{ww' \in W\}}$ has normalized second singular value at most $\tau$ (where $ww'$ denotes the concatenated tuple).*
2) *The collections $W[1, t']$ and $W[t'+1, t]$ are $\tau$-splittable.*

For example, considering walks in $G$ of

length 3 ($t = 4$) and $t' = 2$, we get that $W[1,2] = W[3,4] = E$, the set of oriented edges in $G$. Also $\mathsf{S}(w,w') = 1$ if and only if the second vertex of $w$ and first vertex of $w'$ are adjacent in $G$. Thus, up to permutation of rows and columns, we can write the normalized version of $\mathsf{S}$ as $\mathsf{A}_G \otimes \mathsf{J}_d/d$ where $\mathsf{A}_G$ is normalized adjacency matrix of $G$ and $\mathsf{J}_d$ denotes the $d \times d$ matrix of 1s. Hence such a $W(t)$ satisfies $\sigma_2(\mathsf{S}) \leq \tau$ with $\tau = \sigma_2(\mathsf{A}_G)$, and a similar proof works for walks of all lengths.

The framework in [AJQ+20] and [AJT19] gives that if $W(t)$ is $\tau$-splittable for $\tau = (\delta/2^t)^{O(1)}$, then the above instance of MAX t-XOR can be solved to additive error $O(\delta)$ using the Sum-of-Squares (SOS) SDP hierarchy. Broadly speaking, splittability allows one to (recursively) treat instances as expanding instances of problems with two "tuple variables" in each constraint, which can then be analyzed using known algorithms for 2-CSPs [BRS11], [GS11] in the SOS hierarchy. Combined with parity sampling, this yields a unique decoding algorithm. Crucially, this framework can also be extended to perform *list decoding*[3] up to a radius of $1/2 - \sqrt{\varepsilon} - \delta$ under a similar condition on $\tau$, which will be very useful for our application.

While the above can yield decoding algorithms for suitably expanding $G$, the requirement on $\tau$ (and hence on $\lambda$) makes the rate much worse. We need $\delta = O(\varepsilon)$ (for unique decoding) and $t = O(\log(1/\varepsilon))$ (for parity sampling), which requires $\lambda = \varepsilon^{\Omega(1)}$, yielding only a quasipolynomial rate for the code (recall that we could take $\lambda = O(1)$ earlier yielding polynomial rates).

**Unique decoding: weakening the error requirement.** We first observe that it is possible to get rid of the dependence $\delta = O(\varepsilon)$ above by using the *list decoding* algorithm for unique decoding. It suffices to take $\delta = 0.1$ and return the closest element from the the list of all codewords up to an error radius $1/2 - \sqrt{\varepsilon} - 0.1$, if we are promised that $\Delta(\tilde{y}, \mathcal{C})$ is within the unique decoding radius (see Fig. 2). However, this alone does not improve the rate as we still need the splittability (and hence $\lambda$) to be $2^{-\Omega(t)}$ with $t = O(\log(1/\varepsilon))$.

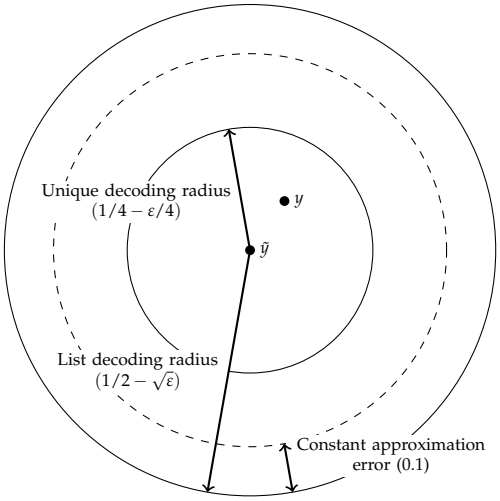


Figure 2. Unique decoding and list decoding balls along with error from approximation. Note that the list decoding ball contains the unique decoding ball even after allowing for a relatively large amount of error.

**Code cascades: handling the dependence on walk length.** To avoid the dependence of the expansion on the length $t-1$ of the walk (and hence on $\varepsilon$), we avoid the "one-shot" decoding above, and instead consider a sequence of intermediate codes between $\mathcal{C}$ and $\mathcal{C}'$. Consider the case when $t = k^2$, and instead of computing $t$-wise sums of bits in each $z \in \mathbb{F}_2^n$, we first compute $k$-wise sums according to walks of length $k-1$ on $G$, and then a $k$-wise sum of these values. In fact, the second sum can also be thought of as arising from a length $k-1$ walk on a different graph, with vertices corresponding to (directed) walks with $k$ vertices in $G$, and edges connecting $w$ and $w'$ when the last vertex of $w$ is connected to the first one in $w'$ (this is similar to the matrix considered for defining splittability). We can thus think of a sequence of codes $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2$ with $\mathcal{C}_0 = \mathcal{C}$ and $\mathcal{C}_2 = \mathcal{C}'$, and both $\mathcal{C}_1$ and $\mathcal{C}_2$ being $k$-wise direct sums. More generally, when $t = k^\ell$ for an appropriate constant $k$ we can think of a sequence $\mathcal{C} = \mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_\ell = \mathcal{C}'$, where each is an $k$-wise direct sum of the previous code, obtained via walks of length $k-1$ (hence $k$ vertices) in an appropriate graph. We refer to such sequences (defined formally in the full version) as *code cascades* (see Fig. 3).

Instead of applying the decoding frame-

[3] While unique decoding can be thought of as recovering a single solution to a constraint satisfaction problem, the goal in the list decoding setting can be thought of as obtaining a "sufficiently rich" set of solutions which forms a good cover. This is achieved in the framework by adding an entropic term to the semidefinite program, which ensures that the SDP solution satisfies such a covering property.

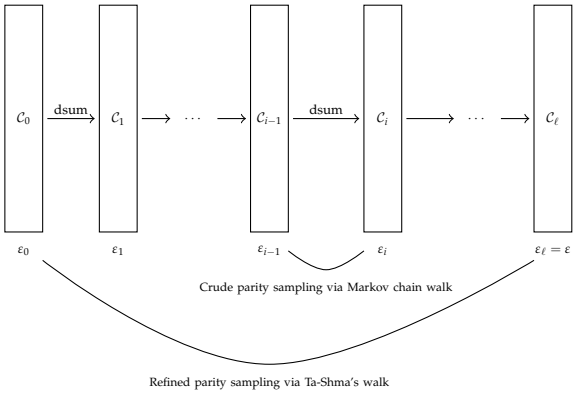


Figure 3. Code cascading.

work above to directly reduce the decoding of a corrupted codeword from $\mathcal{C}'$ to the unique decoding problem in $\mathcal{C}$, we apply it at each level of a cascade, reducing the unique decoding problem in $\mathcal{C}_i$ to that in $\mathcal{C}_{i-1}$. If the direct sum at each level of the cascade is an $(\eta_0, \eta)$-parity sampler, the list decoding algorithm at radius $1/2 - \sqrt{\eta}$ suffices for unique decoding even if $\eta$ is a (sufficiently small) constant independent of $\varepsilon$. This implies that we can take $k$ to be a (suitably large) constant. This also allows the splittability (and hence $\lambda$) to be $2^{-O(k)} = \Omega(1)$, yielding polynomial rates. We present the reduction using cascades and the parameter choices in the full version.

While the above allows for polynomial rate, the *running time* of the algorithm is still exponential in the number of levels $\ell$ (which is $O(\log t) = O(\log\log(1/\varepsilon))$) since the list decoding for each level potentially produces a list of size $\mathrm{poly}(n)$, and recursively calls the decoding algorithm for the previous level on each element of the list. We obtain a fixed polynomial time algorithm by "pruning" the list at each level of the cascade before invoking the decoding algorithm for the previous level, while only slightly increasing the parity sampling requirements. The details are contained in the full version.

**Working with Ta-Shma's construction.** Finally, to obtain near-optimal rates, we need to work with with Ta-Shma's construction, where the set of tuples $W(t) \subseteq [n]^t$ corresponds to walks arising from an $s$-wide replacement product of $G$ with another expanding graph $H$. One issue that arises is that the collection of walks $W(t)$ as defined in [TS17] does not satisfy the important splittability condition required by our algorithms. However, this turns out to be easily fixable by modifying each step in Ta-Shma's construction to be exactly according to the zig-zag product of Reingold, Vadhan and Wigderson [RVW00]. We present Ta-Shma's construction and this modification in the full version.

We also verify that the tuples given by Ta-Shma's construction satisfy the conditions for applying the list decoding framework, in the full version. While the sketch above stated this in terms of splittability, the results in [AJQ+20] are in terms of a more technical condition called *tensoriality*. We show in the full version that this is indeed implied by splittability, and also prove splittability for (the modified version of) Ta-Shma's construction.

## IV. Code Cascading

A code cascade is a sequence of codes generated by starting with a base code $\mathcal{C}_0$ and recursively applying lifting operations.

**Definition IV.1.** *We say that a sequence of codes $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_\ell$ is a* code cascade *provided $\mathcal{C}_i = \mathrm{dsum}_{W_i(t_i)}(\mathcal{C}_{i-1})$ for every $i \in [\ell]$. Each $W_i(t_i)$ is a subset of $[n_{i-1}]^{t_i}$, where $n_{i-1} = |W_{i-1}(t_{i-1})|$ is the block length of the code $\mathcal{C}_{i-1}$.*

Let us see how code cascades may be useful for decoding. Suppose we wish to lift the code $\mathcal{C}_0$ to $\mathcal{C}_\ell$, and there is some $W(t) \subseteq [n_0]^t$ such that $\mathcal{C}_\ell = \mathrm{dsum}_{W(t)}(\mathcal{C}_0)$. In our case of bias boosting, this $t$ will depend on the target bias $\varepsilon$. However, the expansion requirement of the list-decoding framework of [AJQ+20] has a poor dependence on $t$. A way to work around this issue is to go from $\mathcal{C}_0$ to $\mathcal{C}_\ell$ via a code cascade as above such that each $t_i$ is a constant independent of the final bias but $\prod_{i=1}^{\ell} t_i = t$ (which means $\ell$ depends on $\varepsilon$). The final code $\mathcal{C}_\ell$ of the cascade is the same as the code obtained from length-$(t-1)$ walks. While decoding will now become an $\ell$-level recursive procedure, the gain from replacing $t$ by $t_i$ will outweigh this loss, as we discuss below.

### A. Warm-up: Code Cascading Expander Walks

We now describe the code cascading construction and unique decoding algorithm in

more detail. Let $G = (V, E)$ be a $d$-regular graph with uniform distribution over the edges. Let $m$ be a sufficiently large positive integer, which will be the number of vertices of the walks used for the lifting between consecutive codes in the cascade. At first, it will be crucial that we can take $m = O(1)$ so that the triangle inequality arising from the analysis of the lifting between two consecutive codes involves a constant number of terms. We construct a recursive family of codes as follows.

- Start with a code $\mathcal{C}_0$ which is linear and has constant bias $\varepsilon_0$.
- Define the code $\mathcal{C}_1 = \mathrm{dsum}_{W(m)}(\mathcal{C}_0)$, which is the direct sum lifting over the collection $W(m)$ of all length-$(m-1)$ walks on $G$ using the code $\mathcal{C}_0$.
- Let $\widehat{G}_i = (V_i, E_i)$ be the (directed) graph where $V_i$ is the collection of all walks on $m^i$ vertices on $G$ with two walks $(v_1, \ldots, v_{m^i})$ and $(u_1, \ldots, u_{m^i})$ connected iff $v_{m^i}$ is adjacent to $u_1$ in $G$.
- Define $\mathcal{C}_i$ to be the direct sum lifting on the collection $W_i(m)$ of all length-$(m-1)$ walks on $G_{i-1}$ using the code $\mathcal{C}_{i-1}$, i.e., $\mathcal{C}_i = \mathrm{dsum}_{W_i(m)}(\mathcal{C}_{i-1})$.
- Repeat this process to yield a code cascade $\mathcal{C}_0, \ldots, \mathcal{C}_\ell$.

Thanks to the definition of the graphs $\widehat{G}_i$ and the recursive nature of the construction, the final code $\mathcal{C}_\ell$ is the same as the code obtained from $\mathcal{C}_0$ by taking the direct sum lifting over all walks on $t = m^\ell$ vertices of $G$. We can use Ta-Shma's analysis (building on the ideas of Rozenman and Wigderson [Bog12]) for the simpler setting of walks over a single expander graph to determine the amplification in bias that occurs in going from $\mathcal{C}_0$ all the way to $\mathcal{C}_\ell$.

**Theorem IV.2** (Adapted from Ta-Shma [TS17])**.** *Let $\mathcal{C}$ be an $\varepsilon_0$-balanced linear code, and let $\mathcal{C}' = \mathrm{dsum}_{W(t)}(\mathcal{C})$ be the direct sum lifting of $\mathcal{C}$ over the collection of all length-$(t-1)$ walks $W(t)$ on a graph $G$. Then*

$$\mathrm{bias}(\mathcal{C}') \leq (\varepsilon_0 + 2\sigma_2(G))^{\lfloor (t-1)/2 \rfloor}.$$

If $\sigma_2(G) \leq \varepsilon_0/2$ and $\ell = \left\lceil \log_m(2\log_{2\varepsilon_0}(\varepsilon) + 3) \right\rceil$, taking $t = m^\ell \geq 2\log_{2\varepsilon_0}(\varepsilon) + 3$ in the above theorem shows that the final code $\mathcal{C}_\ell$ is $\varepsilon$-balanced. Observe that the required expansion of the graph $G$ only depends on the constant initial bias $\varepsilon_0$, not on the desired final bias $\varepsilon$. It will be important for being able to decode with better parameters that both $\sigma_2(G)$ and $m$ are constant with respect to $\varepsilon$; only $\ell$ depends on the final bias (with more care we can make $\sigma_2(G)$ depend on $\varepsilon$, but we restrict this analysis to Ta-Shma's refined construction on the $s$-wide replacement product).

As mentioned before, to uniquely decode $\mathcal{C}_\ell$ we will inductively employ the list decoding machinery for expander walks from [AJQ+20]. The list decoding algorithm can decode a direct sum lifting $\mathcal{C}' = \mathrm{dsum}_{W(m)}(\mathcal{C})$ as long as the graph $G$ is sufficiently expanding, the walk length $m-1$ is large enough, and the base code $\mathcal{C}$ has an efficient unique decoding algorithm (see the full version for details).

The expansion requirement ultimately depends on the desired list decoding radius of $\mathcal{C}'$, or more specifically, on how close the list decoding radius is to $1/2$. If the distance of $\mathcal{C}'$ is at most $1/2$, its unique decoding radius is at most $1/4$, which means list decoding at the unique decoding radius is at a constant difference from $1/2$ and thus places only a constant requirement on the expansion of $G$. In the case of the code cascade $\mathcal{C}_i = \mathrm{dsum}_{W_i(m)}(\mathcal{C}_{i-1})$, unique decoding of $\mathcal{C}_{i-1}$ is guaranteed by the induction hypothesis. It is not too difficult to see that each graph $\widehat{G}_i$ will have the same second singular value as $G$, so we can uniquely decode $\mathcal{C}_i$ if $G$ meets the constant expansion requirement and $m$ is sufficiently large.

### *B. Code Cascading Ta-Shma's Construction*

We will now describe how to set up a code cascade based on walks on an $s$-wide replacement product. Consider the $s$-wide replacement product of the outer graph $G$ with the inner graph $H$. The first $s$ walk steps are given by the walk operator

$$\prod_{i=0}^{s-1} (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i(\mathsf{I} \otimes \mathsf{A}_H).$$

Let $\mathsf{A}_{s-1} := (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{s-2}(\mathsf{I} \otimes \mathsf{A}_H) \cdots (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_0(\mathsf{I} \otimes \mathsf{A}_H)$. If the total walk length $t-1$ is a multiple of $s$, the walks are generated using the operator

$$\left((\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{s-1}(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{A}_{s-1}\right)^{(t-1)/s}.$$

Here $(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{s-1}(\mathsf{I} \otimes \mathsf{A}_H)$ is used as a "binding" operator to connect two walks containing $s$ vertices at level $\mathcal{C}_2$, $s^2$ vertices at level $\mathcal{C}_3$, and so on. More precisely, we form the following code cascade.

- $\mathcal{C}_0$ is an $\varepsilon_0$-balanced linear code efficiently uniquely decodable from a constant radius.
- $\mathcal{C}_1 = \mathrm{dsum}_{W_1(s)}(\mathcal{C}_0)$, where $W_1(s)$ is the set of length-(s-1) walks given by the operator

$$\underbrace{(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{s-2}(\mathsf{I} \otimes \mathsf{A}_H)}_{(s-2)\text{th step}} \cdots \underbrace{(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_0(\mathsf{I} \otimes \mathsf{A}_H)}_{0\text{th step}}.$$

- $\mathcal{C}_2 = \mathrm{dsum}_{W_2(s)}(\mathcal{C}_1)$, where $W_2(s)$ is the set of length-$(s-1)$ walks over the vertex set $W_1(s)$ (with the latter being the set of length-$(s-1)$ walks on the replacement product graph as mentioned above).
- $\mathcal{C}_{i+1} = \mathrm{dsum}_{W_{i+1}(s)}(\mathcal{C}_i)$, where $W_{i+1}(s)$ is the set of length-$(s-1)$ walks [4] over the vertex set $W_i(s)$. Similarly to the cascade of expander walks above, the lift can be thought of as being realized by taking walks using a suitable operator analogous to $\widehat{G}_i$. Since its description is more technical we postpone its definition to the full version where it appears in full detail.
- $\mathcal{C}_\ell$ denotes the final code in the sequence, which will later be chosen so that its bias is at most $\varepsilon$.


## Acknowledgement

We thank Amnon Ta-Shma for suggesting the problem of decoding in fixed polynomial running time (with the exponent of $N$ independent of $\varepsilon$) which led us to think about this regime in Theorem I.1. Part of this work was done when some of the authors were visiting the Simons Institute in Berkeley, and we thank them for their kind hospitality. Fernando and Shashanks were supported in part by NSF grant CCF-1816372. Madhur Tulsiani was supported by NSF grant CCF-1816372.


## References


[ABN+92] 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*, 28:509–516, 1992. 2

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost $k$-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992. 1

[AJQ+20] Vedat Levi Alev, Fernando Granha Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. List decoding of direct sum codes. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms*, pages 1412–1425. SIAM, 2020. 3, 5, 6, 7, 8, 9

[AJT19] Vedat Levi Alev, Fernando Granha Jeronimo, and Madhur Tulsiani. Approximating constraint satisfaction problems on high-dimensional expanders. In *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science*, pages 180–201, 2019. 6, 7

[Ari09] E. Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, July 2009. 4

[Aro02] Sanjeev Arora. How NP got a new definition: a survey of probabilistically checkable proofs. In *Proceedings of the International Congress of Mathematicians*, pages 637–648, 2002. Volume 3. 2

[Bog12] Andrej Bogdanov. A different way to improve the bias via expanders. Lecture notes, April 2012. URL: http://www.cse.cuhk.edu.hk/~andrejb/csc5060/notes/12L12.pdf. 2, 6, 9

[BRS11] Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science*, pages 472–481, 2011. 7

[Cha16] Siu On Chan. Approximation resistance from pairwise-independent subgroups. *J. ACM*, 63(3), August 2016. 2

[DDG+15] Roee David, Irit Dinur, Elazar Goldenberg, Guy Kindler, and Igor Shinkar. Direct sum testing. ITCS '15, pages 327–336, New York, NY, USA, 2015. ACM. 2


[4] For simplicity we chose the number of vertices in all walks of the cascade to be $s$, but it could naturally be some $s_i \in \mathbb{N}$ depending on $i$.

[Del75] P. Delsarte. The association schemes of coding theory. In *Combinatorics*, pages 143–161. Springer Netherlands, 1975. 1

[DHK+19] Irit Dinur, Prahladh Harsha, Tali Kaufman, Inbal Livni Navon, and Amnon Ta-Shma. List decoding with double samplers. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms*, pages 2134–2153, 2019. 2

[DK17] Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science*, pages 974–985, 2017. 2

[DS14] Irit Dinur and David Steurer. Direct product testing. In *Proceedings of the 29th IEEE Conference on Computational Complexity*, CCC '14, pages 188–196, 2014. 2

[Gal62] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962. 4

[GI01] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 658–667, 2001. 2

[GI03] Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, 2003. 3

[GI04] Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting Gilbert-Varshamov bound for low rates. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 756–757, 2004. 3, 4

[Gil52] E.N. Gilbert. A comparison of signalling alphabets. *Bell System Technical Journal*, 31:504–522, 1952. 1

[GKO+17] Sivakanth Gopi, Swastik Kopparty, Rafael Oliveira, Noga Ron-Zewi, and Shubhangi Saraf. Locally testable and locally correctable codes approaching the Gilbert-Varshamov bound. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2073–2091, 2017. 4

[GR06] Venkatesan Guruswami and Atri Rudra. Explicit capacity-achieving list-decodable codes. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 1–10, 2006. 1, 3, 4

[GR08] Venkatesan Guruswami and Atri Rudra. Concatenated codes can achieve list-decoding capacity. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 258–267, 2008. 4

[Gri01] Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theor. Comput. Sci.*, 259(1-2):613–622, 2001. 6

[GRS19] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. Available at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/index.html, 2019. 1

[GRY19] Venkatesan Guruswami, Andrii Riazanov, and Min Ye. Arikan meets Shannon: Polar codes with near-optimal convergence to channel capacity. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:154, 2019. 4

[GS11] Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with psd objectives. In *FOCS*, pages 482–491, 2011. 7

[Gur05] Venkatesan Guruswami. Algebraic-geometric generalizations of the Parvaresh-Vardy codes. *Electronic Colloquium on Computational Complexity (ECCC)*, (132), 2005. 4

[Gur09] Venkatesan Guruswami. List decoding of binary codes–a brief survey of some recent results. In *Coding and Cryptology*, pages 97–106. Springer Berlin Heidelberg, 2009. 1, 2, 3, 4

[Gur10] Venkatesan Guruswami. Bridging Shannon and Hamming: List error-correction with optimal rate. In *ICM*, 2010. 1, 4

[Ham50] Richard Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, 1950. 4

[Hås97] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 1–10, 1997. 3

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. 6

[HRW17] B. Hemenway, N. Ron-Zewi, and M. Wootters. Local list recovery of high-rate tensor codes applications. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science*, pages 204–215, Oct 2017. 4

[IKW09] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. New direct-product testers and 2-query PCPs. In *Proceedings of the 41st ACM Symposium on Theory of Computing*, STOC '09, pages 131–140, 2009. 2

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ unless $E$ has sub-exponential circuits. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 220–229, 1997. 2

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In *Advances in Cryptology – ASIACRYPT 2011*, pages 107–124, 2011. 3

[MRRW77] R. McEliece, E. Rodemich, H. Rumsey, and L. Welch. New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE Transactions on Information Theory*, 23(2):157–166, 1977. 1

[MRRZ+19] Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. LDPC codes achieve list decoding capacity, 2019. arXiv:1909.06430. 4

[NN90] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd ACM Symposium on Theory of Computing*, pages 213–223, 1990. 1

[NS09] Michael Navon and Alex Samorodnitsky. Linear programming bounds for codes via a covering argument. *Discrete Comput. Geom.*, 41(2):199–207, March 2009. 1

[RVW00] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, 2000. 2, 8

[Sha48] Claude Shannon. A mathematical theory of communications. *Bell System Technical Journal*, 27:379–423, 623–656, 1948. 4

[SKS19] Ronen Shaltiel, Swastik Kopparty, and Jad Silbak. Quasilinear time list-decodable codes for space bounded channels. 2019. 4

[Sti08] Henning Stichtenoth. *Algebraic Function Fields and Codes*. Springer Publishing Company, Incorporated, 2nd edition, 2008. 3

[Tho83] C. Thommesen. The existence of binary linear concatenated codes with Reed- Solomon outer codes which asymptotically meet the Gilbert-Varshamov bound. *IEEE Transactions on Information Theory*, 29(6):850–853, November 1983. 3

[TS17] Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th ACM Symposium on Theory of Computing*, STOC 2017, pages 238–251, New York, NY, USA, 2017. ACM. 1, 2, 4, 5, 6, 8, 9

[Var57] R.R. Varshamov. Estimate of the number of signals in error correcting codes. *Doklady Akademii Nauk SSSR*, 117:739–741, 1957. 1

[vL99] Jacobus H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1999. 3