THE UNIVERSITY OF CHICAGO

ADVENTURES IN DECODING DIRECT SUM CODES

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF MATHEMATICS

BY

DYLAN QUINTANA

CHICAGO, ILLINOIS

JUNE 2021

To Mom and Dad

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

The work that went into this thesis would not have been possible without the help of a great deal of people. Thank you to my advisors, Sasha Razborov and Madhur Tulsiani, for sharing your deep technical wisdom and guidance on navigating the academic world; my coauthors, Vedat Levi Alev, Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani (again), for providing the insights and motivation to push our work as far as we did; and the entire Chicago theory group, for being a welcoming community full of interesting ideas.

Thanks also to my friends who made the past six years of grad school breeze right by. Thanking each of you individually would make these acknowledgments rival the length of the rest of this document, so let me instead express my gratitude in bulk to everyone in The Word, Band of Buddies, and Crossword Crew for bringing me joy—you know who you are. Bonus appreciation to Ronno Das, for all of the fun times and absurd conversations, and Karl Schaefer, for being a great mentor as well as a great friend. And of course, I have to thank my brothers, Kevin and Tyler, for the countless hours of time well spent schlorping around.

Getting to where I am today was the result of the dedicated efforts of the teachers I've had over the years. I owe so much to all of you, especially Mrs. Hamilton, who introduced me to the wonders of math at a young age, and Mr. Oestreich, whose teaching style I aspire to match in my own classes. Most of all, thank you to my parents, my first teachers, for your constant love, support, and encouragement from the very beginning, and for the many sacrifices you made for my sake. This thesis is dedicated to you.

ABSTRACT

Our adventures take us through several problems related to decoding a class of error-correcting codes known as direct sum codes. These codes are obtained from a base code $\mathcal{C}_0 \subseteq \mathbb{F}_2^n$ by summing the entries of each codeword on a collection of subsets $W \subseteq [n]^k$ to yield a code $\mathcal{C} \subseteq \mathbb{F}_2^W$. If the collection W has a particular expansion property, the resulting direct sum code $\mathcal{C}$ will have a large minimum distance and (as we show) an efficient decoding algorithm.

In our first adventure (Chapter 2, joint work with Vedat Levi Alev, Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani), we develop a list decoding algorithm for a generalization of direct sum codes where the direct sum can be replaced by any suitable "lifting" operation. The algorithm creates a Sum-of-Squares program for list decoding, the solution to which allows the desired list of codewords to be recovered through rounding. We apply the general decoding framework to obtain list decoding algorithms for direct sum codes constructed using a collection W which is either the set of walks of k vertices on an expander graph or the set of k-sized faces of a high-dimensional expander.

Our second adventure (Chapter 3, joint work with Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani) is adapting the results of Chapter 2 to decode Ta-Shma's [STOC 2017] direct sum codes. These codes are ε-balanced, meaning any two codewords have distance between $1/2 - \varepsilon/2$ and $1/2 + \varepsilon/2$, and have rate nearly achieving the Gilbert–Varshamov bound of $\Omega(\varepsilon^2)$. Decoding is accomplished using the direct sum list decoding framework in an inductive process that sidesteps its suboptimal parameter requirements. This process can perform both unique and list decoding for Ta-Shma's codes, though at a list decoding radius smaller than the minimum distance.

CHAPTER 1

INTRODUCTION

1.1 Codes

Error-correcting codes have proven to be extremely useful in both practical applications and various areas of theoretical computer science. A binary error-correcting code $\mathcal{C}$ is a collection of elements of $\mathbb{F}_2^n$ called codewords. When a codeword is transmitted across a noisy communication channel that introduces some substitution errors, the recipient can hope to recover the original message if the number of errors is small relative to the number of differences between codewords—hence the "error-correcting" moniker.

The key parameters of a code are its rate $R = \frac{1}{n} \log |\mathcal{C}|$, which measures its size, and distance $\delta = \frac{1}{n} \min_{x,y \in \mathcal{C}} |\{i \mid x_i \neq y_i\}|$, which is the minimum fraction of bits that must be flipped to transform one codeword into another. A code of distance δ is theoretically resilient to substitution errors in up to $\delta/2$ fraction of positions before the original message becomes ambiguous. However, coming up with an efficient unique decoding algorithm that can fix this amount of errors in a given code is often a nontrivial task that depends heavily on the code's structure.

The ideal code has both a large rate and a large distance, possessing an ample quantity of codewords with strong error correction capability. There is a natural tension between these two parameters: cramming lots of words into $\mathbb{F}_2^n$ to achieve a high rate comes at the expense of packing them closer together, causing the distance to suffer. Exploring the optimal tradeoff between distance and rate has been a major area of focus in the study of error-correcting codes since its inception. An early lower bound is due to Gilbert [Gil52] and Varshamov [Var57]:

Theorem 1.1.1 (Gilbert–Varshamov (GV) bound for binary codes)**.** *For any* $0 \leq \delta < 1/2$ *and* $0 < \varepsilon \leq 1 - H(\delta)$*, there exist binary codes with distance* δ *and rate at least* $1 - H(\delta) - \varepsilon$*,*

where H is the binary entropy function $H(\delta) = -\delta \log \delta - (1-\delta)\log(1-\delta)$.

In fact, both random codes (where codewords are selected from $\mathbb{F}_2^n$ uniformly at random) and random linear codes (which are random linear subspaces of $\mathbb{F}_2^n$) achieve the GV bound with high probability. Despite what the abundance of codes at the GV bound suggests, finding *explicit* binary codes that attain the GV bound turns out to be quite challenging.

This thesis is primarily concerned with binary codes of distance close to $1/2$. (Binary codes with distance greater than $1/2$ always have rate approaching 0 as $n \to \infty$.) The holy grail in this regime is a family of explicit codes with distance $1/2 - \varepsilon$, the GV bound rate of $\Omega(\varepsilon^2)$, and an accompanying decoding algorithm. Previous results have come up short in at least one of these aspects:

- Thommesen [Tho83] described concatenated codes with rate matching the GV bound that were later decoded by Guruswami and Indyk [GI04], but the construction requires random binary codes, making it non-explicit.
- Guruswami and Rudra [GR06] developed explicit, efficiently decodable codes with rate $\Omega(\varepsilon^3)$, just shy of the GV bound.
- Ta-Shma [TS17] constructed explicit codes with rate $\Omega(\varepsilon^{2+\beta})$ for any fixed $\beta > 0$, but did not provide an efficient decoding algorithm for them.

A more detailed summary of prior work can be found in Table 3.1.

1.2 Direct Sum Codes

Let us take a closer look at Ta-Shma's aforementioned codes near the GV bound [TS17]. These codes have the property of being ε-balanced, meaning each pair of codewords has distance between $1/2 - \varepsilon/2$ and $1/2 + \varepsilon/2$, and are a type of direct sum code, defined using a "lifting" process of a base code $\mathcal{C}_0 \subseteq \mathbb{F}_2^n$ on a collection of tuples $W \subseteq [n]^k$. Each codeword

in a direct sum code is obtained from a codeword in $z \in \mathcal{C}_0$ by summing the entries in each element of W to obtain a new codeword $y \in \mathbb{F}_2^W$; that is, $y = (z_{w_1} + z_{w_2} + \cdots + z_{w_k})_{w \in W}$. With an appropriate choice of W and an ε_0-balanced base code for constant ε_0, the resulting code can be made ε-balanced for arbitrarily small ε. For instance, taking W to be a random subset of $[n]^k$ of size $O(n/\varepsilon^2)$ and k to be sufficiently large will yield an ε-balanced direct sum code, although it would be non-explicit in this case.

Rozenman and Wigderson [Bog12] suggested derandomizing the above construction by taking W to be the set of all walks containing k vertices on an n-vertex expander graph G. This choice of W still gives ε-balanced codes for any $\varepsilon > 0$ as long as the second eigenvalue of G is small enough. However, the rate of a direct sum code depends on the size of W, and Ta-Shma showed that even with an optimal choice of the graph G, the best rate that this construction can attain seems to be $\Omega(\varepsilon^4)$. Ta-Shma improved the rate to $\Omega(\varepsilon^{2+\beta})$ by restricting W to be a more limited subset of walks on an expander. The walks in this set W can be modeled as walks on a graph formed by combining two expanders using the s-wide replacement product, an operation similar to the zig-zag product.

1.3 Decoding Direct Sum Codes

In Chapter 2, we describe a list decoding algorithm for direct sum codes that outputs a short list of all codewords within a given distance of a received word $\tilde{y} \in \mathbb{F}_2^W$. Taking W to be the set of k-vertex walks on an expander, we prove the following:

Theorem 1.3.1 (Informal version of Theorem 2.9.2)**.** *Let $0 < \varepsilon < \varepsilon_0 < 1/2$. Suppose $\mathcal{C}_0 \subseteq \mathbb{F}_2^n$ is an ε_0-balanced linear code and G is a regular expander graph on n vertices with second eigenvalue $\lambda \leq \varepsilon^{O(1)}$. Then there is an ε-balanced direct sum lifting $\mathcal{C}$ of $\mathcal{C}_0$ using the set W of all walks on G of length $k = O(\log(1/\varepsilon))$ with an efficient algorithm that outputs the list of all codewords of $\mathcal{C}$ within distance $(1/2 - \sqrt{\varepsilon})$ of any $\tilde{y} \in \mathbb{F}_2^W$.*

The same framework can be used to obtain list decoding algorithms for direct sum codes

in settings where an expansion requirement called "splittability" is satisfied, such as when W is the set of edges of size k in a high-dimensional expander. There are several ways of defining high-dimensional expanders as expanding hypergraphs; here we use the γ-high-dimensional expander definition of Dinur and Kaufman [DK17].

Theorem 1.3.2 (Informal version of Theorem 2.7.1)**.** *Let $0 < \varepsilon < \varepsilon_0 < 1/2$. Suppose $\mathcal{C}_0 \subseteq \mathbb{F}_2^n$ is an ε_0-balanced linear code and X is a γ-high-dimensional expander on n vertices with $\gamma \leq (\log(1/\varepsilon))^{-O(\log(1/\varepsilon))}$ and dimension $d = \Omega((\log(1/\varepsilon))^2/\varepsilon^2)$. Then there is an ε-balanced direct sum lifting $\mathcal{C}$ of $\mathcal{C}_0$ using the set W of all edges of X of size $k = O(\log(1/\varepsilon))$ with an efficient algorithm that outputs the list of all codewords of $\mathcal{C}$ within distance $(1/2-\sqrt{\varepsilon})$ of any $\tilde{y} \in \mathbb{F}_2^W$.*

As a corollary of Theorem 1.3.2, we also have a list decoding algorithm for direct product codes on high-dimensional expanders, where the bits of codewords in the base code are collected into tuples on each edge rather than added together. List decoding for these codes works at distances approaching one thanks to their large alphabet size.

The list decoding framework operates by treating the problem of decoding a received word $\tilde{y}$ as a constraint satisfaction problem (CSP). The variables in the CSP correspond to positions in the base code, and each element of W poses one constraint. The objective is to find all elements $z \in \mathcal{C}_0$ that satisfy at least $(1/2 + \eta)$ fraction of the constraints. Fortunately, the expansion of the direct sum lifting structure guarantees that any element $z' \in \mathbb{F}_2^n$ satisfying many constraints will be close to some codeword $z \in \mathcal{C}_0$, so it suffices to solve the CSP ignoring the base code requirement to get a solution z', then run the unique decoding algorithm of the base code to recover z. Even more fortunately, the algorithm of Alev, Jeronimo, and Tulsiani [AJT19] based on the Sum-of-Squares (SOS) hierarchy enables us to approximately solve the CSP for z' when W is splittable.

This procedure is capable of finding at least one codeword close to $\tilde{y}$, but list decoding introduces the additional wrinkle of needing to find every such codeword. Generating the

entire list requires changes to the SOS program. Solutions to the SOS program are converted to elements of $\mathbb{F}_2^n$ by a rounding process that involves conditioning the local distributions on a small number of fixed positions. For list decoding, we need to ensure that every $z \in \mathcal{C}_0$ satisfying the appropriate number of constraints can be approximately reached by rounding after some choice of conditioning. This is done by having the SOS program minimize a function measuring how concentrated the solution is. The solutions to the modified SOS program are conditioned and rounded to get a list containing a word close to every desired codeword in the base code. Finally, the list of codewords close to $\tilde{y}$ is extracted with the help of the unique decoding algorithm of $\mathcal{C}_0$.

1.4 Decoding Direct Sum Codes Near the GV Bound

The decoding algorithms of Theorem 1.3.1 and Theorem 1.3.2 have stringent expansion requirements that cause the rates of the codes to end up quite far from the GV bound. Chapter 3 is about employing a few tricks that let us use the list decoding algorithm to decode Ta-Shma's direct sum codes of rate $\Omega(\varepsilon^{2+\beta})$ [TS17]. The main result is as follows.

Theorem 1.4.1 (Informal version of Theorem 3.1.1)**.** *For any $\beta > 0$, sufficiently small $\varepsilon > 0$, and infinitely many $N \in \mathbb{N}$, there are explicit ε-balanced binary codes $\mathcal{C} \subseteq \mathbb{F}_2^N$ with rate $\Omega(\varepsilon^{2+\beta})$ and a unique decoding algorithm that runs in time polynomial in N.*

Digging a bit deeper into the parameters of Theorem 1.3.1, we find that list decoding an ε-balanced code at a distance of $(1/2 - \sqrt{\varepsilon} - \eta)$ requires a second eigenvalue of $\lambda \leq (\eta/2^k)^{O(1)}$ and a walk length of $k = O(\log(1/\varepsilon))$. Increasing the rate of the code from quasipolynomial in ε to polynomial in ε can be done if we can get away with λ being a constant independent of ε.

As a first step toward weakening the requirement on λ, observe that if we only want a unique decoding algorithm rather than a list decoding algorithm, it only needs to work at

half the distance of the code, which is less than $1/4$. We can thus fix η to be any constant strictly smaller than $1/4$. For unique decoding of a word $\tilde{y}$, we run the list decoding algorithm at distance $1/2 - \sqrt{\varepsilon} - \eta$ and simply pick out the word on the list closest to $\tilde{y}$.

Removing the dependence of λ on the walk length k is more involved. We break the problem of decoding the direct sum code $\mathcal{C}$ into a sequence of easier decoding problems. Let $k = m^\ell$ for integers m and ℓ. Define the code $\mathcal{C}_1$ to be the direct sum lifting of the base code $\mathcal{C}_0$ using m-vertex walks on our graph G. Form a new graph G' whose vertices are m-vertex walks of G, with a directed edge from w to w' if the last vertex of w is adjacent to the first vertex of w' in G. Then take $\mathcal{C}_2$ to be the direct sum lifting of $\mathcal{C}_1$ using m-vertex walks on G', which is equivalent to the direct sum lifting of $\mathcal{C}_0$ using walks of m^2 vertices on G. We continue in this fashion, defining $\mathcal{C}_i$ to be the direct sum lifting of $\mathcal{C}_{i-1}$ using m-vertex walks on a graph of walks on the previous level, until we reach $\mathcal{C}_\ell = \mathcal{C}$.

Creating such a "cascade" of codes $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_\ell$ allows decoding to be accomplished recursively. We decode $\mathcal{C}_i$ to get a list of words at the level of $\mathcal{C}_{i-1}$, which can in turn be decoded to get a list of words at the level of $\mathcal{C}_{i-2}$, and so on, until we get down to $\mathcal{C}_0$ where we can apply the known unique decoding algorithm. The advantage of this process is that we can choose m to be a constant, hiding away the dependence on ε in the number of levels ℓ and removing it from the condition on λ. Decoding Ta-Shma's more complicated construction in this setup is a matter of altering it slightly to meet the splittability requirement of the direct sum list decoding framework and recomputing parameters to make sure the rate is still $\Omega(\varepsilon^{2+\beta})$.

CHAPTER 2

LIST DECODING OF DIRECT SUM CODES

This chapter is joint work with Vedat Levi Alev, Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani that originally appeared in [AJQ+20].

2.1 Introduction

We consider the problem of list decoding binary codes obtained by starting with a binary base code $\mathcal{C}$ and amplifying its distance by "lifting" $\mathcal{C}$ to a new code $\mathcal{C}'$ using an expanding or pseudorandom structure. Examples of such constructions include *direct products* where one lifts (say) $\mathcal{C} \subseteq \mathbb{F}_2^n$ to $\mathcal{C}' \subseteq (\mathbb{F}_2^k)^{n^k}$ with each position in $y \in \mathcal{C}'$ being a k-tuple of bits from k positions in $z \in \mathcal{C}$. Another example is *direct sum* codes where $\mathcal{C}' \subseteq \mathbb{F}_2^{n^k}$ and each position in y is the parity of a k-tuple of bits in $z \in \mathcal{C}$. Of course, for many applications, it is interesting to consider a small "pseudorandom" set of k-tuples, instead of considering the complete set of size n^k.

This kind of distance amplification is well known in coding theory [ABN+92, IW97, GI01, TS17] and it can draw on the vast repertoire of random and pseudorandom expanding objects [HLW06, Lub18]. Such constructions are also known to have several applications to the theory of Probabilitically Checkable Proofs (PCPs) [IKW09, DS14, DDG+15, Cha16, Aro02]. However, despite having several useful properties, it might not always be clear how to *decode* the codes resulting from such constructions, especially when constructed using sparse pseudorandom structures. An important example of this phenomenon is Ta-Shma's explicit construction of binary codes of arbitrarily large distance near the (non-constructive) Gilbert-Varshamov bound [TS17]. Although the construction is explicit, efficient decoding is not known. Going beyond unique-decoding algorithms, it is also useful to have efficient list-decoding algorithms for complexity-theoretic applications [Sud00, Gur01, STV01, Tre04].

The question of list decoding such pseudorandom constructions of direct-product codes was considered by Dinur et al. [DHK$^+$19], extending a unique-decoding result of Alon et al. [ABN$^+$92]. While Alon et al. proved that the code is unique-decodable when the lifting hypergraph (collection of k-tuples) is a good "sampler", Dinur et al. showed that when the hypergraph has additional structure (which they called being a "double sampler") then the code is also list decodable. They also posed the question of understanding structural properties of the hypergraph that might yield even unique decoding algorithms for the *direct sum* based liftings.

We develop a generic framework to understand properties of the hypergraphs under which the lifted code $\mathcal{C}'$ admits efficient list decoding algorithms, assuming only efficient unique decoding algorithms for the base code $\mathcal{C}$. Formally, let X be a downward-closed hypergraph (simplicial complex) defined by taking the downward closure of a k-uniform hypergraph, and let $g : \mathbb{F}_2^k \to \mathbb{F}_2$ be any boolean function. $X(i)$ denotes the collection of sets of size i in X and $X(\leq d)$ the collection of sets of size at most d. We consider the lift $\mathcal{C}' = \mathrm{dsum}^g_{X(k)}(\mathcal{C})$, where $\mathcal{C} \subseteq \mathbb{F}_2^{X(1)}$ and $\mathcal{C}' \subseteq \mathbb{F}_2^{X(k)}$, and each bit of $y \in \mathcal{C}'$ is obtained by applying the function g to the corresponding k bits of $z \in \mathcal{C}$. We study properties of g and X under which this lifting admits an efficient list decoding algorithm.

We consider two properties of this lifting, *robustness* and *tensoriality*, formally defined later, which are sufficient to yield decoding algorithms. The first property (robustness) essentially requires that for any two words in $\mathbb{F}_2^{X(1)}$ at a moderate distance, the lifting amplifies the distance between them. While the second property is of a more technical nature and is inspired by the Sum-of-Squares (SOS) SDP hierarchy used for our decoding algorithms, it is implied by some simpler combinatorial properties. Roughly speaking, this combinatorial property, which we refer to as *splittability*, requires that the graph on (say) $X(k/2)$ defined by connecting $\mathfrak{s}, \mathfrak{t} \in X(k/2)$ if $\mathfrak{s} \cap \mathfrak{t} = \emptyset$ and $\mathfrak{s} \cup \mathfrak{t} \in X(k)$, is a sufficiently good expander (and similarly for graphs on $X(k/4)$, $X(k/8)$, and so on). Splittability requires

that the k-tuples can be (recursively) split into disjoint pieces such that at each step the graph obtained between the pairs of pieces is a good expander.

Expanding Structures. We instantiate the above framework with two specific structures: the collection of k-sized hyperedges of a high-dimensional expander (HDX) and the collection of length k walks on an expander graph.[1] HDXs are downward-closed hypergraphs satisfying certain expansion properties. We will quantify this expansion using Dinur and Kaufman's notion of a γ-HDX [DK17].

HDXs were proved to be splittable by some of the authors [AJT19]. For the expander walk instantiation, we consider a variant of splittability where a walk of length k is split into two halves, which are walks of length $k/2$ (thus we do *not* consider all $k/2$ size subsets of the walk). The spectrum of the graphs obtained by this splitting can easily be related to that of the underlying expander graph. In both cases, we take the function g to be k-XOR which corresponds to the direct sum lifting. We also obtain results for direct product codes via a simple (and standard) reduction to the direct sum case.

Our Results. Now we provide a quantitative version of our main result. For this, we split the main result into two cases (due to their difference in parameters): HDXs and length k walks on expander graphs. We start with the former expanding object.

Theorem 2.1.1 (Direct Sum Lifting on HDX (Informal))**.** *Let $\varepsilon_0 < 1/2$ be a constant and $\varepsilon \in (0, \varepsilon_0)$. Suppose $X(\leq d)$ is a γ-HDX on n vertices with $\gamma \leq (\log(1/\varepsilon))^{-O(\log(1/\varepsilon))}$ and $d = \Omega\left((\log(1/\varepsilon))^2/\varepsilon^2\right)$.*

For every linear code $\mathcal{C}_1 \subset \mathbb{F}_2^n$ with relative distance $\geq 1/2 - \varepsilon_0$, there exists a direct sum lifting $\mathcal{C}_k \subset \mathbb{F}_2^{X(k)}$ with $k = O(\log(1/\varepsilon))$ and relative distance $\geq 1/2 - \varepsilon^{\Omega_{\varepsilon_0}(1)}$ satisfying the following:

- *[Efficient List Decoding] If $\tilde{y}$ is $(1/2 - \varepsilon)$-close to $\mathcal{C}_k$, then we can compute the list of*

1. Actually, we will be working with length $k-1$ walks which can be represented as k-tuples, though this is an unimportant technicality. The reason is to be consistent in the number of vertices (allowing repetitions) with k-sized hyperedges.

all the codewords of $\mathcal{C}_k$ *that are* $(1/2-\varepsilon)$*-close to* $\tilde{y}$ *in time* $n^{\varepsilon^{-O(1)}} \cdot f(n)$*, where* $f(n)$ *is the running time of a unique decoding algorithm for* $\mathcal{C}_1$*.*

- *[Rate] The rate* r_k *of* $\mathcal{C}_k$ *is* $r_k = r_1 \cdot |X(1)| \,/\, |X(k)|$*, where* r_1 *is the rate of* $\mathcal{C}_1$*.*[2]

A consequence of this result is a method of decoding the direct product lifting on a HDX via a reduction to the direct sum case.

Corollary 2.1.2 (Direct Product Lifting on HDX (Informal))**.** *Let* $\varepsilon_0 < 1/2$ *be a constant and* $\varepsilon > 0$*. Suppose* $X(\leq d)$ *is a* γ*-HDX on* n *vertices with* $\gamma \leq (\log(1/\varepsilon))^{-O(\log(1/\varepsilon))}$ *and* $d = \Omega((\log(1/\varepsilon))^2/\varepsilon^2)$*.*

For every linear code $\mathcal{C}_1 \subset \mathbb{F}_2^n$ *with relative distance* $\geq 1/2 - \varepsilon_0$*, there exists a direct product encoding* $\mathcal{C}_\ell \subset (\mathbb{F}_2^\ell)^{X(\ell)}$ *with* $\ell = O(\log(1/\varepsilon))$ *that can be efficiently list decoded up to distance* $(1-\varepsilon)$*.*

Remark 2.1.3. *List decoding the direct product lifting was first established by Dinur et al. in [DHK*$^+$*19] using their notion of double samplers. Since constructions of double samplers are only known using HDXs, we can compare some parameters. In our setting, we obtain* $d = O(\log(1/\varepsilon)^2/\varepsilon^2)$ *and* $\gamma = (\log(1/\varepsilon))^{-O(\log(1/\varepsilon))}$ *whereas in [DHK*$^+$*19]* $d = O(\exp(1/\varepsilon))$ *and* $\gamma = O(\exp(-1/\varepsilon))$*.*

Given a graph G, we denote by $W_G(k)$ the collection of all length $k-1$ walks of G, which plays the role of the local views $X(k)$. If G is sufficiently expanding, we have the following result.

Theorem 2.1.4 (Direct Sum Lifting on Expander Walks (Informal))**.** *Let* $\varepsilon_0 < 1/2$ *be a constant and* $\varepsilon \in (0, \varepsilon_0)$*. Suppose* G *is a* d*-regular* γ*-two-sided spectral expander graph on* n *vertices with* $\gamma \leq \varepsilon^{O(1)}$*.*

2. In the rate computation, $X(k)$ is viewed as a multi-set where each $\mathfrak{s} \in X(k)$ is repeated a certain number of times for technical reasons.

For every linear code $\mathcal{C}_1 \subset \mathbb{F}_2^n$ with relative distance $\geq 1/2 - \varepsilon_0$, there exists a direct sum encoding $\mathcal{C}_k \subset \mathbb{F}_2^{W_G(k)}$ with $k = O(\log(1/\varepsilon))$ and relative distance $\geq 1/2 - \varepsilon^{\Omega_{\varepsilon_0}(1)}$ satisfying the following:

- *[Efficient List Decoding] If $\tilde{y}$ is $(1/2-\varepsilon)$-close to $\mathcal{C}_k$, then we can compute the list of all the codewords of $\mathcal{C}_k$ that are $(1/2-\varepsilon)$-close to $\tilde{y}$ in time $n^{\varepsilon^{-O(1)}} \cdot f(n)$, where $f(n)$ is the running time of a unique decoding algorithm for $\mathcal{C}_1$.*
- *[Rate] The rate r_k of $\mathcal{C}_k$ is $r_k = r_1/d^{k-1}$, where r_1 is the rate of $\mathcal{C}_1$.*

The results in Theorem 2.1.1, Corollary 2.1.2, and Theorem 2.1.4 can all be extended (using a simple technical argument) to nonlinear base codes $\mathcal{C}_1$ with similar parameters. We also note that applying Theorem 2.1.1 to explicit objects derived from Ramanujan complexes [LSV05b, LSV05a] and applying Theorem 2.1.4 to Ramanujan graphs [LPS88] yield explicit constructions of codes with constant relative distance and rate, starting from a base code with constant relative distance and rate. With these constructions, the rate of the lifted code satisfies $r_k \geq r_1 \cdot \exp\left(-(\log(1/\varepsilon))^{O(\log(1/\varepsilon))}\right)$ in the HDX case and $r_k \geq r_1 \cdot \varepsilon^{O(\log(1/\varepsilon))}$ for expander walks. The precise parameters of these applications are given in Corollary 2.7.2 of Section 2.7 and in Corollary 2.9.5 of Section 2.9, respectively.

Our techniques. We connect the question of decoding lifted codes to finding good solutions for instances of Constraint Satisfaction Problems (CSPs) which we then solve using the Sum-of-Squares (SOS) hierarchy. Consider the case of direct sum lifting, where for the lifting y of a codeword z, each bit of y is an XOR of k bits from z. If an adversary corrupts some bits of y to give $\tilde{y}$, then finding the closest codeword to $\tilde{y}$ corresponds to finding $z' \in \mathcal{C}$ such that appropriate k-bit XORs of z' agree with as many bits of $\tilde{y}$ as possible. If the corruption is small, the distance properties of the code ensure that the unique choice for z' is z. Moreover, the distance amplification (robustness) properties of the lifting can be used to show that it suffices to find *any* z' (not necessarily in $\mathcal{C}$) satisfying sufficiently many constraints. We then use results by a subset of the authors [AJT19] showing that splittability

(or the tensorial nature) of the hypergraphs used for lifting can be used to yield algorithms for approximately solving the related CSPs. Of course, the above argument does not rely on the lifting being direct sum and works for any lifting function g.

For list decoding, we solve just a single SOS program whose solution is rich enough to "cover" the list of codewords we intend to retrieve. In particular, the solutions to the CSP are obtained by "conditioning" the SDP solution on a small number of variables, and we try to ensure that in the list decoding case, conditioning the SOS solution on different variables yields solutions close to different elements of the list. To achieve this covering property we consider a convex proxy Ψ for negative entropy measuring how concentrated (on a few codewords) the SOS solution is. Then we minimize Ψ while solving the SOS program. A similar technique was also independently used by Karmalkar, Klivans, and Kothari [KKK19] and Raghavendra–Yau [RY19] in the context of learning regression. Unfortunately, this SOS cover comes with only some weak guarantees which are, a priori, not sufficient for list decoding. However, again using the robustness property of the lifting, we are able to convert weak covering guarantees for the lifted code $\mathcal{C}'$ to strong guarantees for the base code $\mathcal{C}$, and then appeal to the unique decoding algorithm. We regard the interplay between these two properties leading to the final list decoding application as our main technical contribution. A more thorough overview is given in Section 2.3 after introducing some objects and notation in Section 2.2. In Section 2.3, we also give further details about the organization of the document.

Related work. The closest result to ours is the list decoding framework of Dinur et al. [DHK$^+$19] for the direct product encoding, where the lifted code is not binary but rather over the alphabet $\mathbb{F}_2^k$. Our framework instantiated for the direct sum encoding on HDXs (c.f. Theorem 2.1.1) captures and strengthens some of their parameters in Corollary 2.1.2. While Dinur et al. also obtain list decoding by solving an SDP for a specific CSP (Unique Games), the reduction to CSPs in their case uses the combinatorial nature of the double

sampler instances and is also specific to the direct product encoding. They recover the list by iteratively solving many CSP instances, where each newly found solution is pruned from the instance by reducing the alphabet size by one each time. On the other hand, the reduction to CSPs is somewhat generic in our framework and the recovery of the list is facilitated by including an entropic proxy in the convex relation. As mentioned earlier, a similar entropic proxy was also (independently) used by Karmalkar et al. [KKK19] and Raghavendra–Yau [RY19] in the context of list decoding for linear regression and mean estimation. Direct products on expanders were also used as a building block by Guruswami and Indyk [GI03] who used these to construct *linear time* list decodable codes over large alphabets. They gave an algorithm for recovering the list based on spectral partitioning techniques.

2.2 Preliminaries

2.2.1 Simplicial Complexes

It will be convenient to work with hypergraphs satisfying a certain downward-closed property (which is straightforward to obtain).

Definition 2.2.1. *A* simplicial complex X *with ground set* $[n]$ *is a downward-closed collection of subsets of* $[n]$*, i.e., for all sets* $\mathfrak{s} \in X$ *and* $\mathfrak{t} \subseteq \mathfrak{s}$*, we also have* $\mathfrak{t} \in X$*. The sets in* X *are referred to as* faces *of* X*. We use the notation* $X(i)$ *for the set of all faces of a simplicial complex* X *with cardinality* i *and* $X(\leq d)$ *for the set of all faces of cardinality at most* d*.*[3] *By convention, we take* $X(0) := \{\varnothing\}$*.*

A simplicial complex $X(\leq d)$ *is said to be a* pure simplicial complex *if every face of* X

3. Note that it is more common to associate a geometric representation to simplicial complexes, with faces of cardinality i being referred to as faces of *dimension* $i-1$ (and the collection being denoted by $X(i-1)$ instead of $X(i)$). However, we prefer to index faces by their cardinality to improve readability of related expressions.

is contained in some face of size d. Note that in a pure simplicial complex $X(\leq d)$, the top slice $X(d)$ completely determines the complex.

Simplicial complexes are equipped with the following probability measures on their sets of faces.

Definition 2.2.2 (Probability measures $(\Pi_1, \ldots, \Pi_d)$)**.** *Let $X(\leq d)$ be a pure simplicial complex and let Π_d be an arbitrary probability measure on $X(d)$. We define a coupled array of random variables $(\mathfrak{s}^{(d)}, \ldots, \mathfrak{s}^{(1)})$ as follows: sample $\mathfrak{s}^{(d)} \sim \Pi_d$ and (recursively) for each $i \in [d]$, take $\mathfrak{s}^{(i-1)}$ to be a uniformly random subset of $\mathfrak{s}^{(i)}$ of size $i-1$. The distributions $\Pi_{d-1}, \ldots, \Pi_1$ are then defined to be the marginal distributions of the random variables $\mathfrak{s}^{(d-1)}, \ldots, \mathfrak{s}^{(1)}$. We also define the joint distribution of $(\mathfrak{s}^{(d)}, \ldots, \mathfrak{s}^{(1)})$ as Π. Note that the choice of Π_d determines each other distribution Π_i on $X(i)$.*

In order to work with the HDX and expander walk instantiations in a unified manner, we will use also use the notation $X(k)$ to indicate the set of all length $k-1$ walks on a graph G. In this case, $X(k)$ is a set of k-tuples rather than subsets of size k. This distinction will be largely irrelevant, but we will use $W_G(k)$ when referring specifically to walks rather than subsets. The set of walks $W_G(k)$ has a corresponding distribution Π_k as well (see Definition 2.9.1).

2.2.2 Codes and Lifts

Codes

We briefly recall some standard code terminology. Let Σ be a finite alphabet with $q \in \mathbb{N}$ symbols. We will be mostly concerned with the case $\Sigma = \mathbb{F}_2$. Given $z, z' \in \Sigma^n$, recall that the relative Hamming distance between z and z' is $\Delta(z, z') \coloneqq \left|\{i \mid z_i \neq z'_i\}\right| / n$. Any set $\mathcal{C} \subset \Sigma^n$ gives rise to a q-ary code. The distance of $\mathcal{C}$ is defined as $\Delta(\mathcal{C}) \coloneqq \min_{z \neq z'} \Delta(z, z')$

where $z, z' \in \mathcal{C}$. We say that $\mathcal{C}$ is a linear code if $\Sigma = \mathbb{F}_q$ and $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^n$.[4] The rate of $\mathcal{C}$ is $\log_q(|\mathcal{C}|)/n$.

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

Definition 2.2.3 (Bias)**.** *The* bias *of a word $z \in \mathbb{F}_2^n$ is* $\mathrm{bias}(z) \coloneqq \left|\mathbb{E}_{i\in[n]}(-1)^{z_i}\right|$.[5] *The bias of a code $\mathcal{C}$ is the maximum bias of any non-zero codeword in $\mathcal{C}$.*

Lifts

Starting from a code $\mathcal{C}_1 \subset \Sigma_1^{X(1)}$, we amplify its distance by considering a *lifting* operation defined as follows.

Definition 2.2.4 (Lifting Function)**.** *Let $g : \Sigma_1^k \to \Sigma_k$ and $X(k)$ be a collection of k-uniform hyperedges or walks of length $k-1$ on the set $X(1)$. For $z \in \Sigma_1^{X(1)}$, we define $\mathrm{dsum}^g_{X(k)}(z) = y$ such that $y_{\mathfrak{s}} = g(z|_{\mathfrak{s}})$ for all $\mathfrak{s} \in X(k)$, where $z|_{\mathfrak{s}}$ is the restriction of z to the indices in $\mathfrak{s}$.*

The lifting of a code $\mathcal{C}_1 \subseteq \Sigma_1^{X(1)}$ is

$$\mathrm{dsum}^g_{X(k)}(\mathcal{C}_1) = \{\mathrm{dsum}^g_{X(k)}(z) \mid z \in \mathcal{C}_1\},$$

which we will also denote $\mathcal{C}_k$. We will omit g and $X(k)$ from the notation for lifts when they are clear from context.

We will call liftings that amplify the distance of a code *robust*.

Definition 2.2.5 (Robust Lifting)**.** *We say that $\mathrm{dsum}^g_{X(k)}$ is (δ_0, δ)-robust if for every $z, z' \in \Sigma_1^{X(1)}$ we have*

$$\Delta(z, z') \geq \delta_0 \Rightarrow \Delta(\mathrm{dsum}(z), \mathrm{dsum}(z')) \geq \delta.$$

4. In this case, q is required to be a prime power.

5. Equivalently, the bias of $z \in \{\pm 1\}^n$ is $\mathrm{bias}(z) \coloneqq \left|\mathbb{E}_{i\in[n]} z_i\right|$.

For us the most important example of lifting is when the function g is k-XOR and $\Sigma_1 = \Sigma_k = \mathbb{F}_2$, which has been extensively studied in connection with codes and otherwise [TS17, STV01, GNW95, ABN+92]. In our language of liftings, k-XOR corresponds to the *direct sum lifting*.

Definition 2.2.6 (Direct Sum Lifting)**.** *Let $\mathcal{C}_1 \subseteq \mathbb{F}_2^n$ be a base code on $X(1) = [n]$. The* direct sum lifting *of a word $z \in \mathbb{F}_2^n$ on a collection $X(k)$ is $\mathrm{dsum}_{X(k)}(z) = y$ such that $y_{\mathfrak{s}} = \sum_{i \in \mathfrak{s}} z_i$ for all $\mathfrak{s} \in X(k)$.*

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 2.2.7 (Parity Sampler)**.** *Let $g\colon \mathbb{F}_2^k \to \mathbb{F}_2$. We say that $\mathrm{dsum}^g_{X(k)}$ is an (β_0, β)-parity sampler if for all $z \in \mathbb{F}_2^{X(1)}$ with $\mathrm{bias}(z) \le \beta_0$, we have $\mathrm{bias}(\mathrm{dsum}(z)) \le \beta$.*

2.2.3 Constraint Satisfaction Problems (CSPs)

A k-CSP instance $\mathfrak{J}(H, \mathcal{P}, w)$ with alphabet size q consists of a k-uniform hypergraph H, a set of constraints

$$\mathcal{P} = \Big\{ \mathcal{P}_{\mathfrak{a}} \subseteq [q]^{\mathfrak{a}} : \mathfrak{a} \in H \Big\},$$

and a non-negative weight function $w \in \mathbb{R}_+^H$ on the constraints satisfying $\sum_{\mathfrak{a} \in H} w(a) = 1$.

We will think of the constraints as predicates that are satisfied by an assignment σ if we have $\sigma|_{\mathfrak{a}} \in \mathcal{P}_{\mathfrak{a}}$, i.e., the restriction of σ on $\mathfrak{a}$ is contained in $\mathcal{P}_{\mathfrak{a}}$. We write $\mathsf{SAT}_{\mathfrak{J}}(\sigma)$ for the (weighted) fraction of the constraints satisfied by the assignment σ, i.e.,

$$\mathsf{SAT}_{\mathfrak{J}}(\sigma) \;=\; \sum_{\mathfrak{a} \in H} w(\mathfrak{a}) \cdot \mathbf{1}[\sigma|_{\mathfrak{a}} \in \mathcal{P}_{\mathfrak{a}}] \;=\; \mathop{\mathbb{E}}_{\mathfrak{a} \sim w} [\mathbf{1}[\sigma|_{\mathfrak{a}} \in \mathcal{P}_{\mathfrak{a}}]] \ .$$

We denote by $\mathsf{OPT}(\mathfrak{J})$ the maximum of $\mathsf{SAT}_{\mathfrak{J}}(\sigma)$ over all $\sigma \in [q]^{V(H)}$.

A particularly important class of k-CSPs for our work will be k-XOR: here the input consists of a k-uniform hypergraph H with weighting w, and a (right-hand side) vector $r \in \mathbb{F}_2^H$. The constraint for each $\mathfrak{a} \in H$ requires

$$\sum_{i \in \mathfrak{a}} \sigma(i) = r_{\mathfrak{a}} \pmod 2.$$

In this case we will use the notation $\mathfrak{I}(H, r, w)$ to refer to the k-XOR instance. When the weighting w is implicitly clear, we will omit it and just write $\mathfrak{I}(H, r)$.

Any k-uniform hypergraph H can be associated with a pure simplicial complex in a canonical way by setting $X_{\mathfrak{J}} = \{\mathfrak{b} : \exists\ \mathfrak{a} \in H \text{ with } \mathfrak{a} \supseteq \mathfrak{b}\}$; notice that $X_{\mathfrak{J}}(k) = H$. We will refer to this complex as the *constraint complex* of the instance $\mathfrak{I}$. The probability distribution Π_k on $X_{\mathfrak{J}}(k)$ will be derived from the weight function w of the constraint:

$$\Pi_k(\mathfrak{a}) = w(\mathfrak{a}) \quad \forall \mathfrak{a} \in X_{\mathfrak{J}}(k) = H.$$

2.2.4 Sum-of-Squares Relaxations and t-local PSD Ensembles

The Sum-of-Squares (SOS) hierarchy gives a sequence of increasingly tight semidefinite programming relaxations for several optimization problems, including CSPs. Since we will use relatively few facts about the SOS hierarchy, already developed in the analysis of Barak, Raghavendra, and Steurer [BRS11], we will adapt their notation of *t-local distributions* to describe the relaxations. For a k-CSP instance $\mathfrak{I} = (H, \mathcal{P}, w)$ on n variables, we consider the following semidefinite relaxation given by t-levels of the SOS hierarchy, with vectors $v_{(S,\alpha)}$ for all $S \subseteq [n]$ with $|S| \leq t$, and all $\alpha \in [q]^S$. Here, for $\alpha_1 \in [q]^{S_1}$ and $\alpha_2 \in [q]^{S_2}$, $\alpha_1 \circ \alpha_2 \in [q]^{S_1 \cup S_2}$ denotes the partial assignment obtained by concatenating α_1 and α_2.

For any set S with $|S| \leq t$, the vectors $v_{(S,\alpha)}$ induce a probability distribution μ_S over $[q]^S$ such that the assignment $\alpha \in [q]^S$ appears with probability $\|v_{(S,\alpha)}\|^2$. Moreover, these

$$
\begin{aligned}
\text{maximize} \quad & \mathop{\mathbb{E}}_{\mathfrak{a}\sim w}\left[\sum_{\alpha\in\mathcal{P}_{\mathfrak{a}}}\|v_{(\mathfrak{a},\alpha)}\|^2\right] &&=: \ \mathsf{SDP}(\mathfrak{I}) \\
\text{subject to} \quad & \left\langle v_{(S_1,\alpha_1)}, v_{(S_2,\alpha_2)}\right\rangle &&= \ 0 && \forall\ \alpha_1|_{S_1\cap S_2} \neq \alpha_2|_{S_1\cap S_2} \\
& \left\langle v_{(S_1,\alpha_1)}, v_{(S_2,\alpha_2)}\right\rangle &&= \ \left\langle v_{(S_3,\alpha_3)}, v_{(S_4,\alpha_4)}\right\rangle && \forall\ S_1\cup S_2 = S_3\cup S_4,\ \alpha_1\circ\alpha_2 = \alpha_3\circ\alpha_4 \\
& \sum_{j\in[q]}\|v_{(\{i\},j)}\|^2 &&= \ 1 && \forall i\in[n] \\
& \|v_{(\emptyset,\emptyset)}\|^2 &&= \ 1
\end{aligned}
$$

distributions are consistent on intersections: for $T \subseteq S \subseteq [n]$, we have $\mu_{S|T} = \mu_T$, where $\mu_{S|T}$ denotes the restriction of the distribution μ_S to the set T. We use these distributions to define a collection of random variables $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$ taking values in $[q]$, such that for any set S with $|S| \leq t$, the collection of variables $\{\mathbf{Z}_i\}_{i\in S}$ has a joint distribution μ_S. Note that the entire collection $(\mathbf{Z}_1, \ldots, \mathbf{Z}_n)$ *may not* have a joint distribution: this property is only true for sub-collections of size t. We will refer to the collection $(\mathbf{Z}_1, \ldots, \mathbf{Z}_n)$ as a *t-local ensemble* of random variables.

We also have that that for any $T \subseteq [n]$ with $|T| \leq t-2$, and any $\xi \in [q]^T$, we can define a $(t-|T|)$-local ensemble $(\mathbf{Z}'_1, \ldots, \mathbf{Z}'_n)$ by "conditioning" the local distributions on the event $\mathbf{Z}_T = \xi$, where $\mathbf{Z}_T$ is shorthand for the collection $\{\mathbf{Z}_i\}_{i\in T}$. For any S with $|S| \leq t-|T|$, we define the distribution of $\mathbf{Z}'_S$ as $\mu'_S := \mu_{S\cup T}|\{\mathbf{Z}_T = \xi\}$. Finally, the semidefinite program also ensures that for any such conditioning, the conditional covariance matrix

$$
\mathsf{M}_{(S_1,\alpha_1)(S_2,\alpha_2)} \ = \ \mathrm{Cov}\left(\mathbf{1}[\mathbf{Z}'_{S_1} = \alpha_1], \mathbf{1}[\mathbf{Z}'_{S_2} = \alpha_2]\right)
$$

is positive semidefinite, where $|S_1|, |S_2| \leq (t-|T|)/2$. Here, for each pair S_1, S_2 the covariance is computed using the joint distribution $\mu'_{S_1\cup S_2}$. In this paper, we will only consider t-local ensembles such that for every conditioning on a set of size at most $t-2$, the conditional covariance matrix is PSD. We will refer to these as *t-local PSD ensembles*. We will also need a simple corollary of the above definitions.

Fact 2.2.8. *Let $(\mathbf{Z}_1, \ldots, \mathbf{Z}_n)$ be a t-local PSD ensemble, and let X be any collection with $X(1) = [n]$. Then, for all $s \leq t/2$, the collection $\{\mathbf{Z}_{\mathfrak{a}}\}_{\mathfrak{a} \in X(\leq s)}$ is a (t/s)-local PSD ensemble, where $X(\leq s) = \bigcup_{i=1}^{s} X(i)$.*

For random variables $\mathbf{Z}_S$ in a t-local PSD ensemble, we use the notation $\{\mathbf{Z}_S\}$ to denote the distribution of $\mathbf{Z}_S$ (which exists when $|S| \leq t$). We also define $\mathrm{Var}[\mathbf{Z}_S]$ as

$$\mathrm{Var}\left[\mathbf{Z}_S\right] \coloneqq \sum_{\alpha \in [q]^S} \mathrm{Var}\left[\mathbf{1}\left[\mathbf{Z}_S = \alpha\right]\right].$$

Pseudo-expectation Formulation

An equivalent way of expressing this local PSD ensemble is through the use of a pseudo-expectation operator, which is also a language commonly used in the SOS literature (e.g., [BHK$^+$16, BKS17]). The exposition of some of our results is cleaner in this equivalent language. Each variable $\mathbf{Z}_i$ with $i \in [n]$ is modeled by a collection of indicator local random variables $\{\mathbf{Z}_{i,a}\}_{a \in [q]}$ with the intent that $\mathbf{Z}_{i,a} = 1$ iff $\mathbf{Z}_i = a$.[6] To ensure they behave similarly to indicators we add the following restrictions to the SOS formulation:

$$\begin{aligned} \mathbf{Z}_{i,a}^2 &= \mathbf{Z}_{i,a} && \forall i \in [n], a \in [q] \\ \sum_{a \in [q]} \mathbf{Z}_{i,a} &= 1 && \forall i \in [n] \end{aligned}$$

Let $\mathcal{R} = \mathbb{R}[\mathbf{Z}_{1,1}, \ldots, \mathbf{Z}_{n,q}]$ be the ring of polynomials on $\{\mathbf{Z}_{i,a}\}_{i \in [n], a \in [q]}$. We will write $\mathcal{R}^{\leq d}$ for the restriction of $\mathcal{R}$ to polynomials of degree at most d. A feasible solution at the $(2t)$-th level of the SOS hierarchy is a linear operator $\widetilde{\mathbb{E}} : \mathcal{R}^{\leq 2t} \to \mathbb{R}$ called the pseudo-expectation operator. This operator satisfies the following problem-independent constraints: (i) $\widetilde{\mathbb{E}}[1] = 1$ (normalization) and (ii) $\widetilde{\mathbb{E}}[P^2] \geq 0$ for every $P \in \mathcal{R}^{\leq t}$ (non-negative on Sum-of-Squares).[7]

6. Note that $\{\mathbf{Z}_{i,a}\}_{i \in [n], a \in [q]}$ are formal variables in the SOS formulation.

7. From condition (ii), we can recover the PSD properties from the local PSD ensemble definition.

It also satisfies the problem-dependent constraints

$$\widetilde{\mathbb{E}}\left[\mathbf{Z}_{i,a}^2 \cdot P\right] = \widetilde{\mathbb{E}}\left[\mathbf{Z}_{i,a} \cdot P\right] \qquad \text{and} \qquad \widetilde{\mathbb{E}}\left[\left(\sum_{a\in[q]} \mathbf{Z}_{i,a}\right) \cdot Q\right] = \widetilde{\mathbb{E}}\left[Q\right],$$

for every $i \in [n]$, $a \in [q]$, $P \in \mathcal{R}^{\leq 2t-2}$, and $Q \in \mathcal{R}^{\leq 2t-1}$. Note that for any collection of local random variables $\mathbf{Z}_{i_1}, \ldots, \mathbf{Z}_{i_j}$ with $j \leq 2t$ we have the joint distribution

$$\mathbb{P}(\mathbf{Z}_{i_1} = a_1, \ldots, \mathbf{Z}_{i_j} = a_j) = \widetilde{\mathbb{E}}\left[\mathbf{Z}_{i_1,a_1} \ldots \mathbf{Z}_{i_j,a_j}\right].$$

Even though we may not have a global distribution we can implement a form of pseudo-expectation conditioning on a random variable $\mathbf{Z}_i$ taking a given value $a \in [q]$ as long as $\mathbb{P}\left[\mathbf{Z}_i = a\right] = \widetilde{\mathbb{E}}[\mathbf{Z}_{i,a}] > 0$. This can be done by considering the new operator $\widetilde{\mathbb{E}}_{|Z_i=a} \colon \mathcal{R}^{\leq 2t-2} \to \mathbb{R}$ defined as $\widetilde{\mathbb{E}}_{|\mathbf{Z}_i=a}[\cdot] = \widetilde{\mathbb{E}}[\mathbf{Z}_{i,a}^2 \cdot]/\widetilde{\mathbb{E}}[\mathbf{Z}_{i,a}^2]$, which is a valid pseudo-expectation operator at the $(2t-2)$-th level. This conditioning can be naturally generalized to a set of variables $S \subseteq [n]$ with $|S| \leq t$ satisfying $\mathbf{Z}_S = \alpha$ for some $\alpha \in [q]^S$.

Notation

We make some systematic choices for our parameters in order to syntactically stress their qualitative behavior.

- $1/2 - \varepsilon_0$ is a lower bound on the distance of the base code $\mathcal{C}_1$.
- $1/2 - \varepsilon$ is a lower bound on the distance of the lifted code $\mathcal{C}_k$.
- κ is a parameter that will control the list-decodability of the lifted code $\mathcal{C}_k$.
- μ, θ, η are parameters that can be made arbitrarily small by increasing the SOS degree and/or the quality of expansion.
- β, δ are arbitrary error parameters.

- $\lambda_1 \geq \lambda_2 \geq \cdots$ are the eigenvalues of a graph's adjacency matrix (in $[-1, 1]$).
- $\sigma_1 \geq \sigma_2 \geq \cdots$ are the singular values of a graph's adjacency matrix (in $[0, 1]$).

SOS is an analytic tool so we will identify words over $\mathbb{F}_2$ with words over $\{\pm 1\}$.[8] We also make some choices for words and local variables to distinguish the ground space $\mathbb{F}_2^{X(1)}$ or $\{\pm 1\}^{X(1)}$ form the lifted space $\mathbb{F}_2^{X(k)}$ or $\{\pm 1\}^{X(k)}$.

- $z, z', z'', \ldots$ are words in the ground space $\mathbb{F}_2^{X(1)}$ or $\{\pm 1\}^{X(1)}$.
- $y, y', y'', \ldots$ are words in the lifted space $\mathbb{F}_2^{X(k)}$ or $\{\pm 1\}^{X(k)}$
- $\mathbf{Z} \coloneqq \{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ is a local PSD ensemble on the ground set $X(1)$.
- $\mathbf{Y} \coloneqq \{\mathbf{Y}_{\mathfrak{s}} \coloneqq (\mathrm{dsum}(\mathbf{Z}))_{\mathfrak{s}} \mid \mathfrak{s} \in X(k)\}$ is a local ensemble on $X(k)$.

2.3 Proof Strategy and Organization

As discussed earlier, we view the problem of finding the closest codeword(s) as that of finding suitable solution(s) to an instance of a CSP (which is k-XOR in the case of direct sum). We now discuss some of the technical ingredients required in the decoding procedure.

Unique Decoding. Given $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ with the lifting function as k-XOR, we can view the problem of finding the closest codeword to a given $\tilde{y} \in \mathbb{F}_2^{X(k)}$ as that of finding the unique $z \in \mathcal{C}_1$ satisfying the maximum number of equations of the form $\sum_{i \in \mathfrak{s}} z_i = \tilde{y}_{\mathfrak{s}} \pmod 2$, with one equation for each $\mathfrak{s} \in X(k)$. By this property, $y = \mathrm{dsum}(z)$ is the unique codeword of $\mathcal{C}_k$ closest to $\tilde{y}$. Using the results of [AJT19], it is indeed possible to find $z' \in \mathbb{F}_2^n$ such that $\Delta(\mathrm{dsum}(z'), \tilde{y}) \leq \Delta(\mathrm{dsum}(z), \tilde{y}) + \beta$ for any $\beta > 0$. We then argue that z' or its complement $\overline{z'}$ must be close to $z \in \mathcal{C}_1$, which can then be recovered by unique decoding.

If this is not the case, then $z - z'$ must have bias bounded away from 1, which would imply by robustness (parity sampling property of the hypergraph) that $\mathrm{dsum}(z - z')$ has

8. For this, we can use any bijection from $\mathbb{F}_2 \to \{\pm 1\}$.

bias close to zero, i.e., $\Delta(\mathrm{dsum}(z), \mathrm{dsum}(z')) \approx 1/2$. However, if $\Delta(\tilde{y}, \mathcal{C}_k) \leq \eta$, then we must have

$$\Delta(\mathrm{dsum}(z), \mathrm{dsum}(z')) \;\leq\; \Delta(\mathrm{dsum}(z), \tilde{y}) + \Delta(\mathrm{dsum}(z'), \tilde{y}) \;\leq\; 2\eta + \beta\,,$$

which leads to a contradiction if η is significantly below $1/4$ and β is sufficiently small.

List Decoding. We start by describing an abstract list decoding framework which only assumes two general properties of a lifting $\mathrm{dsum}^g_{X(k)}$: (i) it is distance amplifying (*robust*) and (ii) it is amenable to SOS rounding (*tensorial*).

Suppose $\tilde{y} \in \mathbb{F}_2^{X(k)}$ is a word promised to be $(1/2 - \sqrt{\varepsilon})$-close to a lifted code $\mathcal{C}_k = \mathrm{dsum}(\mathcal{C}_1)$ where $\mathcal{C}_k$ has distance at least $1/2 - \varepsilon$ and $\mathcal{C}_1$ has distance at least $1/2 - \varepsilon_0$. By list decoding $\tilde{y}$, we mean finding a list $\mathcal{L} \subseteq \mathcal{C}_k$ of all codewords $(1/2 - \sqrt{\varepsilon})$-close to $\tilde{y}$.

Our framework for list decoding $\tilde{y}$ consists of three stages. In the first stage, we set up and solve a natural SOS program which we treat abstractly in this discussion.[9] One issue with using a rounding algorithm for this relaxation to do list decoding is that this natural SOS program may return a solution that is "concentrated", e.g., a SOS solution corresponding to single codeword in $\mathcal{L}$. Such a solution will of course not have enough information to recover the entire list. To address this issue we now ask not only for feasibility in our SOS program but also to minimize a convex function Ψ measuring how concentrated the SOS solution is. Specifically, if $\mathbf{Z}$ is the PSD ensemble corresponding to the solution of the SOS program and if $\mathbf{Y}$ is the lifted ensemble, then we minimize $\Psi \;:=\; \mathbb{E}_{\mathfrak{s},\mathfrak{t} \in X(k)} \Big[\Big(\widetilde{\mathbb{E}}[\mathbf{Y}_\mathfrak{s} \mathbf{Y}_\mathfrak{t}] \Big)^2 \Big]$.

The key property of the function Ψ is that if the SOS solution "misses" any element in the list $\mathcal{L}$ then it is possible to decrease it. Since our solution is a minimizer of Ψ, this is impossible.[10] Therefore, our solution does "cover" the list $\mathcal{L}$. Even with this SOS cover of $\mathcal{L}$, the list decoding task is not complete. So far we have not talked about rounding, which

9. The precise SOS program used is given in Section 2.6.2.

10. Actually an approximate minimizer is enough in our application.

is necessary to extract codewords out of the (fractional) solution. For now, we will simply assume that rounding is viable (this is handled by the second stage of the framework) and resume the discussion.

Unfortunately, the covering guarantee is somewhat weak, namely, for $y \in \mathcal{L}$ we are only able to obtain a word $y' \in \mathbb{F}_2^{X(k)}$ with weak agreement $\left|\left\langle y', y \right\rangle\right| \geq 2 \cdot \varepsilon$. Converting a word y' from the cover into an actual codeword y is the goal of the third and final stage of the list decoding framework, dubbed *Cover Purification*. At this point we resort to the robustness properties of the lifting and the fact that we actually have "coupled" pairs $(z, y = \mathrm{dsum}(z))$ and $(z', y' = \mathrm{dsum}(z'))$ for some $z, z' \in \mathbb{F}_2^{X(1)}$. Due to this robustness (and up to some minor technicalities) even a weak agreement between y and y' in the lifted space translates into a much stronger agreement between z and z' in the ground space. Provided the latter agreement is sufficiently strong, z' will lie in the unique decoding ball centered at z in $\mathcal{C}_1$. In this case, we can uniquely recover z and thus also $y = \mathrm{dsum}(z)$. Furthermore, if $\mathcal{C}_1$ admits an efficient unique decoder, we can show that this step in list decoding $\tilde{y}$ can be done efficiently.

Now we go back to fill in the rounding step, which constitutes the second stage of the framework, called *Cover Retrieval*. We view the SOS solution as composed of several "slices" from which the weak pairs (z', y') are to be extracted. Note that the framework handles, in particular, k-XOR liftings where it provides not just a single solution but a list of them. Hence, some structural assumption about $X(k)$ is necessary to ensure SOS tractability. Recall that random k-XOR instances are hard for SOS [Gri01, KMOW17]. For this reason, we impose a sufficient tractability condition on $X(k)$ which we denote the *two-step tensorial* property. This notion is a slight strengthening of a *tensorial* property which was (implicitly) first investigated by Barak et al. [BRS11] when $k = 2$ and later generalized for arbitrary $k \geq 2$ in [AJT19]. Roughly speaking, if $X(k)$ is tensorial then the SOS local random variables in a typical slice of the solution behave approximately as product variables from the perspective of the local views $\mathfrak{s} \in X(k)$. A two-step tensorial structure is a tensorial structure in which

the local random variables between pairs of local views $\mathfrak{s}, \mathfrak{t} \in X(k)$ are also close to product variables, which is an extra property required to perform rounding in this framework. With the two-step tensorial assumption, we are able to round the SOS solution to obtain a list of pairs (z', y') weakly agreeing with elements of the code list that will be refined during cover purification.

To recapitulate, the three stages of the abstract list decoding framework are summarized in Fig. 2.1 along with the required assumptions on the lifting.

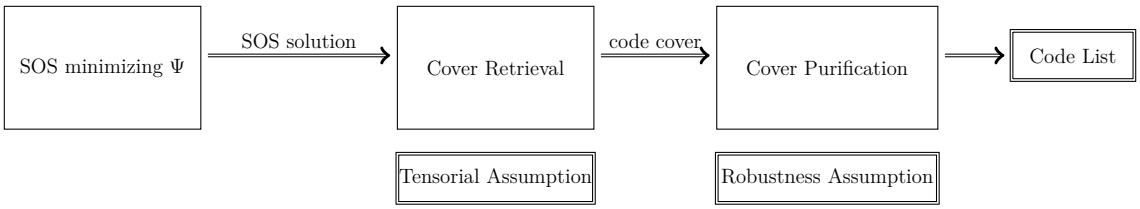

Figure 2.1: The list decoding framework with the assumptions required in each stage.

Finding suitable hypergraphs. Fortunately, objects satisfying the necessary tensorial and robustness assumptions do exist. HDXs were shown to be tensorial in [AJT19], and here we strengthen this result to two-step tensorial as well as prove that HDXs possess the particular robustness property of parity sampling. Walks on expander graphs are already known to be robust [TS17], and we use a modified version of the methods in [AJT19] to show they are also two-step tensorial. For both HDXs and expander walks, we describe how to use known constructions of these objects to get explicit direct sum encodings that can be decoded using our abstract framework.

Reduction from direct product to direct sum. Finally, we describe how to use list decoding results for direct sum codes to obtain results for direct product codes. Given a direct product lifting $\mathcal{C}_k$ on the hypergraph $X(k)$, if $\Delta(\tilde{y}, y) \leq 1 - \varepsilon$ for $y \in \mathcal{C}_k$, then we

must have that

$$\Pr_{\mathfrak{s}\in X(k)}[y_{\mathfrak{s}} = \tilde{y}_{\mathfrak{s}}] \;=\; \mathop{\mathbb{E}}_{\mathfrak{s}\in X(k)}\left[\mathop{\mathbb{E}}_{\mathfrak{t}\subseteq\mathfrak{s}}[\chi_{\mathfrak{t}}(y_{\mathfrak{s}} + \tilde{y}_{\mathfrak{s}})]\right] \;\geq\; \varepsilon\,.$$

Since $\chi_{\mathfrak{t}}(y_{\mathfrak{s}})$ can be viewed as part of a direct sum lifting, we get by grouping subsets $\mathfrak{t}$ by size that there must exist a size i such that the direct sum lifting using $X(i)$ has correlation at least ε with the word y' defined as $y'_{\mathfrak{t}} = \chi_t(\tilde{y}_{\mathfrak{s}})$ for all $\mathfrak{t} \in X(i)$. We can then apply the list decoding algorithm for direct sum codes on $X(i)$. A standard concentration argument can also be used to control the size i to be approximately $k/2$.

Organization of Results

In Section 2.4, we show how the direct sum lifting on HDXs can be used to reduce bias, establishing that HDXs are parity samplers. This will give a very concrete running example of a lifting that can be used in our framework. Before addressing list decoding, we remark in Section 2.5 how this lifting can be used in the simpler regime of unique decoding using a k-CSP algorithm on expanding instances [AJT19]. The abstract list decoding framework is given in Section 2.6. Next, we instantiate the framework with the direct sum lifting on HDXs in Section 2.7. As an interlude between the first and second instantiation, Section 2.8 describes how the first concrete instantiation of Section 2.7 captures the direct product lifting on HDXs via a reduction to the direct sum lifting. Finally, in Section 2.9, we show how to instantiate the framework with the direct sum lifting on the collection of length $k-1$ walks of an expander graph.

2.4 Pseudorandom Hypergraphs and Robustness of Direct Sum

The main robustness property we will consider is parity sampling applied to the case of the direct sum lifting. As this section focuses on this specific instance of a lifting, here we will

say that a collection $X(k)$ is a parity sampler if its associated direct sum lifting $\mathrm{dsum}_{X(k)}$ is a parity sampler. Recall that for such a parity sampler, the direct sum lifting brings the bias of a code close to zero, which means it boosts the distance almost to 1/2.

2.4.1 Expander Walks and Parity Sampling

A known example of a parity sampler is the set $X(k)$ of all walks of length k in a sufficiently expanding graph, as shown by Ta-Shma.

Theorem 2.4.1 (Walks on Expanders are Parity Samplers [TS17])**.** *Suppose G is a graph with second largest singular value at most λ, and let $X(k)$ be the set of all walks of length k on G. Then $X(k)$ is a $(\beta_0, (\beta_0 + 2\lambda)^{\lfloor k/2 \rfloor})$-parity sampler.*

Our goal in this section is to prove a similar result for high-dimensional expanders, where $X(k)$ is the set of k-sized faces.

2.4.2 High-dimensional Expanders

A high-dimensional expander (HDX) is a particular kind of simplicial complex satisfying an expansion requirement. We recall the notion of high-dimensional expansion considered in [DK17]. For a complex $X(\leq d)$ and $\mathfrak{s} \in X(i)$ for some $i \in [d]$, we denote by $X_{\mathfrak{s}}$ the *link complex*

$$X_{\mathfrak{s}} \;:=\; \{\mathfrak{t} \backslash \mathfrak{s} \mid \mathfrak{s} \subseteq \mathfrak{t} \in X\}\,.$$

When $|\mathfrak{s}| \leq d-2$, we also associate a natural weighted graph $G(X_{\mathfrak{s}})$ to a link $X_{\mathfrak{s}}$, with vertex set $X_{\mathfrak{s}}(1)$ and edge set $X_{\mathfrak{s}}(2)$. The edge weights are taken to be proportional to the measure Π_2 on the complex $X_{\mathfrak{s}}$, which is in turn proportional to the measure $\Pi_{|\mathfrak{s}|+2}$ on X. The graph $G(X_{\mathfrak{s}})$ is referred to as the *skeleton* of $X_{\mathfrak{s}}$.

Dinur and Kaufman [DK17] define high-dimensional expansion in terms of spectral expansion of the skeletons of the links.

Definition 2.4.2 (γ-HDX from [DK17])**.** *A simplicial complex $X(\leq d)$ is said to be* γ-High Dimensional Expander (γ-HDX) *if for every $0 \leq i \leq d-2$ and for every $\mathfrak{s} \in X(i)$, the graph $G(X_{\mathfrak{s}})$ satisfies $\sigma_2(G(X_{\mathfrak{s}})) \leq \gamma$.*

We will need the following theorem relating γ to the spectral properties of the graph between two layers of an HDX.

Theorem 2.4.3 (Adapted from [DK17])**.** *Let X be a γ-HDX and let $M_{1,d}$ be the weighted bipartite containment graph between $X(1)$ and $X(d)$, where each edge $(\{i\}, \mathfrak{s})$ has weight $(1/d)\Pi_d(\mathfrak{s})$. Then the second largest singular value σ_2 of $M_{1,d}$ satisfies*

$$\sigma_2^2 \leq \frac{1}{d} + O(d\gamma).$$

We will be defining codes using HDXs by associating each face in some $X(i)$ with a position in the code. The distance between two codewords does not take into account any weights on their entries, which will be problematic when decoding since the distributions Π_i are not necessarily uniform. To deal with this issue, we will work with HDXs where the distributions Π_i satisfy a property only slightly weaker than uniformity.

Definition 2.4.4 (Flatness (from [DHK$^+$19]))**.** *We say that a distribution Π on a finite probability space Ω is D-flat if there exits N such that each singleton $\omega \in \Omega$ has probability in $\{1/N, \ldots, D/N\}$.*

Using the algebraically deep construction of Ramanujan complexes by Lubotzky, Samuels, and Vishne [LSV05b, LSV05a], Dinur and Kaufman [DK17] showed that sparse γ-HDXs do exist, with flat distributions on their sets of faces. The following lemma from [DHK$^+$19] is a refinement of [DK17].

Lemma 2.4.5 (Extracted from [DHK$^+$19])**.** *For every $\gamma > 0$ and every $d \in \mathbb{N}$ there exists an explicit infinite family of bounded degree d-sized complexes which are γ-HDXs. Furthermore,*

there exists a $D \le (1/\gamma)^{O(d^2/\gamma^2)}$ *such that*

$$\frac{|X(d)|}{|X(1)|} \;\le\; D,$$

the distribution Π_1 *is uniform, and the other distributions* $\Pi_d, \ldots, \Pi_2$ *are* D*-flat.*

For a D-flat distribution Π_i, we can duplicate each face in $X(i)$ at most D times to make Π_i the same as a uniform distribution on this multiset. We will always perform such a duplication implicitly when defining codes on $X(i)$.

2.4.3 HDXs are Parity Samplers

To prove that sufficiently expanding HDXs are parity samplers, we establish some properties of the complete complex and then explore the fact that HDXs are locally complete.[11] We first show that the expectation over k-sized faces of a complete complex X on t vertices approximately splits into a product of k expectations over $X(1)$ provided $t \gg k^2$.

Claim 2.4.6 (Complete complex and near independence)**.** *Suppose* X *is the complete complex of dimension at least* k *with* Π_k *uniform over* $X(k)$ *and* Π_1 *uniform over* $X(1) = [t]$*. For a function* $f : X(1) \to \mathbb{R}$*, let*

$$\mu_k = \mathbb{E}_{\mathfrak{s}\sim\Pi_k}\left[\prod_{i\in\mathfrak{s}} f(i)\right] \qquad \text{and} \qquad \mu_1 = \mathbb{E}_{i\sim\Pi_1}\left[f(i)\right].$$

Then

$$\left|\mu_k - \mu_1^k\right| \le \frac{k^2}{t}\left\|f\right\|_\infty^k.$$

Proof. Let $\mathcal{E} = \{(i_1, \ldots, i_k) \in X(1)^k \mid i_1, \ldots, i_k \text{ are distinct}\}$, $\delta = \mathbb{P}_{i_1,\ldots,i_k\sim\Pi_1}[(i_1, \ldots, i_k) \notin$

11. This a recurring theme in the study of HDXs [DK17].

$\mathcal{E}]$, and $\eta = \mathbb{E}_{(i_1,\ldots,i_k)\in X(1)^k\setminus\mathcal{E}}[f(i_1)\cdots f(i_k)]$. Then

$$\begin{aligned}\mu_1^k &= \mathbb{E}_{i_1,\ldots,i_k\sim\Pi_1}[f(i_1)\cdots f(i_k)]\\ &= (1-\delta)\cdot\mathbb{E}_{(i_1,\ldots,i_k)\in\mathcal{E}}[f(i_1)\cdots f(i_k)] + \delta\cdot\mathbb{E}_{(i_1,\ldots,i_k)\in X(1)^k\setminus\mathcal{E}}[f(i_1)\cdots f(i_k)]\\ &= (1-\delta)\cdot\mu_k + \delta\cdot\eta,\end{aligned}$$

where the last equality follows since Π_k is uniform and the product in the expectation is symmetric. As $i_1,\ldots,i_k$ are sampled independently from Π_1, which is uniform over $X(1)$,

$$\delta = 1 - \prod_{j<k}\left(1-\frac{j}{t}\right) \le \sum_{j<k}\frac{j}{t} = \frac{k(k-1)}{2t},$$

so we have

$$\left|\mu_k - \mu_1^k\right| = \delta\left|\mu_k - \eta\right| \le \frac{k^2}{2t}\left(2\left\|f\right\|_\infty^k\right).$$

■

We will derive parity sampling for HDXs from their behavior as samplers. A sampler is a structure in which the average of any function on a typical local view is close to its overall average. More precisely, we have the following definition.

Definition 2.4.7 (Sampler)**.** *Let $G = (U, V, E)$ be a bipartite graph with a probability distribution Π_U on U. Let Π_V be the distribution on V obtained by choosing $u \in U$ according to Π_U, then a uniformly random neighbor v of u. We say that G is an (η, δ)-sampler if for every function $f\colon V \to [0,1]$ with $\mu = \mathbb{E}_{v\sim\Pi_V} f(v)$,*

$$\mathop{\mathbb{P}}_{u\sim\Pi_U}\left[\left|\mathbb{E}_{v\sim u}[f(v)] - \mu\right| \ge \eta\right] \;\le\; \delta.$$

To relate parity sampling to spectral expansion, we use the following fact establishing that samplers of arbitrarily good parameters (η, δ) can be obtained from sufficiently expanding

bipartite graphs. This result is essentially a corollary of the expander mixing lemma.

Fact 2.4.8 (From Dinur et al. [DHK+19])**.** *A weighted bipartite graph with second singular value σ_2 is an $(\eta, \sigma_2^2/\eta^2)$-sampler.*

Using Claim 2.4.6, we show that the graph between $X(1)$ and $X(k)$ obtained from a HDX is a parity sampler, with parameters determined by its sampling properties.

Claim 2.4.9 (Sampler bias amplification)**.** *Let $X(\leq d)$ be a HDX such that the weighted bipartite graph $M_{1,d}$ between $X(1) = [n]$ and $X(d)$ is an (η, δ)-sampler. For any $1 \leq k \leq d$, if $z \in \mathbb{F}_2^n$ has bias at most β_0, then*

$$\mathrm{bias}(\mathrm{dsum}_{X(k)}(z)) \leq (\beta_0 + \eta)^k + \frac{k^2}{d} + \delta.$$

Proof. By downward closure, the subcomplex $X|_{\mathfrak{t}}$ obtained by restricting to edges contained within some $\mathfrak{t} \in X(d)$ is a complete complex on the ground set $\mathfrak{t}$. Since $M_{1,d}$ is an (η, δ)-sampler, the bias of $z|_{\mathfrak{t}}$ must be within η of $\mathrm{bias}(z)$ on all but δ fraction of the edges $\mathfrak{t}$. Hence

$$\begin{aligned}
\mathrm{bias}(\mathrm{dsum}_{X(k)}(z)) &= \left|\mathbb{E}_{\{i_1,\ldots,i_k\}\sim\Pi_k}(-1)^{z_{i_1}+\cdots+z_{i_k}}\right| \\
&= \left|\mathbb{E}_{\mathfrak{t}\sim\Pi_d}\mathbb{E}_{\{i_1,\ldots,i_k\}\in X|_{\mathfrak{t}}(k)}(-1)^{z_{i_1}+\cdots+z_{i_k}}\right| \\
&\leq \left|\mathbb{E}_{\mathfrak{t}\sim\Pi_d}\mathbb{E}_{\{i_1,\ldots,i_k\}\in X|_{\mathfrak{t}}(k)}(-1)^{z_{i_1}+\cdots+z_{i_k}}\mathbb{1}_{[\mathrm{bias}(z|_{\mathfrak{t}})\leq\beta_0+\eta]}\right| \\
&\qquad + \mathop{\mathbb{P}}_{t\sim\Pi_d}[\mathrm{bias}(z|_{\mathfrak{t}}) > \beta_0 + \eta] \\
&\leq \mathbb{E}_{\mathfrak{t}\sim\Pi_d}\mathbb{1}_{[\mathrm{bias}(z|_{\mathfrak{t}})\leq\beta_0+\eta]}\left|\mathbb{E}_{\{i_1,\ldots,i_k\}\in X|_{\mathfrak{t}}(k)}(-1)^{z_{i_1}+\cdots+z_{i_k}}\right| + \delta.
\end{aligned}$$

By Claim 2.4.6, the magnitude of the expectation of $(-1)^{z_i}$ over the edges of size k in

the complete complex $X|_{\mathfrak{t}}$ is close to $\left|\mathbb{E}_{i\sim X|_{\mathfrak{t}}(1)}(-1)^{z_i}\right|$, which is just the bias of $z|_{\mathfrak{t}}$. Then

$$\begin{aligned}\operatorname{bias}(\operatorname{dsum}_{X(k)}(z)) &\leq \mathbb{E}_{\mathfrak{t}\sim X(d)} \mathbb{1}_{[\operatorname{bias}(z|_{\mathfrak{t}})\leq\beta_0+\eta]} \operatorname{bias}(z|_{\mathfrak{t}})^k + \frac{k^2}{d} + \delta \\ &\leq (\beta_0+\eta)^k + \frac{k^2}{d} + \delta\end{aligned}$$

■

Now we can compute the parameters necessary for a HDX to be an (β_0, β)-parity sampler for arbitrarily small β.

Lemma 2.4.10 (HDXs are parity samplers)**.** *Let $0 < \beta \leq \beta_0 < 1$, $0 < \theta < (1/\beta_0) - 1$, and $k \geq \log_{(1+\theta)\beta_0}(\beta/3)$. If $X(\leq d)$ is a γ-HDX with $d \geq \max\{3k^2/\beta, 6/(\theta^2\beta_0^2\beta)\}$ and $\gamma = O\left(1/d^2\right)$, then $X(k)$ is a (β_0, β)-parity sampler.*

Proof. Suppose the graph $M_{1,d}$ between $X(1)$ and $X(d)$ is an (η, δ)-sampler. We will choose d and γ so that $\eta = \theta\beta_0$ and $\delta = \beta/3$. Using Fact 2.4.8 to obtain a sampler with these parameters, we need the second singular value σ_2 of $M_{1,d}$ to be bounded as

$$\sigma_2 \leq \theta\beta_0\sqrt{\frac{\beta}{3}}.$$

By the upper bound on σ_2^2 from Theorem 2.4.3, it suffices to have

$$\frac{1}{d} + O\left(d\gamma\right) \leq \frac{\theta^2\beta_0^2\beta}{3},$$

which is satisfied by taking $d \geq 6/\left(\theta^2\beta_0^2\beta\right)$ and $\gamma = O\left(1/d^2\right)$.

By Claim 2.4.9, $X(k)$ is a $(\beta_0, (\beta_0+\eta)^k + k^2/d + \delta)$-parity sampler. The first term in the bias is $(\beta_0+\eta)^k = ((1+\theta)\beta_0)^k$, so we require $(1+\theta)\beta_0 < 1$ to amplify the bias by making k large. To make this term smaller than $\beta/3$, k must be at least $\log_{(1+\theta)\beta_0}(\beta/3)$. We already chose $\delta = \beta/3$, so ensuring $d \geq 3k^2/\beta$ gives us a (β_0, β)-parity sampler. ■

2.4.4 Rate of the Direct Sum Lifting

By applying the direct sum lifting on a HDX to a base code $\mathcal{C}_1$ with bias β_0, parity sampling allows us to obtain a code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ with arbitrarily small bias β at the cost of increasing the length of the codewords. The following lemma gives a lower bound on the rate of the lifted code $\mathcal{C}_k$.

Lemma 2.4.11 (Rate of direct sum lifting for a HDX)**.** *Let $\beta_0 \in (0,1)$ and $\theta \in (0, (1/\beta_0)-1)$ be constants, and let $\mathcal{C}_1$ be an β_0-biased binary linear code with relative rate r_1. For $\beta \in (0, \beta_0]$, suppose k, d, and γ satisfy the hypotheses of Lemma 2.4.10, with k and d taking the smallest values that satisfy the lemma. The relative rate r_k of the code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ with bias β constructed on a HDX with these parameters satisfies*

$$r_k \geq r_1 \cdot \gamma^{O((\log(1/\beta))^4/(\beta^2\gamma^2))}.$$

If $\gamma = C/d^2$ for some constant C, then this becomes

$$r_k \geq r_1 \cdot \left(\frac{\beta^2}{(\log(1/\beta))^4}\right)^{O((\log(1/\beta))^{12}/\beta^6)}.$$

Proof. Performing the lifting from $\mathcal{C}_1$ to $\mathcal{C}_k$ does not change the dimension of the code, but it does increase the length of the codewords from n to $|X(k)|$, where $|X(k)|$ is the size of the multiset of edges of size k after each edge has been copied a number of times proportional to its weight. Using the bound and flatness guarantee from Lemma 2.4.5, we can compute

$$r_k = \frac{r_1 n}{|X(k)|} \geq \frac{r_1}{D^2},$$

where $D \leq (1/\gamma)^{O(d^2/\gamma^2)}$. Treating β_0 and θ as constants, the values of k and d necessary

to satisfy Lemma 2.4.10 are

$$k = \log_{(1+\theta)\beta_0}(\beta/3) = O(\log(1/\beta))$$

and

$$d = \max\left\{\frac{3k^2}{\beta}, \frac{6}{\theta^2\beta_0^2\beta}\right\} = O\left(\frac{(\log(1/\beta))^2}{\beta}\right).$$

Putting this expression for d into the inequality for D yields

$$D \leq (1/\gamma)^{O((\log(1/\beta))^4/(\beta^2\gamma^2))},$$

from which the bounds in the lemma statement follow. ∎

From Lemma 2.4.11, we see that if $\mathcal{C}_1$ has constant rate, then $\mathcal{C}_k$ has a rate which is constant with respect to n. However, the dependence of the rate on the bias β is quite poor. This is especially striking in comparison to the rate achievable using Ta-Shma's expander walk construction described in Section 2.4.1.

Lemma 2.4.12 (Rate of direct sum lifting for expander walks [TS17])**.** *Let $\beta_0 \in (0,1)$ be a constant and $\mathcal{C}_1$ be an β_0-biased binary linear code with relative rate r_1. Fix $\beta \in (0, \beta_0]$. Suppose G is a graph with second largest singular value $\lambda = \beta_0/2$ and degree $d \leq 4/\lambda^2$. Let $k = 2\log_{2\beta_0}(\beta) + 1$ and $X(k)$ be the set of all walks of length k on G. Then the direct sum lifting $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ has bias β and rate $r_k \geq r_1 \cdot \beta^{O(1)}$.*

Proof. From Theorem 2.4.1 with this choice of λ and k, the direct sum lifting $\mathcal{C}_k$ has bias β. For the rate, observe that the lifting increases the length of the codewords from n to the number of walks of length k on G, which is nd^k. Thus the rate of $\mathcal{C}_k$ is

$$r_k = \frac{r_1 n}{nd^k} = \frac{r_1}{d^k}$$

As $d \leq 16/\beta_0$, which is a constant, and $k = O(\log(1/\beta))$, the rate satisfies $r_k \geq r_1 \cdot \beta^{O(1)}$. ■

2.5 Unique Decoding

In this section, we will show how parity sampling and the ability to solve k-XOR instances with $X(k)$ as their constraint complex allow us to decode the direct sum lifting $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ of a linear base code $\mathcal{C}_1 \in \mathbb{F}_2^n$. With a more technical argument, we can also handle different kinds of liftings and non-linear codes, but for clarity of exposition we restrict our attention to the preceding setting.

2.5.1 Unique Decoding on Parity Samplers

Our approach to unique decoding for $\mathcal{C}_k$ is as follows. Suppose a received word $\tilde{y} \in \mathbb{F}_2^{X(k)}$ is close to $y^\star \in \mathcal{C}_k$, which is the direct sum lifting of some $z^\star \in \mathcal{C}_1$ on $X(k)$. We first find an approximate solution $z \in \mathbb{F}_2^n$ to the k-XOR instance $\mathfrak{I}(X(k), \tilde{y})$ with predicates

$$\sum_{i \in \mathfrak{s}} z_i = \tilde{y}_{\mathfrak{s}} \pmod 2$$

for every $\mathfrak{s} \in X(k)$. Note that z being an approximate solution to $\mathfrak{I}(X(k), \tilde{y})$ is equivalent to its lifting $\mathrm{dsum}_{X(k)}(z)$ being close to $\tilde{y}$. In Lemma 2.5.1, we show that if $\mathrm{dsum}_{X(k)}$ is a sufficiently strong parity sampler, either z or its complement $\overline{z}$ will be close to $z^\star$. Running the unique decoding algorithm for $\mathcal{C}_1$ on z and $\overline{z}$ will recover $z^\star$, from which we can obtain $y^\star$ by applying the direct sum lifting.

Lemma 2.5.1. *Let $0 < \varepsilon < 1/2$ and $0 < \beta < 1/4 - \varepsilon/2$. Suppose $\mathcal{C}_1$ is a linear code that is efficiently uniquely decodable within radius $1/4 - \mu_0$ for some $\mu_0 > 0$, and $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ where $\mathrm{dsum}_{X(k)}$ is a $(1/2 + 2\mu_0, 2\varepsilon)$-parity sampler. Let $\tilde{y} \in \mathbb{F}_2^{X(k)}$ be a word that has distance strictly less than $(1/4 - \varepsilon/2 - \beta)$ from $\mathcal{C}_k$, and let $y^\star = \mathrm{dsum}_{X(k)}(z^\star) \in \mathcal{C}_k$ be the word closest to $\tilde{y}$.*

Then, for any $z \in \mathbb{F}_2^n$ satisfying

$$\Delta(\mathrm{dsum}_{X(k)}(z), \tilde{y}) < \frac{1}{4} - \frac{\varepsilon}{2},$$

we have either

$$\Delta(z^\star, z) \leq \frac{1}{4} - \mu_0 \quad \textit{or} \quad \Delta(z^\star, \overline{z}) \leq \frac{1}{4} - \mu_0.$$

In particular, either z or $\overline{z}$ can be efficiently decoded in $\mathcal{C}_1$ to obtain $z^\star \in \mathcal{C}_1$.

Remark 2.5.2. *Since $\mathrm{dsum}_{X(k)}$ is a $(1/2+2\mu_0, 2\varepsilon)$-parity sampler, the code $\mathcal{C}_k$ has distance $\Delta(\mathcal{C}_k) \geq 1/2 - \varepsilon$. This implies that $z^\star \in \mathcal{C}_1$ is unique, since its direct sum lifting $y^\star$ is within distance $\Delta(\mathcal{C}_k)/2$ of $\tilde{y}$.*

Proof. Let $y = \mathrm{dsum}_{X(k)}(z)$. We have

$$\Delta(y^\star, y) \leq \Delta(y^\star, \tilde{y}) + \Delta(y, \tilde{y}) < \frac{1}{2} - \varepsilon.$$

By linearity of $\mathrm{dsum}_{X(k)}$, $\Delta(\mathrm{dsum}_{X_k}(z^\star - z), 0) < 1/2 - \varepsilon$, so $\mathrm{bias}(\mathrm{dsum}_k(z^\star - z)) > 2\varepsilon$. From the $(1/2 + 2\mu_0, 2\varepsilon)$-parity sampling assumption, $\mathrm{bias}(z^\star - z) > 1/2 + 2\mu_0$. Translating back to distance, either $\Delta(z^\star, z) < 1/4 - \mu_0$ or $\Delta(z^\star, z) > 3/4 + \mu_0$, the latter being equivalent to $\Delta(z^\star, \overline{z}) < 1/4 - \mu_0$. ∎

To complete the unique decoding algorithm, we need only describe how a good enough approximate solution $z \in \mathbb{F}_2^n$ to a k-XOR instance $\mathfrak{I}(X(k), \tilde{y})$ allows us to recover $z^\star \in \mathcal{C}_1$ provided $\tilde{y}$ is sufficiently close to $\mathcal{C}_k$.

Corollary 2.5.3. *Suppose $\mathcal{C}_1$, $X(k)$, $z^\star$, $y^\star$ and $\tilde{y}$ are as in the assumptions of Lemma 2.5.1. If $z \in \mathbb{F}_2^n$ is such that*

$$\mathsf{SAT}_{\mathfrak{I}(X(k),\tilde{y})}(z) \geq \mathsf{OPT}_{\mathfrak{I}(X(k),\tilde{y})} - \beta,$$

then unique decoding either z or $\overline{z}$ gives $z^\star \in \mathcal{C}_1$. Furthermore, if such a z can be found

efficiently, so can $z^\star$.

Proof. By the assumption on z, we have

$$\begin{aligned} 1 - \Delta(\mathrm{dsum}_{X(k)}(z), \tilde{y}) &= \mathsf{SAT}_{\mathfrak{J}(X(k),\tilde{y})}(z) \\ &\geq \mathsf{OPT}_{\mathfrak{J}(X(k),\tilde{y})} - \beta \\ &\geq \mathsf{SAT}_{\mathfrak{J}(X(k),\tilde{y})}(z^\star) - \beta \\ &= 1 - \Delta(y^\star, \tilde{y}) - \beta, \end{aligned}$$

implying $\Delta(\mathrm{dsum}_{X(k)}(z), \tilde{y}) \leq \Delta(y^\star, \tilde{y}) + \beta$. Using the assumption that $\tilde{y}$ has distance strictly less than $(1/4 - \varepsilon/2 - \beta)$ from $\mathcal{C}_k$, we get that $\Delta(\mathrm{dsum}_{X(k)}(z), \tilde{y}) < 1/4 - \varepsilon/2$, in which case we satisfy all of the conditions required for Lemma 2.5.1. ∎

2.5.2 Concrete Instantiations

High Dimensional Expanders

If $X(k)$ is the collection of k-faces of a sufficiently expanding γ-HDX, we can use the following algorithm to approximately solve the k-XOR instance $\mathfrak{J}(X(k), \tilde{y})$ and obtain $z \in \mathbb{F}_2^n$.

Theorem 2.5.4 ([AJT19])**.** *Let $\mathfrak{J}$ be an instance of MAX k-CSP on n variables taking values over an alphabet of size q, and let $\beta > 0$. Let the simplicial complex $X_{\mathfrak{J}}$ be a γ-HDX with $\gamma = \beta^{O(1)} \cdot (1/(kq))^{O(k)}$.*

There is an algorithm based on $(k/\beta)^{O(1)} \cdot q^{O(k)}$ levels of the Sum-of-Squares hierarchy which produces an assignment satisfying at least an $(\mathsf{OPT}_{\mathfrak{J}} - \beta)$ fraction of the constraints in time $n^{(k/\beta)^{O(1)} \cdot q^{O(k)}}$.

If X is a HDX with the parameters necessary to both satisfy this theorem and be a $(1/2 + 2\mu_0, 2\varepsilon)$ parity sampler, we can combine this with Corollary 2.5.3 to achieve efficient unique decodability of $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$.

Corollary 2.5.5. *Let $X(\leq d)$ be a d-dimensional γ-HDX satisfying the premises of Lemma 2.4.10 that would guarantee that $X(k)$ is a $(1/2 + 2\mu_0, 2\varepsilon)$-parity sampler, and let $\mathcal{C}_1 \subseteq \mathbb{F}_2^n$ be a linear code which is efficiently unique decodable within radius $1/4 - \mu_0$ for some $\mu_0 > 0$. Then the code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ can be unique decoded within distance $1/4 - \varepsilon/2 - \beta$ in time $n^{(k/\beta)^{O(1)} \cdot 2^{O(k)}}$,[12] where we have*

$$\beta = (\gamma \cdot (2k)^{O(k)})^{\frac{1}{O(1)}}.$$

Proof. By Lemma 2.4.10, we can achieve $(1/2 + 2\mu_0, 2\varepsilon)$-parity sampling by taking $0 < \theta < \frac{2}{1+4\mu_0} - 1$, $k \geq \log_{(1+\theta)\cdot(\frac{1}{2}+2\mu_0)}(2\varepsilon/3)$, $d \geq \max\left\{\frac{3k^2}{2\varepsilon}, \frac{3}{\theta^2(1/2+2\mu_0)^2\varepsilon}\right\}$, and $\gamma = O(1/d^2)$. Let $\tilde{y} \in \mathbb{F}_2^{X(k)}$ be a received word with distance less than $(1/4 - \varepsilon/2 - \beta)$ from $\mathcal{C}_k$. Applying Theorem 2.5.4 to $\mathfrak{I}(X(k), \tilde{y})$ with $q = 2$ and the given value of β, we obtain a $z \in \mathbb{F}_2^n$ with $\mathsf{SAT}_{\mathfrak{I}(X(k),\tilde{y})}(z) \geq \mathsf{OPT}_{\mathfrak{I}(X(k),\tilde{y})} - \beta$. This z can be used in Corollary 2.5.3 to find z^* and uniquely decode $\tilde{y}$ as $y^* = \mathrm{dsum}_{X(k)}(z^*)$. ∎

Expander Walks

In Section 2.9, we will show that the algorithmic results of [AJT19] can be modified to work when $X(k)$ is a set of tuples of size k which is sufficiently splittable (Corollary 2.9.21), which occurs when $X(k)$ is a set of walks on on a suitably strong expander (Corollary 2.9.18). In particular, we have the following.

Theorem 2.5.6. *Let $G = (V, E)$ be a graph with $\sigma_2(G) = \lambda$ and k be a given parameter. Let $\mathfrak{I}$ be a k-CSP instance over an alphabet of size q whose constraint graph is the set of walks on G of length k. Let $\beta > 0$ be such that $\lambda = O(\beta^2/(k^2 \cdot q^{2k}))$.*

There exists an algorithm based on $O\left(\frac{q^{4k}k^7}{\beta^5}\right)$ levels of the Sum-of-Squares hierarchy which produces an assignment satisfying at least an $(\mathsf{OPT}_{\mathfrak{I}} - \beta)$ fraction of the constraints in time

12. Here we are assuming that uniquely decoding $\mathcal{C}_1$ within radius $1/4 - \mu_0$ takes time less than this.

$n^{O(q^{4k} \cdot k^7/\beta^5)}$.

Using this result, one can efficiently unique decode $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ when $X(k)$ is the set of walks of length k on an expander strong enough to achieve the necessary parity sampling property.

Corollary 2.5.7. *Let $X(k)$ be the set of walks on a graph G with $\sigma_2(G) = \lambda$ such that $\mathrm{dsum}_{X(k)}$ is a $(1/2 + 2\mu_0, 2\varepsilon)$ parity sampler, and let $\mathcal{C}_1 \subseteq \mathbb{F}_2^n$ be a linear code which is efficiently unique decodable within radius $1/4 - \mu_0$ for some $\mu_0 > 0$. Then the code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ can be unique decoded within radius $1/4 - \varepsilon/2 - \beta$ in time $n^{O(2^{4k} \cdot k^7/\beta^5)}$, where we have*

$$\beta = O(\lambda \cdot k^2 \cdot 2^k).$$

Proof. By Theorem 2.4.1, we can obtain a $(1/2 + 2\mu_0, 2\varepsilon)$-parity sampler by ensuring $1/2 + \mu_0 + 2\lambda < 1$ and $k \geq 2\log_{1/2+\mu_0+2\lambda}(2\varepsilon) + 1$. Let $\tilde{y} \in \mathbb{F}_2^{X(k)}$ be a received word with distance less than $(1/4 - \varepsilon/2 - \beta)$ from $\mathcal{C}_k$. Applying Theorem 2.5.6 to $\mathfrak{I}(X(k), \tilde{y})$ with $q = 2$ and the given value of β, we obtain a $z \in \mathbb{F}_2^n$ with $\mathsf{SAT}_{\mathfrak{I}(X(k),\tilde{y})}(z) \geq \mathsf{OPT}_{\mathfrak{I}(X(k),\tilde{y})} - \beta$. This z can be used in Corollary 2.5.3 to find z^* and uniquely decode $\tilde{y}$ as $y^* = \mathrm{dsum}_{X(k)}(z^*)$. ∎

Remark 2.5.8. *In both Corollary 2.5.5 and Corollary 2.5.7, when μ_0 and ε are constants, k can be constant, which means we can decode $\mathcal{C}_k$ from a radius arbitrarily close to $1/4 - \varepsilon/2$ if we have strong enough guarantees on the quality of the expansion of the high-dimensional expander or the graph, respectively.*

Notice, however, that the unique decodability radius of the code $\mathcal{C}_k$ is potentially larger than $1/4 - \varepsilon/2$. Our choice of $(1/2 + 2\mu_0, 2\varepsilon)$-parity sampling is needed to ensure that the approximate k-CSP solutions lie within the unique decoding radius of $\mathcal{C}_1$. Since the bias of the code $\mathcal{C}_1$ will generally be smaller than the parity sampling requirement of $1/2 + 2\mu_0$, the bias of the code $\mathcal{C}_k$ will be smaller than 2ε. In this case, the maximum distance at which our unique decoding algorithm works will be smaller than $\Delta(\mathcal{C}_k)/2$.

2.6 Abstract List Decoding Framework

In this section, we present the abstract list decoding framework with its requirements and prove its guarantees. We introduce the entropic proxy Ψ in Section 2.6.1 and use it to define the SOS program for list decoding in Section 2.6.2. In Section 2.6.3, we establish key properties of Ψ capturing its importance as a list decoding tool. We recall the Propagation Rounding algorithm in Section 2.6.4 and formalize the notion of a slice as a set of assignments to variables in the algorithm. Then, considerations of SOS tractability of the lifting related to tensorial properties are dealt with in Section 2.6.5. Now, assuming we have a fractional SOS solution to our program, the analysis of its covering properties and the precise definition and correctness of the two later stages of the framework are given in Section 2.6.6. This abstract framework will be instantiated using the direct sum lifting: on HDXs in Section 2.7 and on expander walks in Section 2.9.

2.6.1 Entropic Proxy

In our list decoding framework via SOS, we will solve a single optimization program whose resulting pseudo-expectation will in a certain sense be rich enough to cover all intended solutions at once. To enforce this covering property we rely on an analytical artifice, namely, we minimize a convex function Ψ that provides a proxy to how concentrated the SOS solution is. More precisely, we use Ψ from Definition 2.6.1. A similar list decoding technique was also (independently) used by Karmalkar et al. [KKK19] and Raghavendra–Yau [RY19], but in the context of learning.

Definition 2.6.1 (Entropic Proxy)**.** *Let $\mathbf{Y} = \{\mathbf{Y}_\mathfrak{s}\}_{\mathfrak{s}\in X(k)}$ be a t-local PSD ensemble with $t \geq 2$. We define $\Psi = \Psi\left(\{\mathbf{Y}_\mathfrak{s}\}_{\mathfrak{s}\in X(k)}\right)$ as*

$$\Psi \;\coloneqq\; \mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\left(\widetilde{\mathbb{E}}\,[\mathbf{Y}_\mathfrak{s}\mathbf{Y}_\mathfrak{t}]\right)^2.$$

We also denote Ψ *equivalently as* $\Psi = \Psi\left(\widetilde{\mathbb{E}}\right)$ *where* $\widetilde{\mathbb{E}}$ *is the pseudo-expectation operator associated to the ensemble* $\mathbf{Y}$*.*

2.6.2 SOS Program for List Decoding

Let $\tilde{y} \in \{\pm 1\}^{X(k)}$ be a word promised to be $(1/2 - \sqrt{\varepsilon})$-close to a lifted code $\mathcal{C}_k = \mathrm{dsum}(\mathcal{C}_1)$. The word $\tilde{y}$ is to be regarded as a (possibly) corrupted codeword for which we want to do list decoding. We consider the following SOS program.

minimize	$\Psi\left(\{\mathbf{Y}_{\mathfrak{s}}\}_{\mathfrak{s} \in X(k)}\right)$	(List Decoding Program)
subject to		
	$\mathbb{E}_{\mathfrak{s} \sim \Pi_k} \widetilde{\mathbb{E}}\left[\tilde{y}_{\mathfrak{s}} \cdot \mathbf{Y}_{\mathfrak{s}}\right] \geq 2\sqrt{\varepsilon}$	(Agreement Constraint)
	$\mathbf{Z}_1, \ldots, \mathbf{Z}_n$ being $(L + 2k)$-local PSD ensemble	

Table 2.1: List decoding SOS formulation for $\tilde{y}$.

2.6.3 Properties of the Entropic Proxy

We establish some key properties of our negative entropic function Ψ. First, we show that Ψ is a convex function. Since the feasible set defined by the SOS List Decoding Program is convex and admits an efficient separation oracle,[13] the convexity of Ψ implies that the List Decoding Program can be efficiently solved within η-optimality in time $n^{O(t)} \cdot \mathrm{polylog}(\eta^{-1})$ where t is the SOS degree.

Lemma 2.6.2 (Convexity)**.** Ψ *is convex, i.e., for every pair of pseudo-expectations* $\widetilde{\mathbb{E}}_1$ *and*

13. In our setting the pseudo-expectation has trace bounded by $n^{O(t)}$ in which case semidefinite programming can be solved efficiently [GM12, RW17].

$\widetilde{\mathbb{E}}_2$ *and* $\alpha \in [0,1]$,

$$\Psi\left(\alpha \cdot \widetilde{\mathbb{E}}_1 + (1-\alpha) \cdot \widetilde{\mathbb{E}}_2\right) \;\leq\; \alpha \cdot \Psi\left(\widetilde{\mathbb{E}}_1\right) + (1-\alpha) \cdot \Psi\left(\widetilde{\mathbb{E}}_2\right).$$

Proof. Suppose $\mathfrak{s} \cup \mathfrak{t} = \{i_1, \ldots, i_t\}$. By definition $\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}} = \mathrm{dsum}(\mathbf{Z})_{\mathfrak{s}} \cdot \mathrm{dsum}(\mathbf{Z})_{\mathfrak{t}}$, i.e., $\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}$ is a function f on input $\mathbf{Z}_{i_1}, \ldots, \mathbf{Z}_{i_t} \in \{\pm 1\}$. Let

$$f(\mathbf{Z}_{i_1}, \ldots, \mathbf{Z}_{i_t}) = \sum_{S \subseteq \mathfrak{s} \cup \mathfrak{t}} \widehat{f}(S) \cdot \prod_{i \in S} \mathbf{Z}_i,$$

be the Fourier decomposition of f. Then

$$\widetilde{\mathbb{E}}\left[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}\right] = \widetilde{\mathbb{E}}\left[f\right] = \sum_{S \subseteq \mathfrak{s} \cup \mathfrak{t}} \widehat{f}(S) \cdot \widetilde{\mathbb{E}}\left[\prod_{i \in S} \mathbf{Z}_i\right].$$

Since $\widetilde{\mathbb{E}}\left[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}\right]$ is a linear function of $\widetilde{\mathbb{E}}$, we obtain $\left(\widetilde{\mathbb{E}}\left[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}\right]\right)^2$ is convex. Now, the convexity of Ψ follows by noting that Ψ is a convex combination of convex functions. ∎

The (sole) problem-specific constraint appearing in the SOS List Decoding Program allows us to deduce a lower bound on Ψ. This lower bound will be important later to show that a feasible solution that does not cover all our intended solutions must have Ψ bounded away from 0 so that we still have room to decrease Ψ. We note that an improvement in the conclusion of the following lemma would directly translate to stronger list decoding parameters in our framework.

Lemma 2.6.3 (Correlation ⇒ entropic bound)**.** *Let* $\{\mathbf{Y}_{\mathfrak{s}}\}_{\mathfrak{s} \in X(k)}$ *be* t*-local PSD ensemble with* $t \geq 2$*. If there is some* $y \in \{\pm 1\}^{X(k)}$ *such that*

$$\left|\mathbb{E}_{\mathfrak{s} \sim \Pi_k} \widetilde{\mathbb{E}}\left[y_{\mathfrak{s}} \cdot \mathbf{Y}_{\mathfrak{s}}\right]\right| \geq \beta,$$

then

$$\Psi\left(\{\mathbf{Y}_{\mathfrak{s}}\}_{\mathfrak{s}\in X(k)}\right) \;\geq\; \beta^4.$$

Proof. We calculate

$$\begin{aligned}
\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\left(\widetilde{\mathbb{E}}\left[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}\right]\right)^2 &= \mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\left(\widetilde{\mathbb{E}}\left[\left(y_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{s}}\right)\left(y_{\mathfrak{t}}\mathbf{Y}_{\mathfrak{t}}\right)\right]\right)^2 \\
&\geq \left(\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\widetilde{\mathbb{E}}\left[\left(y_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{s}}\right)\left(y_{\mathfrak{t}}\mathbf{Y}_{\mathfrak{t}}\right)\right]\right)^2 \quad \text{(Jensen's Inequality)} \\
&= \left(\widetilde{\mathbb{E}}\left[\left(\mathbb{E}_{\mathfrak{s}\sim\Pi_k} y_{\mathfrak{s}}\cdot\mathbf{Y}_{\mathfrak{s}}\right)^2\right]\right)^2 \\
&\geq \left(\widetilde{\mathbb{E}}\left[\mathbb{E}_{\mathfrak{s}\sim\Pi_k}\left[y_{\mathfrak{s}}\cdot\mathbf{Y}_{\mathfrak{s}}\right]\right]\right)^4 \quad \text{(Cauchy–Schwarz Inequality)} \\
&= \left(\mathbb{E}_{\mathfrak{s}\sim\Pi_k}\widetilde{\mathbb{E}}\left[y_{\mathfrak{s}}\cdot\mathbf{Y}_{\mathfrak{s}}\right]\right)^4 \;\geq\; \beta^4.
\end{aligned}$$

■

We now show the role of Ψ in list decoding: if an intended solution is not represented in the pseudo-expectation $\widetilde{\mathbb{E}}$, we can get a new pseudo-expectation $\widetilde{\mathbb{E}}'$ which attains a smaller value of Ψ.

Lemma 2.6.4 (Progress lemma)**.** *Suppose there exist* $z \in \{\pm 1\}^{X(1)}$ *and* $y = \mathrm{dsum}(z) \in \{\pm 1\}^{X(k)}$ *satisfying*

$$\widetilde{\mathbb{E}}\left[\left(\mathbb{E}_{\mathfrak{s}\sim\Pi_k} y_{\mathfrak{s}}\cdot\mathbf{Y}_{\mathfrak{s}}\right)^2\right] \leq \delta^2.$$

If $\Psi \geq \delta^2$*, then there exists a pseudo-expectation* $\widetilde{\mathbb{E}}'$ *such that*

$$\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\left(\widetilde{\mathbb{E}}'\left[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}\right]\right)^2 \;\leq\; \Psi - \frac{\left(\Psi-\delta^2\right)^2}{2}.$$

In particular, if $\Psi \geq 2\delta^2$*, then*

$$\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\left(\widetilde{\mathbb{E}}'\left[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}\right]\right)^2 \;\leq\; \Psi - \frac{\delta^4}{2}.$$

Proof. Let $\widetilde{\mathbb{E}}'$ be the pseudo-expectation[14]

$$\widetilde{\mathbb{E}}' \;\coloneqq\; (1-\alpha)\cdot\widetilde{\mathbb{E}} + \alpha\cdot\mathbb{E}_{\delta_z},$$

where $\mathbb{E}_{\delta_z}$ is the expectation of the delta distribution on z and $\alpha \in (0,1)$ is to be defined later. We have

$$\begin{aligned}
\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\left(\widetilde{\mathbb{E}}'\left[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}\right]\right)^2 &= \mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\left((1-\alpha)\cdot\widetilde{\mathbb{E}}\left[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}\right] + \alpha\cdot y_{\mathfrak{s}}y_{\mathfrak{t}}\right)^2 \\
&= (1-\alpha)^2\cdot\Psi + \alpha^2\cdot\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}(y_{\mathfrak{s}}y_{\mathfrak{t}})^2 + 2\alpha(1-\alpha)\cdot\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\left[\widetilde{\mathbb{E}}[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}]y_{\mathfrak{s}}y_{\mathfrak{t}}\right] \\
&\leq (1-\alpha)^2\cdot\Psi + \alpha^2 + 2\alpha(1-\alpha)\cdot\delta^2.
\end{aligned}$$

The value of α minimizing the quadratic expression of the RHS above is

$$\alpha^\star \;=\; \frac{\Psi-\delta^2}{1+\Psi-2\delta^2}.$$

Using this value yields

$$\begin{aligned}
\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}\left(\widetilde{\mathbb{E}}'\left[\mathbf{Y}_{\mathfrak{s}}\mathbf{Y}_{\mathfrak{t}}\right]\right)^2 &\leq \Psi - \frac{\left(\Psi-\delta^2\right)^2}{1+\Psi-2\delta^2} \\
&\leq \Psi - \frac{\left(\Psi-\delta^2\right)^2}{2},
\end{aligned}$$

where in the last inequality we used $\Psi \leq 1$. ∎

2.6.4 Propagation Rounding

A central algorithm in our list decoding framework is the Propagation Rounding Algorithm 2.6.5. It was studied by Barak et al. [BRS11] in the context of approximating 2-CSPs

14. By summing the pseudo-expectation $\widetilde{\mathbb{E}}$ and actual expectation $\mathbb{E}_{\delta_z}$, we mean that we are summing $\widetilde{\mathbb{E}}$ to pseudo-expectation of the same dimensions obtained from operator $\mathbb{E}_{\delta_z}$.

on low threshold rank graphs and it was later generalized to HDXs (and low threshold rank hypergraphs) in the context of k-CSPs [AJT19].

Given an $(L+2k)$-local PSD ensemble $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$, the Propagation Rounding Algorithm 2.6.5 chooses a subset of variables $S \subseteq [n]$ at random. Then it samples a joint assignment σ to the variables in S according to $\{\mathbf{Z}_S\}$. The value of the remaining variables $\mathbf{Z}_i$ are sampled according to the conditional marginal distributions $\{\mathbf{Z}_i | \mathbf{Z}_S = \sigma\}$. An important byproduct of this algorithm is the $2k$-local PSD ensemble $\mathbf{Z}' = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_n | \mathbf{Z}_S = \sigma\}$.

The precise description of the Propagation Rounding Algorithm 2.6.5 follows.

Algorithm 2.6.5 (Propagation Rounding Algorithm)**.**

Input *An $(L+2k)$-local PSD ensemble $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ and some distribution Π_k on $X(k)$.*

Output *A random assignment $(\sigma_1, \ldots, \sigma_n) \in [q]^n$ and $2k$-local PSD ensemble $\mathbf{Z}'$.*

1. *Choose $m \in \{1, \ldots, L/k\}$ uniformly at random.*
2. *Independently sample m k-faces, $\mathfrak{s}_j \sim \Pi_k$ for $j = 1, \ldots, m$.*
3. *Write $S = \bigcup_{j=1}^{m} \mathfrak{s}_j$, for the set of the seed vertices.*
4. *Sample assignment $\sigma : S \to [q]$ according to the local distribution $\{\mathbf{Z}_S\}$.*
5. *Set $\mathbf{Z}' = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_n | \mathbf{Z}_S = \sigma\}$, i.e. the local ensemble $\mathbf{Z}$ conditioned on agreeing with σ.*
6. *For all $j \in [n]$, sample independently $\sigma_j \sim \{\mathbf{Z}'_j\}$.*
7. *Output $(\sigma_1, \ldots, \sigma_n)$ and $\mathbf{Z}'$.*

To our list decoding task we will show that an ensemble minimizing Ψ covers the space of possible solutions in the sense that for any intended solution there will be a choice of S and σ such that the conditioned ensemble $\mathbf{Z}'$ enables the sampling of a word within the unique

decoding radius in $\mathcal{C}_1$ of this intended solution.

An execution of the Algorithm 2.6.5 is completely determined by the tuple (m, S, σ) which we will refer to as a slice of the PSD ensemble.

Definition 2.6.6 (Slice)**.** *We call a tuple (m, S, σ) obtainable by Algorithm 2.6.5 a slice and let Ω denote the set of all slices obtainable by Algorithm 2.6.5.*

We can endow Ω with a natural probability distribution, where the measure of each (m, S, σ) is defined as the probability that this slice is picked during an execution of Algorithm 2.6.5. We also define a pseudo-expectation operator for each slice.

Definition 2.6.7 (Pseudo-Expectation Slice)**.** *Given a slice (m, S, σ), we define the pseudo-expectation operator $\widetilde{\mathbb{E}}_{|_{S,\sigma}}$ which is the pseudo-expectation operator of the conditioned local PSD ensemble $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_n | \mathbf{Z}_s = \sigma\}$.*

2.6.5 Tensorial Structures

In general, a local PSD ensemble $\mathbf{Z}' = \{\mathbf{Z}'_1, \ldots, \mathbf{Z}'_n\}$ output by the Propagation Rounding Algorithm 2.6.5 may be far from corresponding to any underlying joint global distribution.[15] In our application, we will be interested in the case where the ensemble approximately behaves as being composed of independent random variables over the collection of "local views" given by the hyperedges in $X(k)$. In such case, rounding the SOS solution via independent rounding is straightforward. A collection of local views admitting this property with a given SOS degree parameter L is denoted *tensorial* (variables behave as products over the local views).

Definition 2.6.8 (Tensorial Hypergraphs)**.** *Let $X(k)$ be a collection of k-uniform hyperedges endowed with a distribution Π_k, $\mu \in [0, 1]$, and $L \in \mathbb{N}$. We say that $X(k)$ is (μ, L)-tensorial*

15. In fact, if this was the case, then we would be able to approximate any k-CSP with SOS degree $(L+2k)$. However, even for L as large as linear in n this is impossible for SOS [Gri01, KMOW17].

if the local PSD ensemble $\mathbf{Z}'$ *returned by Propagation Rounding Algorithm 2.6.5 with SOS degree parameter* L *satisfies*

$$\mathop{\mathbb{E}}_{\Omega} \mathop{\mathbb{E}}_{\mathfrak{a} \sim \Pi_k} \left\| \{\mathbf{Z}'_{\mathfrak{a}}\} - \left\{\mathbf{Z}'_{a_1}\right\} \cdots \left\{\mathbf{Z}'_{a_k}\right\} \right\|_1 \leq \mu. \tag{2.1}$$

To analyze the potential Ψ we will need that the variables between pairs of local views, i.e., pairs of hyperedges, behave as product.

Definition 2.6.9 (Two-Step Tensorial Hypergraphs)**.** *Let* $X(k)$ *be a collection of* k*-uniform hyperedges endowed with a distribution* Π_k*,* $\mu \in [0,1]$*, and* $L \in \mathbb{N}$*. We say that* $X(k)$ *is* (μ, L)*-two-step tensorial if it is* (μ, L)*-tensorial and the PSD ensemble* $\mathbf{Z}'$ *returned by Propagation Rounding Algorithm 2.6.5 with SOS degree parameter* L *satisfies*

$$\mathop{\mathbb{E}}_{\Omega} \mathop{\mathbb{E}}_{\mathfrak{s},\mathfrak{t} \sim \Pi_k} \left\| \{\mathbf{Z}'_{\mathfrak{s}} \mathbf{Z}'_{\mathfrak{t}}\} - \left\{\mathbf{Z}'_{\mathfrak{s}}\right\}\left\{\mathbf{Z}'_{\mathfrak{t}}\right\} \right\|_1 \leq \mu.$$

In Section 2.7.1, we establish the relationship between the parameters μ and L and the expansion that will ensure HDXs are (μ, L)-two-step tensorial. Similarly, in Section 2.9.1 we provide this relationship when $X(k)$ is the collection of walks of an expander graph.

Tensorial over Most Slices

By choosing μ sufficiently small it is easy to show that most slices (m, S, σ) satisfy the tensorial (or two-step tensorial) statistical distance condition(s) with a slightly worse parameter $\tilde{\mu}$ such that $\tilde{\mu} \to 0$ as $\mu \to 0$. If we could construct tensorial (or two-step tensorial) objects for arbitrarily small parameter μ with $L = O_{k,q,\mu}(1)$, then we would be able to obtain $\tilde{\mu}$ arbitrarily small. Lemma 2.7.4 establishes that HDXs of appropriate expansion satisfy this assumption, and Lemma 2.9.20 does the same for walks on expanders.

We introduce two events. The first event captures when a slice (m, S, σ) leads to the conditioned local variables $\mathbf{Z}'_1, \ldots, \mathbf{Z}'_n$ being close to k-wise independent over the k-sized

hyperedges.

Definition 2.6.10 (Ground Set Close to k-wise Independent)**.** *Let $\mu \in (0,1]$. We define the event K_μ as*

$$K_\mu \coloneqq \left\{(m,S,\sigma) \in \Omega \mid \mathop{\mathbb{E}}_{\mathfrak{a}\sim\Pi_k} \left\| \{\mathbf{Z}_{\mathfrak{a}}|\mathbf{Z}_S=\sigma\} - \{\mathbf{Z}_{a_1}|\mathbf{Z}_S=\sigma\}\cdots\left\{\mathbf{Z}_{a_k}|\mathbf{Z}_S=\sigma\right\}\right\|_1 < \mu^2/2\right\}.$$

The second event captures when the variables between pairs of hyperedges are close to independent.

Definition 2.6.11 (Lifted Variables Close to Pairwise Independent)**.** *Let $\mu \in (0,1]$. We define the event P_μ as*

$$P_\mu \coloneqq \left\{(m,S,\sigma) \in \Omega \mid \mathop{\mathbb{E}}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k} \|\{\mathbf{Z}_{\mathfrak{s}}\mathbf{Z}_{\mathfrak{t}}|\mathbf{Z}_S=\sigma\} - \{\mathbf{Z}_{\mathfrak{s}}|\mathbf{Z}_S=\sigma\}\{\mathbf{Z}_{\mathfrak{t}}|\mathbf{Z}_S=\sigma\}\|_1 < \mu^2/2\right\}.$$

These events satisfy a simple concentration property.

Claim 2.6.12 (Concentration)**.** *Suppose a simplicial complex $X(\leq k)$ with $X(1)=[n]$ and an $(L+2k)$-local PSD ensemble $\mathbf{Z}=\{\mathbf{Z}_1,\ldots,\mathbf{Z}_n\}$ are given as input to Propagation Rounding Algorithm 2.6.5. Let $\mu \in (0,1]$. If $X(k)$ is $(\mu^4/4, L)$-two-step tensorial, then*

$$\mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[K_\mu^c\right] \leq \frac{\mu^2}{2}, \tag{2.2}$$

and

$$\mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[P_\mu^c\right] \leq \frac{\mu^2}{2}. \tag{2.3}$$

Proof. We only prove Eq. (2.2) since the proof of Eq. (2.3) is similar. Define the random variable $\mathbf{R} \coloneqq \mathbb{E}_{\mathfrak{a}\sim\Pi_k}\left\|\{\mathbf{Z}'_{\mathfrak{a}}\} - \left\{\mathbf{Z}'_{a_1}\right\}\cdots\left\{\mathbf{Z}'_{a_k}\right\}\right\|_1$ on the sample space $\Omega=\{(m,S,\sigma)\}$. From our $(\mu^4/4, L)$-two-step tensorial assumption we have

$$\mathbb{E}_\Omega[\mathbf{R}] \leq \frac{\mu^4}{4}.$$

Now, we can conclude

$$\mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[K_\mu^c\right] = \mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[\mathbf{R} \geq \frac{\mu^2}{2}\right] \leq \frac{\mu^2}{2},$$

using Markov's inequality. ∎

2.6.6 Further Building Blocks and Analysis

Before we delve into further phases of the list decoding framework, we introduce some notation for the list of codewords we want to retrieve.

Definition 2.6.13 (Code list)**.** *Given $\tilde{y} \in \{\pm 1\}^{X(k)}$ and a code $\mathcal{C}$ on $X(k)$ with relative distance at least $1/2 - \varepsilon$, we define the list $\mathcal{L}(\tilde{y}, \mathcal{C})$ as*

$$\mathcal{L}(\tilde{y}, \mathcal{C}) \coloneqq \left\{ y \in \mathcal{C} \mid \Delta(y, \tilde{y}) \leq \frac{1}{2} - \sqrt{\varepsilon} \right\}.$$

Under these assumptions the Johnson bound establishes that the list size is constant whenever $\varepsilon > 0$ is constant.

Remark 2.6.14. *The Johnson bound [GRS19] guarantees that*

$$|\mathcal{L}(\tilde{y}, \mathcal{C})| \leq \frac{1}{2 \cdot \varepsilon}$$

provided the relative distance of $\mathcal{C}$ is at least $1/2 - \varepsilon$.

In the case of lifted codes, it is more appropriate to consider a list of pairs $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ defined as follows.

Definition 2.6.15 (Coupled code list)**.** *Given $\tilde{y} \in \{\pm 1\}^{X(k)}$ and a lifted code $\mathcal{C}_k$ on $X(k)$ with relative distance at least $1/2 - \varepsilon$, we define the coupled code list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ as*

$$\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k) \coloneqq \left\{ (z, \mathrm{dsum}(z)) \mid z \in \mathcal{C}_1 \quad \textit{and} \quad \Delta(\mathrm{dsum}(z), \tilde{y}) \leq \frac{1}{2} - \sqrt{\varepsilon} \right\}.$$

Recovering this list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ is the main goal of this section. This task will be accomplished by Algorithm 2.6.16 stated below whose building blocks and analysis we develop in this section.

Algorithm 2.6.16 (List Decoding Algorithm)**.**

Input *A word $\tilde{y} \in \{\pm 1\}^{X(k)}$ which is $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k = \mathrm{dsum}(\mathcal{C}_1)$.*

Output *Coupled code list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$.*

1. *Solve the List Decoding Program with η-accuracy, obtaining $\mathbf{Z}$, where $\eta = \varepsilon^8/2^{22}$*
2. *Let $\mathcal{M}$ be the output of the Cover Retrieval Algorithm 2.6.29 on $\mathbf{Z}$*
3. *Let $\mathcal{L}'$ be the output of the Cover Purification Algorithm 2.6.36 on $\mathcal{M}$*
4. *Let $\mathcal{L}'' = \{(z, y) \in \mathcal{L}' \mid \Delta(\tilde{y}, y) \leq 1/2 - \sqrt{\varepsilon}\}$*
5. *Output $\mathcal{L}''$*

As shown in Fig. 2.1 of Section 2.3, the first step is to solve the List Decoding Program which results in a pseudo-expectation "covering" the list $\mathcal{L}(\tilde{y}, \mathcal{C})$ as we will make precise. A precursor property to covering and some considerations about SOS rounding are treated in Section 2.6.6. Next, the formal definition of cover is presented in Section 2.6.6 and we have all the elements to present the Cover Retrieval Algorithm 2.6.29 with its correctness in Section 2.6.6. Then, we use the *robustness* properties of the lifting to purify the cover in Section 2.6.6. Finally, in Section 2.6.6, we assemble the building blocks and prove the main technical result, Theorem 2.6.17, whose proof follows easily once the properties of the building blocks are in place.

Note that Theorem 2.6.17 embodies an abstract list decoding framework which relies only on the *robustness* and *tensorial* properties of the lifting. We provide a concrete instantiation of the framework to the direct sum lifting on HDXs in Section 2.7 and to the direct sum

lifting on expander walks in Section 2.9.2.

Theorem 2.6.17 (List Decoding Theorem)**.** *Suppose that* dsum *is a* $(1/2-\varepsilon_0, 1/2-\varepsilon)$*-robust* $(\varepsilon^8/2^{22}, L)$*-two-step tensorial lifting from* $\mathcal{C}_1$ *to* $\mathcal{C}_k$ *which is either*

- *linear and a* $(1/2+\varepsilon_0, 2\cdot\varepsilon)$*-parity sampler; or*
- $(1/4-\varepsilon_0, 1/2-\varepsilon/2)$*-robust and odd.*

Let $\tilde{y} \in \{\pm 1\}^{X(k)}$ *be* $(1/2-\sqrt{\varepsilon})$*-close to* $\mathcal{C}_k$*. Then w.v.h.p. the List Decoding Algorithm 2.6.16 returns the coupled code list* $\mathcal{L}(\widetilde{y}, \mathcal{C}_1, \mathcal{C}_k)$*. Furthermore, the running time is*

$$n^{O(L+k)}\left(\text{polylog}(\varepsilon^{-1}) + f(n)\right),$$

where $n = |X(1)|$ *and* $f(n)$ *is the running time of a unique decoding algorithm of* $\mathcal{C}_1$*.*

Remark 2.6.18. *Regarding Theorem 2.6.17, we stress that although the lifting is* $(1/2-\varepsilon_0, 1/2-\varepsilon)$*-robust and we can perform list decoding at least up to distance* $1/2-\sqrt{\varepsilon}$*, our framework does not recover the Johnson bound. The issue is that our framework requires one of the additional amplification guarantees of Theorem 2.6.17, which both make the distance of* $\mathcal{C}_k$ *become* $1/2-\varepsilon^{\Omega_{\varepsilon_0}(1)} > 1/2-\varepsilon$*. Efficiently recovering the Johnson bound remains an interesting open problem.*

We observe that the algorithms themselves used in this framework are quite simple (although their analyses might not be). Moreover, the tasks of cover retrieval and purification are reasonably straightforward. However, Section 2.6.6 combines *tensorial* properties of the lifting with properties of Ψ, requiring a substantial analysis. The list decoding framework is divided into stages to make it modular so that key properties are isolated and their associated functionality can be presented in a simple manner. Most of the power of this framework comes from the combination of these blocks and the concrete expanding objects capable of instantiating it.

SOS Rounding and Recoverability

We show that if a slice (m, S, σ) "captures" an intended solution $y \in \{\pm 1\}^{X(k)}$ (this notion is made precise in the assumptions of Lemma 2.6.20), then we can retrieve a $z \in \{\pm 1\}^{X(1)}$ such that $\mathrm{dsum}(z)$ has some agreement with y. This agreement is somewhat weak, but combined with the robustness of the lifting, it will be enough for our purposes. In this subsection, we first explore how to recover such words within a slice, which can be seen as local rounding in the slice. Next, we establish sufficient conditions for an intended solution to be recoverable, now not restricted to a given slice but rather with respect to the full pseudo-expectation. Finally, we use all the tools developed so far to show that by minimizing Ψ in a two-step tensorial structure we end up with a pseudo-expectation in which all intended solutions are recoverable. The interplay between weak agreement and robustness of the lifting is addressed in Section 2.6.6.

We will be working with two-step tensorial structures where the following product distribution associated to a slice naturally appears.

Definition 2.6.19 (Product Distribution on a Slice)**.** *We define* $\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}$ *to be the product distribution on the marginals* $\{\mathbf{Z}_i|\mathbf{Z}_S = \sigma\}_{i\in X(1)}$*, i.e.,* $\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\} \coloneqq \prod_{i\in X(1)}\{\mathbf{Z}_i|\mathbf{Z}_S = \sigma\}$.

Under appropriate conditions, Lemma 2.6.20 shows how to round the pseudo-expectation in a slice.

Lemma 2.6.20 (From fractional to integral in a slice)**.** *Let* $(m, S, \sigma) \in \Omega$ *be a slice. Suppose*

$$\mathop{\mathbb{E}}_{\mathfrak{a}\sim\Pi_k} \left\|\{\mathbf{Z}_{\mathfrak{a}}|\mathbf{Z}_S = \sigma\} - \{\mathbf{Z}_{a_1}|\mathbf{Z}_S = \sigma\}\cdots\left\{\mathbf{Z}_{a_k}|\mathbf{Z}_S = \sigma\right\}\right\|_1 \leq \mu, \tag{2.4}$$

and

$$\mathop{\mathbb{E}}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2} \|\{\mathbf{Z}_{\mathfrak{s}}\mathbf{Z}_{\mathfrak{t}}|\mathbf{Z}_S = \sigma\} - \{\mathbf{Z}_{\mathfrak{s}}|\mathbf{Z}_S = \sigma\}\{\mathbf{Z}_{\mathfrak{t}}|\mathbf{Z}_S = \sigma\}\|_1 \leq \mu. \tag{2.5}$$

For $\beta \in (0,1)$, if $\mu \le \beta \cdot \kappa^2/6$ and $y \in \{\pm 1\}^{X(k)}$ is such that

$$\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2} \widetilde{\mathbb{E}}_{|S,\sigma} \left[y_{\mathfrak{s}} y_{\mathfrak{t}} \mathbf{Y}_{\mathfrak{s}} \mathbf{Y}_{\mathfrak{t}} \right] \ge \kappa^2,$$

then

$$\mathop{\mathbb{P}}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} \left[\left| \mathbb{E}_{\mathfrak{s}\sim\Pi_k} y_{\mathfrak{s}} \cdot \mathrm{dsum}(z)_{\mathfrak{s}} \right| \ge \sqrt{1-\beta} \cdot \kappa \right] \ge \frac{\beta \cdot \kappa^2}{4}. \tag{2.6}$$

Proof. Let $\mu_{\mathfrak{s},\mathfrak{t}} \coloneqq \| \{\mathbf{Z}_{\mathfrak{s}} \mathbf{Z}_{\mathfrak{t}} | \mathbf{Z}_s = \sigma\} - \prod_{i\in\mathfrak{s}} \{\mathbf{Z}_i | \mathbf{Z}_s = \sigma\} \prod_{i\in\mathfrak{t}} \{\mathbf{Z}_i | \mathbf{Z}_s = \sigma\} \|_1$. Using triangle inequality and simplifying, we get

$$\begin{aligned} \mu_{\mathfrak{s},\mathfrak{t}} \le & \| \{\mathbf{Z}_{\mathfrak{s}} \mathbf{Z}_{\mathfrak{t}} | \mathbf{Z}_s = \sigma\} - \{\mathbf{Z}_{\mathfrak{s}} | \mathbf{Z}_s = \sigma\} \{\mathbf{Z}_{\mathfrak{t}} | \mathbf{Z}_s = \sigma\} \|_1 \\ & + \left\| \{\mathbf{Z}_{\mathfrak{s}} | \mathbf{Z}_s = \sigma\} - \prod_{i\in\mathfrak{s}} \{\mathbf{Z}_i | \mathbf{Z}_s = \sigma\} \right\|_1 + \left\| \{\mathbf{Z}_{\mathfrak{t}} | \mathbf{Z}_s = \sigma\} - \prod_{i\in\mathfrak{t}} \{\mathbf{Z}_i | \mathbf{Z}_s = \sigma\} \right\|_1 . \end{aligned}$$

From our assumptions Eq. (2.4) and Eq. (2.5), it follows that $\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2} \mu_{\mathfrak{s},\mathfrak{t}} \le 3 \cdot \mu$. Using the fact that $|y_{\mathfrak{s}} y_{\mathfrak{t}}| = 1$ and Hölder's inequality, we get

$$\begin{aligned} \mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2} \mathbb{E}_{\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} \left[y_{\mathfrak{s}} y_{\mathfrak{t}} \mathbf{Y}_{\mathfrak{s}} \mathbf{Y}_{\mathfrak{t}} \right] &\ge \mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2} \widetilde{\mathbb{E}}_{|S,\sigma} \left[y_{\mathfrak{s}} y_{\mathfrak{t}} \mathbf{Y}_{\mathfrak{s}} \mathbf{Y}_{\mathfrak{t}} \right] - \mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2} \mu_{\mathfrak{s},\mathfrak{t}} \\ &\ge \mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2} \widetilde{\mathbb{E}}_{|S,\sigma} \left[y_{\mathfrak{s}} y_{\mathfrak{t}} \mathbf{Y}_{\mathfrak{s}} \mathbf{Y}_{\mathfrak{t}} \right] - 3 \cdot \mu \ge \left(1 - \frac{\beta}{2} \right) \cdot \kappa^2. \end{aligned}$$

Alternatively,

$$\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2} \mathbb{E}_{\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} \left[y_{\mathfrak{s}} y_{\mathfrak{t}} \mathbf{Y}_{\mathfrak{s}} \mathbf{Y}_{\mathfrak{t}} \right] = \mathbb{E}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} \left(\mathbb{E}_{\mathfrak{s}\sim\Pi_k} y_{\mathfrak{s}} \cdot \mathrm{dsum}(z)_{\mathfrak{s}} \right)^2 \ge \left(1 - \frac{\beta}{2} \right) \cdot \kappa^2.$$

Define the random variable $\mathbf{R} \coloneqq \left(\mathbb{E}_{\mathfrak{s}\sim\Pi_k} [y_{\mathfrak{s}} \cdot \mathrm{dsum}(z)_{\mathfrak{s}}] \right)^2$. Using Fact 2.10.1 with approximation parameter $\beta/2$, we get

$$\mathbb{E}[\mathbf{R}] \ge \left(1 - \frac{\beta}{2} \right) \cdot \kappa^2 \Rightarrow \Pr \left[\mathbf{R} \ge (1-\beta) \cdot \kappa^2 \right] \ge \frac{\beta \cdot \kappa^2}{4},$$

from which Eq. (2.6) readily follows. ∎

To formalize the notion of a word being recoverable with respect to the full pseudo-expectation rather than in a given slice we will need two additional events. The first event captures correlation as follows.

Definition 2.6.21 (y-Correlated Event)**.** *Let $\kappa \in (0,1]$ and $y \in \{\pm 1\}^{X(k)}$. We define the event $C_\kappa(y)$ as*

$$C_\kappa(y) \coloneqq \left\{ (m,S,\sigma) \in \Omega \mid \mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2} \widetilde{\mathbb{E}}_{|S,\sigma} \left[y_\mathfrak{s} y_\mathfrak{t} \mathbf{Y}_\mathfrak{s} \mathbf{Y}_\mathfrak{t} \right] \geq \kappa^2 \right\}.$$

The second event is a restriction of the first where we also require the slice to satisfy the two-step tensorial condition from Definition 2.6.9.

Definition 2.6.22 (y-Recoverable Event)**.** *Let $\kappa, \mu \in (0,1]$ and $y \in \{\pm 1\}^{X(k)}$. We define the event $R_{\kappa,\mu}(y)$ as*

$$R_{\kappa,\mu}(y) \coloneqq K_\mu \cap P_\mu \cap C_\kappa(y).$$

Lemma 2.6.20 motivates the following "recoverability" condition.

Definition 2.6.23 (Recoverable Word)**.** *Let $\kappa, \mu \in (0,1]$ and $y \in \{\pm 1\}^{X(k)}$. We say that y is (κ,μ)-recoverable provided*

$$\mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega} \Big[R_{\kappa,\mu}(y) \Big] > 0.$$

One of the central results in our framework is the following "recoverability" lemma. It embodies the power SOS brings to our framework.

Lemma 2.6.24 (Recoverability lemma)**.** *Let $\mathcal{C}_k$ be a lifted code on $X(\leq k)$ with $X(1) = [n]$ and distance at least $1/2 - \varepsilon$. Let $\tilde{y} \in \{\pm 1\}^{X(k)}$ be a word promised to be $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$ and let $\mathcal{L} = \mathcal{L}(\tilde{y}, \mathcal{C}_k)$ be its code list.*

Let $\theta \in (0,1]$ be arbitrary and set $\mu = \kappa \cdot \theta/2$ and $\kappa = (4-\theta)\cdot\varepsilon$. Suppose $\mathbf{Z} = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ is an $(L+2k)$-local PSD ensemble which is a solution to the List Decoding Program with objective value Ψ within η additive value from the optimum where $0 \leq \eta \leq \theta^2 \cdot \varepsilon^4$.

If $X(k)$ is $(\mu^4/4, L)$-two-step tensorial, then every $y \in \mathcal{L}$ is (κ, μ)-recoverable. In particular, for every $\theta \in (0,1)$ and under the preceding assumptions, we have that every $y \in \mathcal{L}$ is $((4-\theta)\cdot\varepsilon, \theta)$-recoverable.

Proof. First observe that since $\tilde{y}$ is $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$ the List Decoding Program is feasible and so the solution $\mathbf{Z}$ is well defined. Towards a contradiction with the η-optimality of the SOS solution $\mathbf{Z}$, suppose there exists a word $y \in \mathcal{L}$ that is not (κ, μ)-recoverable. Let $z \in \{\pm 1\}^{X(1)}$ be such that $y = \mathrm{dsum}(z)$. Then

$$1 \;=\; \mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[R_{\kappa,\mu}(y)^c\right] \;\leq\; \mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[K_\mu^c\right] + \mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[P_\mu^c\right] + \mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[C_\kappa(y)^c\right].$$

Using Claim 2.6.12, we get

$$\mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[C_\kappa(y)^c\right] \;\geq\; 1-\mu^2. \tag{2.7}$$

Since $\widetilde{\mathbb{E}}$ is a valid solution to the List Decoding Program, Lemma 2.6.3 implies the lower bound

$$\Psi\left(\{\mathbf{Y}_\mathfrak{s}\}_{\mathfrak{s}\in X(k)}\right) \;\geq\; 16\cdot\varepsilon^2. \tag{2.8}$$

By definition, for $(m,S,\sigma) \in C_\kappa(y)^c$ we have

$$\mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k^2}\widetilde{\mathbb{E}}_{|S,\sigma}\left[y_\mathfrak{s} y_\mathfrak{t}\mathbf{Y}_\mathfrak{s}\mathbf{Y}_\mathfrak{t}\right] \leq \kappa^2,$$

implying

$$\widetilde{\mathbb{E}}\left[\left(\mathbb{E}_{\mathfrak{s}\sim\Pi_k} y_\mathfrak{s}\cdot\mathbf{Y}_\mathfrak{s}\right)^2\right] \leq \mathbb{E}_{m,S,\sigma}\widetilde{\mathbb{E}}_{|S,\sigma}\left[\left(\mathbb{E}_{\mathfrak{s}\sim\Pi_k} y_\mathfrak{s}\cdot\mathbf{Y}_\mathfrak{s}\right)^2\cdot\mathbf{1}_{C^c_{\kappa(y)}}\right] + \mathop{\mathbb{P}}_{(m,S,\sigma)\sim\Omega}\left[C_\kappa(y)\right] \leq \kappa^2+\mu^2.$$

Let $\widetilde{\mathbb{E}}$ be the pseudo-expectation of the ground set ensemble $\mathbf{Z}$ and let $\mathbb{E}'$ be the expectation on the delta distribution δ_z. Note that the pseudo-expectation obtained from $\mathbb{E}'$ is a valid solution to the List Decoding Program. Since

$$\kappa^2 + \mu^2 \leq \left(1 + \frac{\theta^2}{4}\right) \cdot \kappa^2 = \left(1 + \frac{\theta^2}{4}\right) \cdot (4 - \theta)^2 \cdot \varepsilon^2 \leq (16 - 2 \cdot \theta) \cdot \varepsilon^2,$$

and $\theta \geq 0$, Lemma 2.6.4 gives that there is a convex combination of $\widetilde{\mathbb{E}}$ and $\mathbb{E}'$ such that the new Ψ, denoted Ψ', can be bounded as

$$\Psi' \;\leq\; \Psi - \frac{\left(\Psi - \left(\kappa^2 + \mu^2\right)\right)^2}{2} \leq \Psi - 2 \cdot \theta^2 \cdot \varepsilon^4,$$

contradicting the η-optimality of the SOS solution $\mathbf{Z}$ since $\eta \leq \theta^2 \cdot \varepsilon^4$. ∎

Coupled Pairs, Coupled Lists, and Covers

The List Decoding Program minimizing Ψ was instrumental to ensure that every $y' \in \mathcal{L}(\tilde{y}, \mathcal{C}_k)$ is recoverable in the sense of the conclusion of Lemma 2.6.24. Unfortunately, this guarantee is somewhat weak, namely, associated to every $y' \in \mathcal{L}(\tilde{y}, \mathcal{C}_k)$ there is a slice (m, S, σ) from which we can sample y (our approximation of y') satisfying

$$|\mathbb{E}_{\mathfrak{s} \sim \Pi_k} y_{\mathfrak{s}} \cdot y'_{\mathfrak{s}}| > C \cdot \varepsilon, \tag{2.9}$$

where C is a constant strictly smaller than 4. A priori this seems insufficient for our list decoding task. However, there are two properties which will help us with list decoding. The first is that SOS finds not only y but also $z \in \{\pm 1\}^{X(1)}$ such that $y = \mathrm{dsum}(z)$. The second property is that the lifting is robust: even the weak agreement given by Eq. (2.9) translates into a much stronger agreement in the ground set between z and $z' \in \mathcal{C}_1$ where $y' = \mathrm{dsum}(z')$. This stronger agreement on the ground set can be used to ensure that z (or

$-z$) lies inside the unique decoding ball of z' in the base code $\mathcal{C}_1$.

To study this coupling phenomenon between words in the lifted space $\{\pm 1\}^{X(k)}$ and on the ground space $\{\pm 1\}^{X(1)}$ we introduce some terminology. The most fundamental one is a coupled pair.

Definition 2.6.25 (Coupled Pair)**.** *Let $z \in \{\pm 1\}^{X(1)}$ and $y \in \{\pm 1\}^{X(k)}$. We say that (z, y) is a coupled pair with respect to a lift function* dsum *provided $y = \mathrm{dsum}(z)$.*

Remark 2.6.26. *If the function* dsum *is clear in the context, we may assume that the coupled pair is with respect to this function.*

Coupled pairs can be combined in a list.

Definition 2.6.27 (Coupled List)**.** *We say that a list $\mathcal{M} = \{(z^{(1)}, y^{(1)}), \ldots, (z^{(h)}, y^{(h)})\}$ is coupled with respect to lift function* dsum *provided $(z^{(i)}, y^{(i)})$ is a coupled pair for every i in $[h]$.*

A coupled list can "cover" a list of words in the lifted space $\{\pm 1\}^{X(k)}$ as defined next.

Definition 2.6.28 (Coupled Bias Cover)**.** *Let $\mathcal{M} = \{(z^{(1)}, y^{(1)}), \ldots, (z^{(h)}, y^{(h)})\}$ be a coupled list and $\mathcal{L} \subset \{\pm 1\}^{X(k)}$. We say that $\mathcal{M}$ is a δ-bias cover of $\mathcal{L}$ provided*

$$\left(\forall y' \in \mathcal{L}\right) \left(\exists (z, y) \in \mathcal{M}\right) \left(\left|\mathbb{E}_{\mathfrak{s} \sim \Pi_k} y'_{\mathfrak{s}} \cdot y_{\mathfrak{s}}\right| > \delta\right).$$

A δ-bias cover for "small" δ might seem a rather weak property, but as alluded to, when combined with enough robustness of the lifting, it becomes a substantial guarantee enabling list decoding.

Cover Retrieval

When the code list $\mathcal{L}(\tilde{y}, \mathcal{C}_k)$ becomes recoverable in the SOS sense as per Lemma 2.6.24, we still need to conduct local rounding on the slices to collect a bias cover. Recall that this

local rounding is probabilistic (c.f. Lemma 2.6.20), so we need to repeat this process a few times to boost our success probability.[16] This is accomplished by Algorithm 2.6.29.

Algorithm 2.6.29 (Cover Retrieval Algorithm)**.**

Input *An $(L+2k)$-local PSD ensemble $\mathbf{Z}$ which is a $(\theta^2\varepsilon^4)$-optimal solution to the List Decoding Program.*

Output *A 2ε-bias cover $\mathcal{M}$ for $\mathcal{L}(\tilde{y}, \mathcal{C}_k)$.*

1. *Let $\mathcal{M} = \emptyset$*
2. *Let $T = 4 \cdot \ln(|\Omega|) \cdot n/(\beta \cdot \varepsilon^2)$*
3. *For $(m, S, \sigma) \in \Omega$ do*
4. *If $(m, S, \sigma) \in K_\mu \cap P_\mu$ then*
5. *Run Propagation Rounding T times conditioned on (m, S, σ)*
6. *Let $\mathcal{M}|_{m,S,\sigma} = \{(z^{(1)}, y^{(1)}), \ldots, (z^{(T)}, y^{(T)})\}$ be the coupled list*
7. *Set $\mathcal{M} = \mathcal{M} \cup \mathcal{M}|_{m,S,\sigma}$*
8. *Output $\mathcal{M}$.*

The correctness of Algorithm 2.6.29 follows easily given the properties established so far.

Lemma 2.6.30 (Cover lemma)**.** *Let $\beta \in (0,1)$. Suppose that* dsum *is a $(1/2 - \varepsilon_0, 1/2 - \varepsilon)$-robust $(\beta^4 \cdot \varepsilon^8/2^{18}, L)$-two-step tensorial lifting from $\mathcal{C}_1$ to $\mathcal{C}_k$. Let $\tilde{y} \in \{\pm 1\}^{X(k)}$ be $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$. If $\theta \leq \beta \cdot \varepsilon/2^4$, then w.v.h.p.[17] the Cover Retrieval algorithm 2.6.29 returns a δ-bias cover $\mathcal{M}$ of the code list $\mathcal{L}(\tilde{y}, \mathcal{C}_k)$ where $\delta = (4 - \beta) \cdot \varepsilon$. Furthermore, the running*

16. In fact, this process can be derandomized using standard techniques in our instantiations. See Lemma 2.10.5 for details.

17. The abbreviation w.v.h.p. stands for *with very high probability* and means with probability $1 - \exp(-\Theta(n))$.

time is at most $n^{O(L+k)}/(\beta \cdot \varepsilon^2)$ *where* $n = |X(1)|$.

Proof. Let $\mathbf{Z} = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ be an η-optimum solution to the List Decoding Program where $\eta \leq \theta^2 \cdot \varepsilon^4$ and $\theta = \beta \cdot \varepsilon / 2^4$. By our $(\beta^4 \cdot \varepsilon^8 / 2^{18}, L)$-two-step tensorial assumption and our choice of SOS degree for the List Decoding Program, we can apply Lemma 2.6.24 to conclude that every $y \in \mathcal{L} = \mathcal{L}(\tilde{y}, \mathcal{C}_k)$ is $((4-\theta)\cdot\varepsilon, (4-\theta)\cdot\varepsilon\cdot\theta/2)$-recoverable. Then for $y \in \mathcal{L}$, there exists $(m, S, \sigma) \in \Omega$ such that Lemma 2.6.20 yields

$$\mathop{\mathbb{P}}_{z \sim \{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} \left[\left| \mathbb{E}_{\mathfrak{s} \sim \Pi_k} y_{\mathfrak{s}} \cdot \mathrm{dsum}(z) \right| \geq (4-\beta)\cdot\varepsilon \right] \geq \frac{\beta \cdot (4-\theta)^2 \cdot \varepsilon^2}{32} \geq \frac{\beta \cdot \varepsilon^2}{4}.$$

where $\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}$ (c.f. Definition 2.6.19) is the product distribution of the marginal distributions after conditioning the ensemble on slice (m, S, σ). By sampling $\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}$ independently T times we obtain $z^{(1)}, \ldots, z^{(T)}$ and thus also the coupled list

$$\mathcal{M}|_{m,S,\sigma} = \{(z^{(1)}, y^{(1)}), \ldots, (z^{(T)}, y^{(T)})\},$$

where $y^{(i)} = \mathrm{dsum}(z^{(i)})$. Then

$$\begin{aligned} \mathop{\mathbb{P}}_{z^{(1)},\ldots,z^{(T)} \sim \{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}^{\otimes T}} \left[\forall i \in [T] : \left| \mathbb{E}_{\mathfrak{s} \sim \Pi_k} y_{\mathfrak{s}} \cdot \mathrm{dsum}\left(z^{(i)}\right) \right| < (4-\beta)\cdot\varepsilon \right] &\leq \exp\left(-\frac{\beta \cdot \varepsilon^2 \cdot T}{4} \right) \\ &\leq \frac{\exp(-n)}{|\Omega|}, \end{aligned}$$

where the last inequality follows from our choice of T. Then by union bound

$$\mathbb{P}\left[\mathcal{M} \text{ is not a } 2\varepsilon\text{-bias cover of } \mathcal{L}\right] \leq |\mathcal{L}| \cdot \frac{\exp(-n)}{|\Omega|} \leq \exp(-n),$$

concluding the proof. ∎

Cover Purification and Robustness

Now we consider the third and final stage of the list decoding framework. We show how despite the weak guarantee of the bias cover returned by the Cover Retrieval Algorithm 2.6.29 we can do a further processing to finally obtain the coupled code list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ provided the lifting admits some *robustness* properties. We first develop these properties and later present this process, denoted Cover Purification.

Further Lifting Properties

Given two coupled pairs $(z, y = \mathrm{dsum}(z))$ and $(z', y' = \mathrm{dsum}(z'))$ (where $z \in \mathcal{C}_1$), we show how weak agreement between y and y' on the lifted space is enough to provide non-trivial guarantees between z and z' as long as the lifting admits appropriate *robustness*.

Claim 2.6.31 (Coupled unique decoding from distance)**.** *Suppose that* dsum *is a* $(1/4 - \varepsilon_0/2, 1/2 - \varepsilon)$*-robust lifting from* $\mathcal{C}_1$ *to* $\mathcal{C}_k$*. Let* (z, y) *and* (z', y') *be coupled pairs. If* $y \in \mathcal{C}_k$ *(equivalently* $z \in \mathcal{C}_1$*) and* $\Delta(y, y') < 1/2 - \varepsilon$*, then* $\Delta\Big(z, z'\Big) \leq 1/4 - \varepsilon_0/2$*, i.e.,* z' *is within the unique decoding radius of* z*.*

Proof. Towards a contradiction suppose that $\Delta(z, z) \geq 1/4 - \varepsilon_0/2$. Since the lifting is $(1/4 - \varepsilon_0/2, 1/2 - \varepsilon)$-robust, this implies that $\Delta\Big(y, y'\Big) \geq 1/2 - \varepsilon$ contradicting our assumption. ∎

From bias amplification (i.e., parity sampling), we deduce Claim 2.6.32.

Claim 2.6.32 (Coupled unique decoding from bias I)**.** *Suppose* dsum *is a* $(1/2 - \varepsilon_0, 1/2 - \varepsilon)$*-robust linear lifting from* $\mathcal{C}_1$ *to* $\mathcal{C}_k$ *which is also a* $(1/2 + \varepsilon_0, 2 \cdot \varepsilon)$*-parity sampler. Let* (z, y) *and* (z', y') *be coupled pairs. If* $y \in \mathcal{C}_k$ *(equivalently* $z \in \mathcal{C}_1$*) and* $|\mathbb{E}_{\mathfrak{s} \sim \Pi_k}[y_{\mathfrak{s}} \cdot y'_{\mathfrak{s}}]| > 2 \cdot \varepsilon$*, then*

$$|\mathbb{E}_{i \sim \Pi_1}[z_i \cdot z'_i]| \geq 1/2 + \varepsilon_0,$$

i.e., either z' or $-z'$ is within the unique decoding radius of z.

Proof. The verification follows easily from our assumptions. Towards a contradiction suppose that $|\mathbb{E}_{i\sim\Pi_1}[z_i \cdot z'_i]| < 1/2+\varepsilon_0$, i.e., the word $z'' = z \cdot z'$ has bias at most $1/2+\varepsilon_0$. Using the assumption that the lift is linear, we have $\mathrm{dsum}(z'') = \mathrm{dsum}(z) \cdot \mathrm{dsum}(z')$. Since the lifting takes bias $1/2+\varepsilon_0$ to $2\cdot\varepsilon$, we have

$$\mathrm{bias}(\mathrm{dsum}(z) \cdot \mathrm{dsum}(z')) = \mathrm{bias}(\mathrm{dsum}(z'')) \leq 2\cdot\varepsilon,$$

or equivalently $|\mathbb{E}_{\mathfrak{s}\sim\Pi_k}[y_{\mathfrak{s}} \cdot y'_{\mathfrak{s}}]| \leq 2\cdot\varepsilon$ contradicting our assumption. ∎

If the lifting function is odd, then we obtain Claim 2.6.33.

Claim 2.6.33 (Coupled unique decoding from bias II)**.** *Suppose* dsum *is a* $(1/4-\varepsilon_0/2, 1/2-\varepsilon)$*-robust lifting from* $\mathcal{C}_1$ *to* $\mathcal{C}_k$ *which is odd, i.e.,* $\mathrm{dsum}(-z) = -\mathrm{dsum}(z)$*. Let* (z,y) *and* (z',y') *be coupled pairs. If* $y \in \mathcal{C}_k$ *(equivalently* $z \in \mathcal{C}_1$*) and* $|\mathbb{E}_{\mathfrak{s}\sim\Pi_k}[y_{\mathfrak{s}} \cdot y'_{\mathfrak{s}}]| > 2\cdot\varepsilon$*, then either* z' *or* $-z'$ *is within the unique decoding radius of* z*.*

Proof. Since $|\mathbb{E}_{\mathfrak{s}\sim\Pi_k}[y_{\mathfrak{s}} \cdot y'_{\mathfrak{s}}]| > 2\cdot\varepsilon$ and the lifting is odd, either

$$\mathbb{E}_{\mathfrak{s}\sim\Pi_k}[y_{\mathfrak{s}} \cdot \mathrm{dsum}(z')_{\mathfrak{s}}] > 2\cdot\varepsilon,$$

or

$$\mathbb{E}_{\mathfrak{s}\sim\Pi_k}[y_{\mathfrak{s}} \cdot \mathrm{dsum}(-z')_{\mathfrak{s}}] = \mathbb{E}_{\mathfrak{s}\sim\Pi_k}\left[-y_{\mathfrak{s}} \cdot \mathrm{dsum}(z')_{\mathfrak{s}}\right] > 2\cdot\varepsilon.$$

Then either $\Delta(y, \mathrm{dsum}(z')) \leq 1/2-\varepsilon$ or $\Delta(y, \mathrm{dsum}(-z')) \leq 1/2-\varepsilon$. Using Claim 2.6.31 we conclude the proof. ∎

Cover Purification

A δ-bias cover $\mathcal{M}$ of $\mathcal{L}$ for small δ may require further processing in order to actually retrieve $\mathcal{L}$. Provided the lifting is sufficiently robust, trying to unique decode z for $(z,y) \in \mathcal{M}^{\pm}$,

where $\mathcal{M}^{\pm}$ is the sign completion as defined next, and then lifting the decoded word yields a new coupled list that contains $\mathcal{L}$. This process is referred to as cover purification and its formalization is the object of this section.

Definition 2.6.34 (Sign Completion)**.** *Let $\mathcal{M}$ be coupled list. We say that $\mathcal{M}^{\pm}$ defined as*

$$\mathcal{M}^{\pm} \coloneqq \{(z, \mathrm{dsum}(z)), (-z, \mathrm{dsum}(-z)) \mid (z, y) \in \mathcal{M}\},$$

is the sign completion of $\mathcal{M}$.

The correctness of the cover purification process is established next.

Lemma 2.6.35 (Purification lemma)**.** *Suppose* dsum *is a $(1/2 - \varepsilon_0, 1/2 - \varepsilon)$-robust lifting from $\mathcal{C}_1$ to $\mathcal{C}_k$ which is either*

- *linear and a $(1/2 + \varepsilon_0, 2 \cdot \varepsilon)$-parity sampler; or*
- *$(1/4 - \varepsilon_0/2)$-robust and odd.*

Let $\tilde{y} \in \{\pm 1\}^{X(k)}$ be $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$ and $\mathcal{L} = \mathcal{L}(\tilde{y}, \mathcal{C}_k)$ be its code list. If $\mathcal{M} = \{(z^{(i)}, y^{(i)}) | i \in [h]\}$ is a 2ε-bias cover of $\mathcal{L}$, then

$$\mathcal{L} \subseteq \left\{\mathrm{dsum}(z) \mid z \in \mathrm{Dec}_{\mathcal{C}_1}\left(\mathrm{P}_{\mathbf{1}}\left(\mathcal{M}^{\pm}\right)\right)\right\} =: \mathcal{L}',$$

where $\mathrm{P}_{\mathbf{1}}$ is the projection on the first coordinate and $\mathrm{Dec}_{\mathcal{C}_1}$ is a unique decoder for $\mathcal{C}_1$. Furthermore, $\mathcal{L}'$ can be computed in time $O(|\mathcal{M}| \cdot f(n))$ where $f(n)$ is the running time of a unique decoding algorithm of $\mathcal{C}_1$.

Proof. Let $y \in \mathcal{L}$. By the 2ε-cover property, there exists a coupled pair $(z', y') \in \mathcal{M}$ satisfying $|\mathbb{E}_{\mathfrak{s} \sim \Pi_k}[y_{\mathfrak{s}} \cdot y'_{\mathfrak{s}}]| > 2 \cdot \varepsilon$. Combining this bound with the appropriate robustness assumptions, Claim 2.6.32 or Claim 2.6.33 yields that either z' or $-z'$ can be uniquely decoded in $\mathcal{C}_1$.

Then

$$y \in \left\{ \mathrm{dsum}(z) \mid z \in \mathrm{Dec}_{\mathcal{C}_1}\left(\mathrm{P}_{\mathbf{1}}\left(\mathcal{M}^{\pm}\right)\right)\right\}.$$

Finally, observe that computing $\mathcal{L}'$ with the claimed running time is straightforward. ■

Algorithmically, cover purification works by running the unique decoding algorithm of $\mathcal{C}_1$ on every element of the sign completion $\mathcal{M}^{\pm}$, described below in Algorithm 2.6.36.

Algorithm 2.6.36 (Cover Purification Algorithm)**.**

Input *A 2ε-bias cover $\mathcal{M}$ for $\mathcal{L}(\tilde{y}, \mathcal{C}_k)$.*

Output *Coupled List $\mathcal{L}'$ s.t. $\mathrm{P}_2(\mathcal{L}') \supseteq \mathcal{L}(\tilde{y}, \mathcal{C}_k)$.*

1. *Let $\mathcal{L}' = \emptyset$*
2. *For $(z', y') \in \mathcal{M}^{\pm}$ do*
3. *If z' is uniquely decodable in $\mathcal{C}_1$ then*
4. *Let $z = \mathrm{UniqueDecode}_{\mathcal{C}_1}(z')$*
5. *Let $y = \mathrm{dsum}(z)$*
6. *Set $\mathcal{L}' = \mathcal{L}' \cup \{(z, y)\}$*
7. *Output $\mathcal{L}'$.*

Correctness of the List Decoding Algorithm

The building blocks developed so far are assembled to form the final list decoding algorithm (Algorithm 2.6.16), which is restated below for convenience.

Algorithm 2.6.37 (List Decoding Algorithm)**.**

Input *A word* $\tilde{y} \in \{\pm 1\}^{X(k)}$ $(1/2 - \sqrt{\varepsilon})$*-close to* $\mathcal{C}_k = \mathrm{dsum}(\mathcal{C}_1)$

Output *Coupled code list* $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$*.*

1. *Solve the List Decoding Program with* η*-accuracy obtaining* $\mathbf{Z}$ *where* $\eta = \varepsilon^8/2^{22}$
2. *Let* $\mathcal{M}$ *be the output of the Cover Retrieval Algorithm 2.6.29 on* $\mathbf{Z}$
3. *Let* $\mathcal{L}'$ *be the output of the Cover Purification Algorithm 2.6.36 on* $\mathcal{M}$
4. *Let* $\mathcal{L}'' = \{(z, y) \in \mathcal{L}' \mid \Delta(\tilde{y}, y) \leq 1/2 - \sqrt{\varepsilon}\}$
5. *Output* $\mathcal{L}''$*.*

We are ready to prove the main theorem of the abstract list decoding framework which follows easily from the properties developed so far.

Theorem 2.6.38 (List Decoding Theorem (Restatement of Theorem 2.6.17))**.** *Suppose that* dsum *is a* $(1/2 - \varepsilon_0, 1/2 - \varepsilon)$*-robust* $(\varepsilon^8/2^{22}, L)$*-two-step tensorial lifting from* $\mathcal{C}_1$ *to* $\mathcal{C}_k$ *which is either*

- *linear and a* $(1/2 + \varepsilon_0, 2 \cdot \varepsilon)$*-parity sampler; or*
- $(1/4 - \varepsilon_0, 1/2 - \varepsilon/2)$*-robust and odd.*

Let $\tilde{y} \in \{\pm 1\}^{X(k)}$ *be* $(1/2 - \sqrt{\varepsilon})$*-close to* $\mathcal{C}_k$*. Then w.v.h.p. the List Decoding Algorithm 2.6.16 returns the coupled code list* $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$*. Furthermore, the running time is*

$$n^{O(L+k)} \left(\mathrm{polylog}(\varepsilon^{-1}) + f(n) \right),$$

where $n = |X(1)|$ *and* $f(n)$ *is the running time of a unique decoding algorithm of* $\mathcal{C}_1$*.*

Proof. Under the assumptions of the theorem, Lemma 2.6.30 establishes that the Cover Retrieval Algorithm 2.6.29 returns w.v.h.p. a 2ε-bias cover. Then, Lemma 2.6.35 states

that providing this 2ε-bias cover as input to the Cover Purification Algorithm 2.6.36 yields a coupled list containing the code list $\mathcal{L}(\widetilde{y}, \mathcal{C}_1, \mathcal{C}_k)$. Finally, the last step in Algorithm 2.6.16 ensures the output is precisely $\mathcal{L}(\widetilde{y}, \mathcal{C}_1, \mathcal{C}_k)$. ∎

2.7 Instantiation I: Direct Sum on HDXs

We instantiate the list decoding framework to the direct sum lifting on HDXs obtaining Theorem 2.7.1, which is the main result in this section. For this instantiation we need to establish that HDXs are two-step tensorial which will be done in Section 2.7.1.

Theorem 2.7.1 (Direct Sum Lifting on HDX)**.** *Let $\varepsilon_0 < 1/2$ be a constant and $\varepsilon \in (0, \varepsilon_0)$. There exist universal constants $c, C > 0$ such that for any γ-HDX $X(\leq d)$ on ground set $X(1) = [n]$ and Π_1 uniform, if*

$$\gamma \leq (\log(1/\varepsilon))^{-C \cdot \log(1/\varepsilon)} \quad \text{and} \quad d \geq c \cdot \frac{(\log(1/\varepsilon))^2}{\varepsilon},$$

then the following holds:

For every binary code $\mathcal{C}_1$ with $\Delta(\mathcal{C}_1) \geq 1/2 - \varepsilon_0$ on $X(1) = [n]$, there exists a binary lifted code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\varphi(\mathcal{C}_1))$ with $\Delta(\mathcal{C}_k) \geq 1/2 - \varepsilon^{\Omega_{\varepsilon_0}(1)}$ on $X(k)$ where $k = O(\log(1/\varepsilon))$, φ is an explicit linear projection, and

- *[Efficient List Decoding] If $\tilde{y}$ is $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$, then we can compute the list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ (c.f. Definition 2.6.15) in time*

$$n^{\varepsilon^{-O(1)}} \cdot f(n),$$

  where $f(n)$ is the running time of a unique decoding algorithm for $\mathcal{C}_1$.

- *[Rate] The rate r_k of $\mathcal{C}_k$ satisfies $r_k = r_1 \cdot |X(1)| / |X(k)|$ where r_1 is the relative rate of $\mathcal{C}_1$.*

- *[Linearity] If $\mathcal{C}_1$ is linear, then φ is the identity and $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ is linear.*

In particular, invoking Theorem 2.7.1 on HDXs extracted from Ramanujan complexes (as in Lemma 2.4.5), we obtain Corollary 2.7.2.

Corollary 2.7.2. *Let $\varepsilon_0 < 1/2$ be a constant and $\varepsilon \in (0, \varepsilon_0)$. There is an infinite sequence of HDXs $X_1, X_2, \ldots$ on ground sets of size $n_1, n_2, \ldots$ such that the following holds:*

For every sequence of binary codes $\mathcal{C}_1^{(i)}$ on $[n_i]$ with rate and distance uniformly bounded by r_1 and $(1/2 - \varepsilon_0)$ respectively, there exists a sequence of binary lifted codes $\mathcal{C}_k^{(i)} = \mathrm{dsum}_{X(k)}(\varphi(\mathcal{C}_1^{(i)}))$ on a collection $X_i(k)$ with $\Delta(\mathcal{C}_k^{(i)}) \geq 1/2 - \varepsilon^{\Omega_{\varepsilon_0}(1)}$ where φ is an explicit linear projection and

- *[Efficient List Decoding] If $\tilde{y}$ is $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$, then we can compute the list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ (c.f. Definition 2.6.15) in time $n^{\varepsilon^{-O(1)}} \cdot f(n)$, where $f(n)$ is the running time of a unique decoding algorithm of $\mathcal{C}_1$.*

- *[Explicit Construction] The collection $X_i(k)$ is part of an explicit γ-HDX $X_i(\leq d)$ where $k = O\left(\log(1/\varepsilon)\right)$, $d = O\left((\log(1/\varepsilon))^2/\varepsilon\right)$, and $\gamma = (\log(1/\varepsilon))^{-O(\log(1/\varepsilon))}$.*

- *[Rate] The rate $r_k^{(i)}$ of $\mathcal{C}_k^{(i)}$ satisfies $r_k^{(i)} \geq r_1 \cdot \exp\left(-(\log(1/\varepsilon))^{O(\log(1/\varepsilon))}\right)$.*

- *[Linearity] If $\mathcal{C}_1^{(i)}$ is linear, then φ is the identity and $\mathcal{C}_k^{(i)} = \mathrm{dsum}_{X(k)}(\mathcal{C}_1^{(i)})$ is linear.*

Proof. Efficient list decoding and linearity follow directly from Theorem 2.7.1, and the parameters of the explicit construction match the requirements of the theorem. The only thing left to do is to calculate the rate. Since the lifting $\mathrm{dsum}_{X_i(k)}$ needs to be a $(2\varepsilon_0, 2\varepsilon)$-parity sampler to achieve the promised distance, by Lemma 2.4.11 the rate $r_k^{(i)}$ of $\mathcal{C}_k^{(i)}$ satisfies

$$r_k^{(i)} \geq r_1 \cdot \gamma^{O((\log(1/\varepsilon))^4/(\varepsilon^2\gamma))}$$

Since $\gamma = (\log(1/\varepsilon))^{-O(\log(1/\varepsilon))}$, this reduces to

$$r_k^{(i)} \geq r_1 \cdot (\log(1/\varepsilon))^{-O((\log(1/\varepsilon))^5/(\varepsilon^2 \cdot \gamma))} = r_1 \cdot \exp(1/\gamma^{O(1)}) = r_1 \cdot \exp\left(-(\log(1/\varepsilon))^{O(\log(1/\varepsilon))}\right).$$

■

2.7.1 HDXs are Two-Step Tensorial

Theorem 2.7.3 proven in [AJT19] establishes that HDXs of appropriate expansion parameter are tensorial objects for constant $L = O_{k,q,\mu}(1)$.

Theorem 2.7.3 (HDXs are Tensorial)**.** *There exist some universal constants $c' \geq 0$ and $C' \geq 0$ satisfying the following: If $L \geq c' \cdot (q^k \cdot k^5/\mu^4)$, $\mathrm{Supp}(\mathbf{Z}_j) \leq q$ for all $j \in [n]$, and X is a γ-HDX for $\gamma \leq C' \cdot \mu^4/(k^{8+k} \cdot 2^{6k} \cdot q^{2k})$ and size $\geq k$, then $X(k)$ endowed with a distribution Π_k is (μ, L)-tensorial.*

The next result shows that HDXs are also two-step tensorial objects with the same parameters as above.

Lemma 2.7.4 (HDXs are two-step tensorial)**.** *There exist some universal constants $c' \geq 0$ and $C' \geq 0$ satisfying the following: If $L \geq c' \cdot (q^k \cdot k^5/\mu^4)$, $\mathrm{Supp}(\mathbf{Z}_j) \leq q$ for all $j \in [n]$, and X is a γ-HDX for $\gamma \leq C' \cdot \mu^4/(k^{8+k} \cdot 2^{6k} \cdot q^{2k})$ and size $\geq k$, then $X(k)$ is (μ, L)-two-step tensorial.*

Proof. Under our assumptions the (μ, L)-tensorial property follows from Theorem 2.7.3 (this is the only place where the assumption on γ is used), so we only need to show

$$\mathop{\mathbb{E}}_{\mathfrak{s},\mathfrak{t} \sim \Pi_k} \left\| \{\mathbf{Z}'_{\mathfrak{s}} \mathbf{Z}'_{\mathfrak{t}}\} - \left\{\mathbf{Z}'_{\mathfrak{s}}\right\}\left\{\mathbf{Z}'_{\mathfrak{t}}\right\} \right\|_1 \leq \mu,$$

which can be proven by adapting a potential argument technique from [BRS11]. First, set

the potential

$$\Phi_m = \mathop{\mathbb{E}}_{S\sim\Pi_k^m} \mathop{\mathbb{E}}_{\sigma\sim\{\mathbf{Z}_S\}} \mathop{\mathbb{E}}_{\mathfrak{s}\sim\Pi_k} \mathrm{Var}\left[\mathbf{Z}_{\mathfrak{s}} \mid \mathbf{Z}_S = \sigma\right], \tag{2.10}$$

and consider the error term

$$\mu_m \coloneqq \mathop{\mathbb{E}}_{S\sim\Pi_k^m} \mathop{\mathbb{E}}_{\sigma\sim\{\mathbf{Z}_S\}} D(S,\sigma), \tag{2.11}$$

where $D(S,\sigma) \coloneqq \mathbb{E}_{\mathfrak{s},\mathfrak{t}\sim\Pi_k}[\|\{\mathbf{Z}_{\mathfrak{s}}\mathbf{Z}_{\mathfrak{t}} \mid \mathbf{Z}_S = \sigma\} - \{\mathbf{Z}_{\mathfrak{s}}|\mathbf{Z}_S = \sigma\}\{\mathbf{Z}_{\mathfrak{t}}|\mathbf{Z}_S = \sigma\}\|_1]$. If $\mu_m \geq \mu/2$, then

$$\mathop{\mathbb{P}}_{S\sim\Pi_k^m,\sigma\sim\{\mathbf{Z}_S\}} [D(S,\sigma) \geq \mu_m/2] \geq \frac{\mu}{4}.$$

Let $G = (V = X(k), E)$ be the weighted graph where $E = \{\{\mathfrak{s},\mathfrak{t}\} \mid \mathfrak{s},\mathfrak{t} \in X(k)\}$ and each edge $\{\mathfrak{s},\mathfrak{t}\}$ receives weight $\Pi_k(\mathfrak{s}) \cdot \Pi_k(\mathfrak{t})$. Local correlation (expectation over the edges) on this graph G is the same as to global correlation (expectation over two independent copies of vertices). Then, we obtain[18]

$$\Phi_m - \Phi_{m+1} \geq \mathop{\mathbb{P}}_{S\sim\Pi_k^m,\sigma\sim\{\mathbf{Z}_S\}} [D(S,\sigma) \geq \mu_m/2] \cdot \frac{\mu^2}{2q^{2k}}.$$

Since $1 \geq \Phi_1 \geq \cdots \geq \Phi_{L/k} \geq 0$, there can be at most $8q^{2k}/\mu^3$ indices $m \in [L/k]$ such that $\mu_m \geq \mu/2$. In particular, since the total number of indices is L/k, we have

$$\mathop{\mathbb{E}}_{m\in[L/k]} \mu_m \leq \frac{\mu}{2} + \frac{k}{L} \cdot \frac{8q^{2k}}{\mu^3}.$$

Our choice of L is more than enough to ensure $\mathbb{E}_{m\in[L/k]}[\mu_m] \leq \mu$. ∎

18. See [AJT19] or [BRS11] for the details.

2.7.2 Instantiation to Linear Base Codes

First, we instantiate the list decoding framework to the seemingly simpler case of binary *linear* base codes in Lemma 2.7.5. As we show later, with a simple observation we can essentially use the proof of Lemma 2.7.5 to obtain Theorem 2.7.1 for general codes.

Lemma 2.7.5 (Direct sum lifting of linear biased codes)**.** *Let $\varepsilon_0 < 1/2$ be a constant and $\varepsilon \in (0, \varepsilon_0)$. There exist universal constants $c, C > 0$ such that for any γ-HDX $X(\leq d)$ on ground set $X(1) = [n]$ and Π_1 uniform, if*

$$\gamma \leq \log(1/\varepsilon)^{-C\cdot(\log(1/\varepsilon))} \quad \text{and} \quad d \geq c \cdot \frac{(\log(1/\varepsilon))^2}{\varepsilon},$$

then the following holds:

For every binary $2\varepsilon_0$-biased linear code $\mathcal{C}_1$ on $X(1) = [n]$, there exists a 2ε-biased binary lifted linear code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ on $X(k)$ where $k = O(\log(1/\varepsilon))$ and

- *[Efficient List Decoding] If $\tilde{y}$ is $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$, then we can compute the list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ (c.f. Definition 2.6.15) in time*

$$n^{\varepsilon^{-O(1)}} \cdot f(n),$$

  where $f(n)$ is the running time of a unique decoding algorithm for $\mathcal{C}_1$.

- *[Rate] The rate r_k of $\mathcal{C}_k$ satisfies $r_k = r_1 \cdot |X(1)| / |X(k)|$ where r_1 is the relative rate of $\mathcal{C}_1$.*[19]

- *[Linear] The lifted code $\mathcal{C}_k$ is linear.*

Proof. We show that under our assumption on the γ-HDX $X(\leq d)$ we obtain sufficient *robustness* and *tensorial* parameters to apply Theorem 2.6.17. In this application, we will

19. For the rate computation, we assume that $X(k)$ can be expressed as a multi-set such that the uniform distribution on it coincides with Π_k, which is true in the case that Π_k is D-flat.

rely on parity sampling for robustness. If $\mathrm{dsum}_{X(k)}$ is a $(2\varepsilon_0, 2\varepsilon)$-parity sampler, using the linearity of $\mathcal{C}_1$ we obtain a lifted code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ which is linear and has bias 2ε; thus the lifting is indeed $(1/2-\varepsilon_0, 1/2-\varepsilon)$-robust. If we want to fully rely on parity sampling in Theorem 2.6.17, the lifting must be a $(\beta_0 = 1/2+\varepsilon_0, \beta = 2\varepsilon)$-parity sampler, which is more stringent than the first parity sampling requirement.[20] To invoke Lemma 2.4.10 and obtain this (β_0, β)-parity sampler, we need to choose a parameter θ (where $0 < \theta < (1-\beta_0)/\beta_0$) and

$$\begin{aligned} &k \geq \log_{(1+\theta)\beta_0}(\beta/3), \\ &d \geq \max\left(\frac{3\cdot k^2}{\beta}, \frac{6}{\theta^2\beta_0^2\beta}\right), \text{ and} \\ &\gamma = O\left(\frac{1}{d^2}\right). \end{aligned}$$

To get a (μ, L)-tensorial HDX, Theorem 2.7.3 requires

$$L \geq \frac{c'\cdot 2^k \cdot k^5}{\mu^4} \quad \text{and} \quad \gamma \leq \frac{C'\cdot \mu^4}{k^{8+k}\cdot 2^{8k}}.$$

where we used that our alphabet is binary (i.e., $q = 2$) and $c', C' > 0$ are constants. Finally, Theorem 2.6.17 requires $\mu \leq \varepsilon^8/2^{22}$. The conceptual part of the proof is essentially complete and we are left to compute parameters. Set $\zeta_0 = 3/4 + \varepsilon_0 - \varepsilon_0^2$. We choose $\theta = 1/2 - \varepsilon_0$ which makes $(1+\theta)\beta_0$ equal to ζ_0 (provided $\varepsilon_0 < 1/2$ we have $\zeta_0 < 1$). This choice results in

$$k \geq \lceil \log_{\zeta_0}(2\varepsilon/3)\rceil \quad \text{and} \quad d = O\left(\max\left(\frac{\log_{\zeta_0}(2\varepsilon/3)}{\varepsilon}, \frac{1}{(1/4-\varepsilon_0^2)^4\cdot\varepsilon}\right)\right).$$

Combining the parity sampling and tensorial requirements and after some simplification, the

20. Recall that this strengthening is used in our list decoding framework.

expansion γ is constrained as

$$\gamma \leq C'' \cdot \min\left(\frac{\varepsilon^{32}}{k^{8+k} \cdot 2^{8k}}, \frac{\varepsilon^2}{k^4}, \left(1/4 - \varepsilon_0^2\right)^4 \cdot \varepsilon^2\right),$$

where $C'' > 0$ is a constant. We deduce that taking γ as

$$\gamma \leq C'' \cdot \frac{\left(1/4 - \varepsilon_0^2\right)^4 \cdot \varepsilon^{32}}{k^{8+k} \cdot 2^{8k}},$$

is sufficient. Further simplifying the above bound gives

$$\gamma = O\left(\frac{\left(1/4 - \varepsilon_0^2\right)^4 \cdot \varepsilon^{32}}{\left(\log_{\zeta_0}(2\varepsilon/3)\right)^{8+\log_{\zeta_0}(2\varepsilon/3)} \cdot (2\varepsilon/3)^{8/\log(\zeta_0)}}\right).$$

Now, we turn to the SOS-related parameter L which is constrained to be

$$L \geq c'' \cdot \frac{2^k \cdot k^5}{\varepsilon^{32}},$$

where $c'' > 0$. Note that in this case the exponent $O(L + k)$ appearing in the running time of Theorem 2.6.17 becomes $O(L)$. Similarly, further simplification leads to

$$L = O\left(\frac{\left(\log_{\zeta_0}(2\varepsilon/3)\right)^5 \cdot (3/2\varepsilon)^{-1/\log(\zeta_0)}}{\varepsilon^{32}}\right).$$

Taking ε_0 to be a constant and simplifying yields the claimed parameters. ∎

2.7.3 Instantiation to General Base Codes

We can extend Lemma 2.7.5 to an arbitrary (not necessarily linear) binary base code $\mathcal{C}_1$ with the natural caveat of no longer obtaining linear lifted code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$. However, even if $\mathcal{C}_1$ has small bias, it might not be the case that the difference of any two codewords

will have small bias, which is required for list decoding. To this end we modify the code $\mathcal{C}_1$ by employing a projection φ which converts a condition on the distance of the code to a condition on the bias of the difference of any two codewords.

Claim 2.7.6. *If $\mathcal{C}_1$ is binary code on $[n]$ with relative distance δ and rate r, then there exists an explicit linear projection $\varphi\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that the code $\mathcal{C}_1' = \varphi(\mathcal{C}_1)$ has relative distance at least $\delta/2$ and rate r. Furthermore, for every $z, z' \in \mathcal{C}_1'$ we have*

$$\operatorname{bias}(z - z') \leq 1 - \frac{\delta}{2}.$$

Proof. Take φ to be the projector onto $\mathbb{F}_2^{n-s} \oplus \{0\}^s$ where $s = \lfloor \delta n/2 \rfloor$. Then

$$\mathcal{C}_1' \coloneqq \varphi(\mathcal{C}_1) = \{(z_1, \ldots, z_{n-s}, \underbrace{0, \ldots, 0}_{s}) \mid (z_1, \ldots, z_n) \in \mathcal{C}_1\},$$

and the claim readily follows. ∎

With this modification in mind, we can now restate and prove Theorem 2.7.1.

Theorem 2.7.7 (Direct Sum Lifting on HDX (Restatement of Theorem 2.7.1))**.** *Let $\varepsilon_0 < 1/2$ be a constant and $\varepsilon \in (0, \varepsilon_0)$. There exist universal constants $c, C > 0$ such that for any γ-HDX $X(\leq d)$ on ground set $X(1) = [n]$ and Π_1 uniform, if*

$$\gamma \leq (\log(1/\varepsilon))^{-C \cdot \log(1/\varepsilon)} \quad \text{and} \quad d \geq c \cdot \frac{(\log(1/\varepsilon))^2}{\varepsilon},$$

then the following holds:

For every binary code $\mathcal{C}_1$ with $\Delta(\mathcal{C}_1) \geq 1/2 - \varepsilon_0$ on $X(1) = [n]$, there exists a binary lifted code $\mathcal{C}_k = \operatorname{dsum}_{X(k)}(\varphi(\mathcal{C}_1))$ with $\Delta(\mathcal{C}_k) \geq 1/2 - \varepsilon^{\Omega_{\varepsilon_0}(1)}$ on $X(k)$ where $k = O(\log(1/\varepsilon))$, φ is an explicit linear projection, and

- *[Efficient List Decoding] If $\tilde{y}$ is $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$, then we can compute the list*

$\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ *(c.f. Definition 2.6.15) in time*

$$n^{\varepsilon^{-O(1)}} \cdot f(n),$$

where $f(n)$ *is the running time of a unique decoding algorithm for* $\mathcal{C}_1$*.*

- *[Rate] The rate* r_k *of* $\mathcal{C}_k$ *satisfies* $r_k = r_1 \cdot |X(1)| / |X(k)|$ *where* r_1 *is the relative rate of* $\mathcal{C}_1$*.*

- *[Linearity] If* $\mathcal{C}_1$ *is linear, then* φ *is the identity and* $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ *is linear.*

Proof. By virtue of Lemma 2.7.5, it is enough to consider when $\mathcal{C}_1$ is not linear. Note that in the proof of Lemma 2.7.5 the only assumption about linearity of $\mathcal{C}_1$ we used to obtain $(1/2-\varepsilon_0, 1/2-\varepsilon)$-robustness was that the sum of two codewords is in the code and hence it has small bias. For a general code $\mathcal{C}_1$ of constant distance $1/2-\varepsilon_0$, applying Claim 2.7.6 we obtain a new code $\mathcal{C}_1'$ with this guarantee at the expense of a distance $1/2$ times the original one. Naturally, in the current proof we no longer obtain a linear lifted code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1')$. Excluding the two previous remarks the proof of Theorem 2.7.1 is now the same as the proof of Lemma 2.7.5. ∎

2.8 List Decoding Direct Product Codes

2.8.1 Direct Product Codes

Having developed a decoding algorithm for direct sum, a promising strategy for list decoding other lifted codes on expanding objects is reducing them to instances of direct sum list decoding. One such reduction involves the direct product lifting, which was first studied in the context of samplers by Alon et al. in [ABN$^+$92]. The direct product lifting collects the entries of a code on each subset of size ℓ.

Definition 2.8.1 (Direct Product Lifting)**.** *Let $\mathcal{C}_1 \subseteq \mathbb{F}_2^n$ be a base code on $X(1) = [n]$. The* direct product lifting *of a word $z \in \mathbb{F}_2^n$ on a collection $X(\ell)$ is $\mathrm{dprod}_{X(\ell)}(z) = (x_\mathfrak{t})_{\mathfrak{t} \in X(\ell)}$, where $x_\mathfrak{t} = (z_i)_{i \in \mathfrak{t}}$. The direct product lifting of the entire code is $\mathrm{dprod}_{X(\ell)}(\mathcal{C}_1) = \{\mathrm{dprod}_{X(\ell)}(z) \mid z \in \mathcal{C}_1\}$, which is a code of length $|X(\ell)|$ over the alphabet $\mathbb{F}_2^\ell$.*

If X is a HDX, its sampling properties ensure that the direct product lifting has very high distance. It follows from the definition that if the bipartite graph between $X(1)$ and $X(k)$ is an (η, δ)-sampler and the code $\mathcal{C}_1$ has minimum distance η, then the direct product lifting $\mathrm{dprod}_{X(\ell)}(\mathcal{C}_1)$ has minimum distance at least $(1 - \delta)$. Recalling from Fact 2.4.8 that the bipartite graph between two levels of a HDX can be a sampler with arbitrarily small parameters if the expansion is good enough, we can reasonably hope to list decode the direct product lifting on a HDX up to a distance close to 1. In fact, Dinur et al. [DHK+19] provided a list decoding algorithm accomplishing exactly that. We offer a very different approach to the same list decoding problem.

2.8.2 Direct Product List Decoding

We will reduce direct product decoding on $X(\ell)$ to direct sum decoding on $X(k)$, where $k \approx \ell/2$. This requires converting a received word $\tilde{x} \in (\mathbb{F}_2^\ell)^{X(\ell)}$ to a word $\tilde{y} \in \mathbb{F}_2^{X(k)}$ that we will decode using the direct sum algorithm. If we knew that $\tilde{x} = \mathrm{dprod}_{X(\ell)}(\tilde{z})$ for some $\tilde{z} \in \mathbb{F}_2^{X(1)}$, we would do so by simply taking $\tilde{y}_\mathfrak{s} = \sum_{i \in \mathfrak{s}} \tilde{z}_i$ to be the direct sum lifting on each edge $\mathfrak{s}$; that is, $\tilde{y} = \mathrm{dsum}_{X(k)}(\tilde{z})$.

Unfortunately, performing list decoding also involves dealing with words $\tilde{x}$ that might not have arisen from the direct product lifting. To construct a corrupted instance of direct sum $\tilde{y}$ from $\tilde{x}$, we need to assign values to each face $\mathfrak{s} \in X(k)$ based only on the information we have on the faces $X(\ell)$, as there is no word on the ground set to refer to. Since different faces $\mathfrak{t}, \mathfrak{t}' \in X(\ell)$ containing $\mathfrak{s}$ might not agree on $\mathfrak{s}$, there could be ambiguity as to what value to assign for the sum on $\mathfrak{s}$.

This is where the D-flatness of the distribution Π_ℓ (which holds for the γ-HDX construction described in Lemma 2.4.5) comes in. Recall that to obtain codewords in $\mathrm{dprod}_{X(\ell)}(\mathcal{C}_1)$ without weights on their entries, we duplicate each face $\mathfrak{t} \in X(\ell)$ at most D times to make the distribution Π_ℓ uniform. To perform the same kind of duplication on $X(k)$ that makes Π_k uniform, note that each face $\mathfrak{s} \in X(k)$ has $\Pi_k(\mathfrak{s})$ proportional to $|\{\mathfrak{t} \in X(\ell) \mid \mathfrak{t} \supset \mathfrak{s}\}|$ (where $X(\ell)$ is thought of as a multiset), so we will create one copy of $\mathfrak{s}$ for each $\mathfrak{t}$ containing it. Thus we can assign a unique $\mathfrak{t} \supset \mathfrak{s}$ to each copy. By downward closure, the distribution on $X(\ell)$ obtained by choosing $\mathfrak{s}$ uniformly from the multiset $X(k)$ and then selecting its associated face $\mathfrak{t}$ will be uniform, just like Π_ℓ. With this careful duplication process, we are ready to define the function ρ_k that takes a corrupted direct product word $\tilde{x}$ to a corrupted direct sum word $\tilde{y}$.

Definition 2.8.2 (Reduction Function)**.** *Let $k < \ell$ and X be a HDX where the distribution Π_ℓ is D-flat. Duplicate faces in $X(k)$ so that Π_k is uniform, and assign a face $\mathfrak{t}_\mathfrak{s} \in X(\ell)$ to each $\mathfrak{s} \in X(k)$ (after duplication) such that $\mathfrak{t}_\mathfrak{s}$ is distributed according to Π_ℓ when $\mathfrak{s}$ is selected uniformly from $X(k)$. The function $\rho_k : (\mathbb{F}_2^\ell)^{X(\ell)} \to \mathbb{F}_2^{X(k)}$ is defined as*

$$(\rho_k(\tilde{x}))_\mathfrak{s} = \sum_{i \in \mathfrak{s}} (\tilde{x}_{\mathfrak{t}_\mathfrak{s}})_i.$$

The reduction function ρ_k resolves the ambiguity of which face $\mathfrak{t} \supset \mathfrak{s}$ to sample the sum from by assigning a different face to each copy of $\mathfrak{s}$ in a manner compatible with the distribution Π_ℓ. Observe that if $\tilde{x} = \mathrm{dprod}_{X(\ell)}(\tilde{z})$ for some $\tilde{z} \in \mathbb{F}_2^{X(1)}$, then $\rho_k(\tilde{x}) = \mathrm{dsum}_{X(k)}(\tilde{z})$.

The following lemma shows that performing this reduction from direct product to direct sum maintains agreement between words. It essentially says that if a received word $\tilde{x}$ exhibits some agreement with $x \in \mathrm{dprod}_{X(\ell)}(\mathcal{C}_1)$, then there is a k for which $\rho_k(\tilde{x})$ and $\rho_k(x)$ have agreement larger than $1/2$.

Lemma 2.8.3 (Product-to-sum agreement)**.** *Fix $\varepsilon > 0$ and $C' > 2$. Let $z \in \mathcal{C}_1$, $x = \mathrm{dprod}_{X(\ell)}(z)$, and $\tilde{x} \in (\mathbb{F}_2^\ell)^{X(\ell)}$. If $\Delta(x, \tilde{x}) \leq 1 - \varepsilon$, then there exists a k satisfying*

$$|k - \ell/2| < \frac{1}{2}\sqrt{C'\ell \log(1/\varepsilon)}$$

such that

$$\Delta(y, \tilde{y}) \leq 1/2 - \varepsilon/2 + \varepsilon^{C'/2},$$

where $y = \rho_k(x)$ and $\tilde{y} = \rho_k(\tilde{x})$ are words in $\mathbb{F}_2^{X(k)}$.

Proof. For $t \in X(\ell)$ and $\mathfrak{s} \subseteq \mathfrak{t}$, define the function $\chi_{\mathfrak{s},\mathfrak{t}} : \mathbb{F}_2^{\mathfrak{t}} \to \{-1, 1\}$ by

$$\chi_{\mathfrak{s},\mathfrak{t}}(w) = \prod_{i \in \mathfrak{s}} (-1)^{w_i}.$$

For each face $\mathfrak{t} \in X(\ell)$, consider the expectation $\mathbb{E}_{\mathfrak{s} \subseteq \mathfrak{t}}[\chi_{\mathfrak{s},\mathfrak{t}}(x_\mathfrak{t} - \tilde{x}_\mathfrak{t})]$, where $\mathfrak{s}$ is a subset of $\mathfrak{t}$ of any size chosen uniformly. If $x_\mathfrak{t} = \tilde{x}_\mathfrak{t}$, which happens for at least ε fraction of faces $\mathfrak{t}$, the expression in the expectation is always 1. Otherwise, this expectation is zero, so taking the expectation over the faces yields

$$\mathbb{E}_{\mathfrak{t} \sim \Pi_\ell} \mathbb{E}_{\mathfrak{s} \subseteq \mathfrak{t}}[\chi_{\mathfrak{s},\mathfrak{t}}(x_\mathfrak{t} - \tilde{x}_\mathfrak{t})] = \mathop{\mathbb{P}}_{t \sim \Pi_\ell}[x_\mathfrak{t} = \tilde{x}_\mathfrak{t}] \geq \varepsilon.$$

We would like to restrict to a fixed size of faces $\mathfrak{s}$ for which this inequality holds; as this will be the size of the direct sum faces, we need to make sure it's large enough to give us the expansion required for decoding later. Using a Chernoff bound (Fact 2.10.2 with $a = \sqrt{C'\ell \log(1/\varepsilon)}$), we see that the size of the faces is highly concentrated around $\ell/2$:

$$\mathop{\mathbb{P}}_{\mathfrak{s} \subseteq \mathfrak{t}}\left[\left||\mathfrak{s}| - \frac{\ell}{2}\right| \geq \frac{1}{2}\sqrt{C'\ell \log(1/\varepsilon)}\right] \leq 2e^{-C' \log(1/\varepsilon)/2} \leq 2\varepsilon^{C'/2}.$$

Let I be the interval

$$I = \left(\frac{\ell}{2} - \frac{1}{2}\sqrt{C'\ell \log(1/\varepsilon)}, \frac{\ell}{2} + \frac{1}{2}\sqrt{C'\ell \log(1/\varepsilon)} \right).$$

The expectation inequality becomes

$$\begin{aligned}
\varepsilon &\leq \mathbb{E}_{\mathfrak{t}\sim\Pi_\ell}[\mathbb{E}_{\mathfrak{s}\subseteq\mathfrak{t}}[\mathbb{1}_{|\mathfrak{s}|\in I} \cdot \chi_{\mathfrak{s},\mathfrak{t}}(x_\mathfrak{t} - \tilde{x}_\mathfrak{t})] + \mathbb{E}_{\mathfrak{s}\subseteq\mathfrak{t}}[\mathbb{1}_{|\mathfrak{s}|\notin I} \cdot \chi_{\mathfrak{s},\mathfrak{t}}(x_\mathfrak{t} - \tilde{x}_\mathfrak{t})]] \\
&\leq \mathbb{E}_{\mathfrak{t}\sim\Pi_\ell}\mathbb{E}_{\mathfrak{s}\subseteq\mathfrak{t},|\mathfrak{s}|\in I}[\chi_{\mathfrak{s},\mathfrak{t}}(x_\mathfrak{t} - \tilde{x}_\mathfrak{t})] + 2\varepsilon^{C'/2}.
\end{aligned}$$

Thus there exists a $k \in I$ such that

$$\varepsilon - 2\varepsilon^{C'/2} \leq \mathbb{E}_{\mathfrak{t}\sim\Pi_\ell}\mathbb{E}_{\mathfrak{s}\subseteq\mathfrak{t},|\mathfrak{s}|=k}[\chi_{\mathfrak{s},\mathfrak{t}}(x_\mathfrak{t} - \tilde{x}_\mathfrak{t})].$$

Choosing a face $\mathfrak{t}$ and then a uniformly random $\mathfrak{s} \subseteq \mathfrak{t}$ of size k results in choosing $\mathfrak{s}$ according to Π_k. Moreover, the edge $\mathfrak{t}_\mathfrak{s}$ containing $\mathfrak{s}$ from Definition 2.8.2 is distributed according to Π_ℓ. Bearing in mind the definitions of $y_\mathfrak{s}$ and $\tilde{y}_\mathfrak{s}$, we have

$$\begin{aligned}
\varepsilon - 2\varepsilon^{C'/2} &\leq \mathbb{E}_{\mathfrak{t}\sim\Pi_\ell}\mathbb{E}_{\mathfrak{s}\subseteq\mathfrak{t},|\mathfrak{s}|=k}[\chi_{\mathfrak{s},\mathfrak{t}}(x_\mathfrak{t} - \tilde{x}_\mathfrak{t})] \\
&= \mathbb{E}_{\mathfrak{s}\sim\Pi_k}[\chi_{\mathfrak{s},\mathfrak{t}_\mathfrak{s}}(x_{\mathfrak{t}_\mathfrak{s}} - \tilde{x}_{\mathfrak{t}_\mathfrak{s}})] \\
&= \mathbb{E}_{\mathfrak{s}\sim\Pi_k}[(-1)^{(\rho_k(x))_\mathfrak{s} - (\rho_k(\tilde{x}))_\mathfrak{s}}] \\
&= \mathrm{bias}(y - \tilde{y})
\end{aligned}$$

which translates to a Hamming distance of $\Delta(y, \tilde{y}) \leq 1/2 - \varepsilon/2 + \varepsilon^{C'/2}$. ∎

With Lemma 2.8.3 in hand to reduce a direct product list decoding instance to a direct sum list decoding instance, we can decode by using a direct sum list decoding algorithm as a black box.

Algorithm 2.8.4 (Direct Product List Decoding Algorithm)**.**

Input *A word* $\tilde{x} \in (\mathbb{F}_2^\ell)^{X(\ell)}$ *with distance at most* $(1-\varepsilon)$ *from* $\mathrm{dprod}_{X(\ell)}(\mathcal{C}_1)$

Output *The list* $\mathcal{L}' = \{z \in \mathbb{F}_2^n \mid \Delta(\mathrm{dprod}_{X(\ell)}(z), \tilde{x}) \leq 1-\varepsilon\}$

1. *Let* I *be the interval* $\left(\ell/2 - \sqrt{C'\ell\log(1/\varepsilon)}/2, \ell/2 + \sqrt{C'\ell\log(1/\varepsilon)}/2\right)$.
2. *For each integer* $k \in I$*, run the direct sum list decoding algorithm on the input* $\tilde{y} = \rho_k(\tilde{x}) \in \mathbb{F}_2^{X(k)}$ *to obtain a coupled list* $\mathcal{L}_k$ *of all pairs* (z, y) *with* $\Delta(y, \tilde{y}) \leq 1/2 - \varepsilon/2 + \varepsilon^{C'/2}$.
3. *Let* $\mathcal{L} = \cup_{k\in I}\{z \in \mathcal{C}_1 \mid (z, y) \in \mathcal{L}_k\}$.
4. *Let* $\mathcal{L}' = \{z \in \mathcal{L} \mid \Delta(\mathrm{dprod}_{X(\ell)}(z), \tilde{x}) \leq 1-\varepsilon\}$.
5. *Output* $\mathcal{L}'$.

Theorem 2.8.5 (Product-to-sum Reduction)**.** *Let* $\varepsilon > 0$ *and* $C' > 2$*. Let* $\mathrm{dprod}_{X(\ell)}(\mathcal{C}_1)$ *be the direct product lifting of a base code* $\mathcal{C}_1$ *on a simplicial complex* X*. If the direct sum lifting* $\mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ *is list decodable up to distance* $\left(1/2 - \varepsilon/2 + \varepsilon^{C'/2}\right)$ *in time* $\tilde{f}(n)$ *for all* k *satisfying* $|k - \ell/2| < \sqrt{C'\ell\log(1/\varepsilon)}/2$*, then Algorithm 2.8.4 list decodes* $\mathrm{dprod}_{X(\ell)}(\mathcal{C}_1)$ *up to distance* $(1-\varepsilon)$ *in running time*

$$\sqrt{C'\ell\log(1/\varepsilon)}\tilde{f}(n) + |X(\ell)||\mathcal{L}|.$$

Proof. Let $\tilde{x} \in (\mathbb{F}_2^\ell)^{X(\ell)}$ be a received word and let $z \in \mathcal{C}_1$ satisfy $\Delta(\mathrm{dprod}_{X(\ell)}(z), \tilde{x}) \leq 1-\varepsilon$. By Lemma 2.8.3, there exists a $k \in I$ such that $\Delta(y, \tilde{y}) \leq 1/2 - \varepsilon/2 + \varepsilon^{C'/2}$. Thanks to this distance guarantee, the pair (z, y) will appear on the list $\mathcal{L}_k$ when the direct sum list decoding algorithm is run for this k. Then z will be on the combined list $\mathcal{L}$ and the trimmed list $\mathcal{L}'$, with the trimming ensuring that no elements of $\mathcal{C}_1$ appear on this list beyond those with the promised distance. The set $\{\mathrm{dprod}_{X(\ell)}(z) \mid z \in \mathcal{L}'\}$ thus contains all words in

$\mathrm{dprod}_{X(\ell)}(\mathcal{C}_1)$ with distance at most $(1-\varepsilon)$ from $\tilde{x}$.

To obtain the promised the running time, note that Algorithm 2.8.4 runs the direct sum list decoding algorithm $\sqrt{C'\ell\log(1/\varepsilon)}$ times and then computes the direct product lifting of each element of $\mathcal{L}$ in the trimming step. ∎

Combining the parameters in the reduction with those required for our direct sum list decoding algorithm, we obtain the following. Note that for very small values of ε, we can choose the constant C' to be close to 2, and we will be list decoding the direct sum code up to distance $1/2-\sqrt{\beta}\approx 1/2-\varepsilon/4$.

Corollary 2.8.6 (Direct Product List Decoding)**.** *Let $\varepsilon_0 < 1/2$ be a constant, and let $\varepsilon > 0$, $C' \geq 2 + 4/\log(1/\varepsilon)$, and $\beta = (\varepsilon/2 - \varepsilon^{C'/2})^2$. There exist universal constants $c, C > 0$ such that for any γ-HDX $X(\leq d)$ on ground set $[n]$ and Π_1 uniform, if*

$$\gamma \leq \log(1/\beta)^{-C\log(1/\beta)} \quad \text{and} \quad d \geq c\cdot\frac{\log(1/\beta)^2}{\beta},$$

then the following holds:

For every binary code $\mathcal{C}_1$ with $\Delta(\mathcal{C}_1)\geq 1/2-\varepsilon_0$ on $X(1)=[n]$, there exists a lifted code $\mathcal{C}_\ell = \mathrm{dprod}_{X(\ell)}(\varphi(\mathcal{C}_1))$ on C_ℓ where $\ell = O(\log(1/\beta))$, φ is an explicit linear projection, and

- *[Efficient List Decoding] If $\tilde{x}$ is $(1-\varepsilon)$-close to $\mathcal{C}_\ell$, then we can compute the list of all codewords of $\mathcal{C}_\ell$ that are $(1-\varepsilon)$-close to $\tilde{x}$ in time $n^{\varepsilon^{-O(1)}}\cdot f(n)$, where $f(n)$ is the running time of the unique decoding algorithm for $\mathcal{C}_1$.*
- *[Rate] The rate r_ℓ of $\mathcal{C}_\ell$ satisfies $r_\ell = r_1\cdot|X(1)|/(\ell\,|X(\ell)|)$, where r_1 is the relative rate of $\mathcal{C}_1$.*
- *[Linearity] If $\mathcal{C}_1$ is linear, then φ is the identity and $\mathcal{C}_\ell$ is linear.*

Proof. Let $k = \ell/2 - \sqrt{C'\ell\log(1/\varepsilon)}/2$. The choice of parameters ensures that $\mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ is list decodable up to distance $1/2-\sqrt{\beta} = 1/2-\varepsilon/2+\varepsilon^{C'/2}$ in running time $g(n) = n^{\beta^{-O(1)}}f(n)$

by Theorem 2.7.1 (noting that the bound on C' implies $\beta \geq \varepsilon^2/16$). Since increasing k increases the list decoding radius of the direct sum lifting, this holds for any value of k with $|k - \ell/2| \leq \sqrt{C'\ell \log(1/\varepsilon)}/2$. By Theorem 2.8.5, the direct product lifting $\mathrm{dprod}_{X(\ell)}(\mathcal{C}_1)$ is list decodable up to distance $(1-\varepsilon)$ in running time

$$\sqrt{C'\ell \log(1/\varepsilon)} n^{\beta^{-O(1)}} f(n) + |X(\ell)|\,|\mathcal{L}|\,.$$

The HDX has $|X(\ell)| \leq \binom{n}{\ell} = n^{O(\log(1/\beta))}$, and the list size $|\mathcal{L}|$ is bounded by the sum of the sizes of the lists $\mathcal{L}_k$ obtained from each direct sum decoding. Each of these lists has $|\mathcal{L}_k| \leq 1/(2\beta)$ by the Johnson bound (see Remark 2.6.14) and the number of lists is constant with respect to n, so the overall running time is dominated by the first term, $n^{\beta^{-O(1)}} f(n) = n^{\varepsilon^{-O(1)}} f(n)$.

The rate and linearity guarantees follow in the same manner as they do in Theorem 2.7.1, where the rate calculation requires a slight modification for dealing with the increased alphabet size and φ is the projection from Claim 2.7.6. ∎

Using Corollary 2.8.6 with HDXs obtained from Ramanujan complexes as in Corollary 2.7.2, we can perform list decoding with an explicit construction up to distance $(1-\varepsilon)$ with HDX parameters $d = O(\log(1/\varepsilon)^2/\varepsilon^2)$ and $\gamma = (\log(1/\varepsilon))^{-O(\log(1/\varepsilon))}$. The direct product list decoding algorithm of Dinur et al. [DHK+19] is based on a more general expanding object known as a double sampler. As the only known double sampler construction is based on a HDX, we can compare our parameters to their HDX requirements of $d = O(\exp(1/\varepsilon))$ and $\gamma = O(\exp(-1/\varepsilon))$.

2.9 Instantiation II: Direct Sum on Expander Walks

We instantiate the list decoding framework to the direct sum lifting where the sum is taken over the collection $X(k)$ of length k walks of a sufficiently expanding graph G. To stress the

different nature of this collection and its dependence on G we equivalently denote $X(k)$ by $W_G(k)$ and endow it with a natural measure in Definition 2.9.1.

Definition 2.9.1 (Walk Collection)**.** *Let $G = (V, E, w)$ be a weighted graph with weight distribution $w\colon E \to [0,1]$. For $k \in \mathbb{N}^+$, we denote by $W_G(k)$ the collection of all walks of length k in G, i.e.,*

$$W_G(k) \coloneqq \{w = (w_1, \ldots, w_k) \mid w \text{ is a walk of length } k \text{ in } G\}.$$

We endow $W_G(k)$ with the distribution Π_k arising from taking a random vertex w_1 according to the stationary distribution on V and then taking $k-1$ steps according to the normalized random walk operator of G.

One simple difference with respect to the HDX case is that now we are working with a collection of (ordered) tuples instead of subsets. The Propagation Rounding Algorithm 2.6.5 remains the same, but we need to establish the tensorial properties of $W_G(k)$ which is done in Section 2.9.1.

The main result of this section follows.

Theorem 2.9.2 (Direct Sum Lifting on Expander Walks)**.** *Let $\varepsilon_0 < 1/2$ be a constant and $\varepsilon \in (0, \varepsilon_0)$. There exists a universal constant $C > 0$ such that for any d-regular γ-two-sided expander graph G on ground set $W_G(1) = [n]$, if $\gamma \leq \varepsilon^C$, then the following holds:*

For every binary code $\mathcal{C}_1$ with $\Delta(\mathcal{C}_1) \geq 1/2 - \varepsilon_0$ on $W_G(1) = [n]$, there exists a binary lifted code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\varphi(\mathcal{C}_1))$ with $\Delta(\mathcal{C}_k) \geq 1/2 - \varepsilon^{\Omega_{\varepsilon_0}(1)}$ on $W_G(k)$ where $k = O(\log(1/\varepsilon))$, φ is an explicit linear projection, and

- *[Efficient List Decoding] If $\tilde{y}$ is $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$, then we can compute the list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ (c.f. Definition 2.6.15) in time*

$$n^{\varepsilon^{-O(1)}} \cdot f(n),$$

where $f(n)$ is the running time of a unique decoding algorithm for $\mathcal{C}_1$.

- *[Rate] The rate r_k of $\mathcal{C}_k$ satisfies $r_k = r_1/d^{k-1}$ where r_1 is the relative rate of $\mathcal{C}_1$.*
- *[Linearity] If $\mathcal{C}_1$ is linear, then φ is the identity and $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ is linear.*

In particular, we apply Theorem 2.9.2 to the explicit family of Ramanujan expanders of Lubotzky et al. from Theorem 2.9.3.

Theorem 2.9.3 (Lubotzky-Phillips-Sarnak abridged [LPS88])**.** *Let $p \equiv 1 \pmod 4$ be a prime. Then there exists an explicit infinite family of $(p+1)$-regular Ramanujan graphs $G_1, G_2, \ldots$ on $n_1 < n_2 < \cdots$ vertices, i.e., $\sigma_2(G_i) \leq 2 \cdot \sqrt{p}/(p+1)$.*

In order to construct Ramanujan expanders with arbitrarily good expansion, we will use the following lemma for finding primes.

Lemma 2.9.4 (From [TS17])**.** *For every $\alpha > 0$ and sufficiently large n, there exists an algorithm that given a and m relatively prime, runs in time $\mathrm{poly}(n)$ and outputs a prime number p with $p \equiv a \pmod m$ in the interval $[(1-\alpha)n, n]$.*

This results in Corollary 2.9.5.

Corollary 2.9.5. *Let $\varepsilon_0 < 1/2$ be a constant and $\varepsilon \in (0, \varepsilon_0)$. There is an infinite sequence of explict Ramanujan expanders $G_1, G_2, \ldots$ on ground sets of size $n_1 < n_2 < \cdots$ such that the following holds:*

For every sequence of binary codes $\mathcal{C}_1^{(i)}$ on $[n_i]$ with rate and distance uniformly bounded by $r_1^{(i)}$ and $(1/2 - \varepsilon_0)$ respectively, there exists a sequence of binary lifted codes $\mathcal{C}_k^{(i)} = \mathrm{dsum}_{X(k)}(\varphi(\mathcal{C}_1^{(i)}))$ on a collection $X_i(k)$ with distance $\left(1/2 - \varepsilon^{\Omega_{\varepsilon_0}(1)}\right)$ where φ is an explicit linear projection and

- *[Efficient List Decoding] If $\tilde{y}$ is $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$, then we can compute the list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ (c.f. Definition 2.6.15) in time $n^{\varepsilon^{-O(1)}} \cdot f(n)$, where $f(n)$ is the running time of a unique decoding algorithm of $\mathcal{C}_1$.*

- *[Explicit Construction] The collection $W_{G_i}(k)$ is obtained from length k walks on a Ramanujan d-regular expander G_i where $k = O(\log(1/\varepsilon))$, $d = 8 \cdot \varepsilon^{-O(1)}$ and $\gamma = \varepsilon^{O(1)}$.*
- *[Rate] The rate $r_k^{(i)}$ of $\mathcal{C}_k^{(i)}$ satisfies $r_k^{(i)} \geq r_1^{(i)} \cdot \varepsilon^{O(\log(1/\varepsilon))}$.*
- *[Linearity] If $\mathcal{C}_1^{(i)}$ is linear, then φ is the identity and $\mathcal{C}_k^{(i)} = \mathrm{dsum}_{X(k)}(\mathcal{C}_1^{(i)})$ is linear.*

Proof. Using Lemma 2.9.4 with $a = 1$ and $m = 4$, we see that given n, α, a prime p such that $p \equiv 1 \pmod 4$ may be found in the interval $[(1-\alpha)n, n]$ for large enough n. For Ramanujan expanders, the condition that $\gamma \leq \varepsilon^C$ translates to $p \geq 4 \cdot \varepsilon^{-2C}$. Choose $\alpha = 1/2$ and $n > 8 \cdot \varepsilon^{-2C}$ so that we find a prime greater than $4 \cdot \varepsilon^{-2C}$, but at most $8 \cdot \varepsilon^{-2C}$.

Based on this prime, we use the above Theorem 2.9.3 to get a family of Ramanujan graphs $G_1, G_2, \ldots$ with $n_1 < n_2 < \cdots$ vertices, such that the degree is bounded by $8\varepsilon^{-2C}$. Using the parameters of this family in Theorem 2.9.2, we obtain the desired claims. ∎

2.9.1 Expander Walks are Two-Step Tensorial

To apply the list decoding framework we need to establish the tensorial parameters of expander walks $W_G(k)$ for a γ-two-sided expander graph G. Although the tensorial property is precisely what the abstract list decoding framework uses, when faced with a concrete object such as $W_G(k)$ it will be easier to prove that it satisfies a *splittable* property defined in [AJT19] for complexes which implies the tensorial property. In turn, this splittable property is defined in terms of some natural operators denoted *Swap* operators whose definition is recalled in Section 2.9.1 in a manner tailored to the present case $X(k) = W_G(k)$. Then, in Section 2.9.1, we formally define the splittable property and show that the expansion of the Swap operator is controlled by the expansion parameter γ of G allowing us to deduce the splittable parameters of $W_G(k)$. Finally, in Section 2.9.1, we show how $W_G(k)$ being splittable gives the tensorial parameters. Some results are quite similar to the hypergraph case in [AJT19] (which built on [BRS11]). The key contribution in this new case of $W_G(k)$ is

observing the existence of these new Swap operators along with their expansion properties.

Emergence of Swap Operators

To motivate the study of Swap operators on $W_G(k)$, we show how they naturally emerge from the study of k-CSPs. The treatment is quite similar to the hypergraph case developed in [AJT19], but this will give us the opportunity to formalize the details that are specific to $W_G(k)$. Suppose that we solve a k-CSP instance as defined in Section 2.2.4 whose constraints were placed on the tuples corresponding to walks in $W_G(k)$. The result is a local PSD ensemble $\{\mathbf{Z}\}$ which can then be fed to the Propagation Rounding Algorithm 2.6.5. It is easy to show that the tensorial condition of Eq. (2.12) (below) is sufficient to guarantee an approximation to this k-CSP on $W_G(k)$ within μ additive error. The precise parameters are given in Section 2.9.1. For now, we take this observation for granted and use it to show how the Swap operators emerge in obtaining the inequality

$$\mathop{\mathbb{E}}_{\Omega} \mathop{\mathbb{E}}_{w\sim W_G(k)} \left\| \{\mathbf{Z}'_w\} - \left\{\mathbf{Z}'_{w_1}\right\} \cdots \left\{\mathbf{Z}'_{w_k}\right\} \right\|_1 \leq \mu \tag{2.12}$$

present in the definition of tensoriality.

The following piece of notation will be convenient when referring to sub-walks of a given walk.

Definition 2.9.6 (Sub-Walk)**.** *Given $1 \leq i \leq j \leq k$ and $w = (w_1, \ldots, w_k) \in W_G(k)$, we define the sub-walk $w(i,j)$ from w_i to w_j as*

$$w(i,j) \coloneqq (w_i, w_{i+1}, \ldots, w_j).$$

We will need the following simple observation about marginal distributions of Π_k on sub-walks.

Claim 2.9.7 (Marginals of the walk distribution)**.** *Let $k \in \mathbb{N}^+$ and $1 \leq i \leq j \leq k$. Then sampling $w \sim \Pi_k$ in $W_G(k)$ and taking $w(i,j)$ induces the distribution Π_{j-i+1} on $W_G(j-i+1)$.*

Proof. Let $w = (w_1, \ldots, w_i, \ldots, w_j, \ldots, w_k) \sim \Pi_k$. Since $w_1 \sim \Pi_1$ where Π_1 is the stationary measure of G and $w_2, \ldots, w_i$ are obtained by $(i-1)$ successive steps of a random walk on G, the marginal distribution on w_i is again the stationary measure Π_1. Then by taking $(j-i)$ successive random walk steps from w_i on G, we obtain a walk $(w_i, \ldots, w_j)$ distributed according to Π_{j-i+1}. ∎

We also need the notion of a *splitting tree* as follows.

Definition 2.9.8 (Splitting Tree [AJT19])**.** *We say that a binary tree $\mathcal{T}$ is a k-splitting tree if it has exactly k leaves and*

- *the root of $\mathcal{T}$ is labeled with k and all other vertices are labeled with positive integers,*
- *the leaves are labeled with 1, and*
- *each non-leaf vertex satisfies the property that its label is the sum of the labels of its two children.*

The Swap operators arise naturally from the following triangle inequality where the quantity $\mathbb{E}_{w \sim W_G(k)} \left\| \{\mathbf{Z}'_w\} - \prod_{i=1}^{k} \left\{ \mathbf{Z}'_{w(i)} \right\} \right\|_1$ is upper bounded by a sum of terms of the form

$$\mathop{\mathbb{E}}_{w \sim W_G(k_1+k_2)} \left\| \{\mathbf{Z}'_w\} - \left\{ \mathbf{Z}'_{w(1,k_1)} \right\} \left\{ \mathbf{Z}'_{w(k_1+1,k_2)} \right\} \right\|_1.$$

We view the above expectation as taking place over the edges $W_G(k_1+k_2)$ of a bipartite graph on vertex bipartition $(W_G(k_1), W_G(k_2))$. This graph gives rise to a Swap operator which we formally define later in Section 2.9.1. The following claim shows how a splitting tree defines all terms (and hence also their corresponding graphs and operators) that can appear in this upper bound.

Claim 2.9.9 (Triangle inequality)**.** *Let $k \in \mathbb{N}^+$ and $\mathcal{T}$ be a k-splitting tree. Then*

$$\mathop{\mathbb{E}}_{w\sim W_G(k)} \left\| \{\mathbf{Z}'_w\} - \prod_{i=1}^{k} \left\{\mathbf{Z}'_{w(i)}\right\} \right\|_1 \leq \sum_{(k_1,k_2)} \mathop{\mathbb{E}}_{w\sim W_G(k_1+k_2)} \left\| \{\mathbf{Z}'_w\} - \left\{\mathbf{Z}'_{w(1,k_1)}\right\}\left\{\mathbf{Z}'_{w(k_1+1,k_2)}\right\} \right\|_1,$$

where the sum $\sum_{(k_1,k_2)}$ is taken over all pairs of labels of the two children of each internal node of $\mathcal{T}$.

Proof. We prove the claim by induction on k. Let (k_1, k_2) be the labels of the children of the root of the splitting tree $\mathcal{T}$. Suppose $\mathcal{T}_1$ and $\mathcal{T}_2$ are the corresponding splitting trees rooted at these children with labels k_1 and k_2, respectively. By this choice, we have $k = k_1 + k_2$. Applying the triangle inequality yields

$$\begin{aligned}\mathop{\mathbb{E}}_{w\sim W_G(k)} \left\| \{\mathbf{Z}'_w\} - \prod_{i=1}^{k} \left\{\mathbf{Z}'_{w_i}\right\} \right\|_1 \leq & \mathop{\mathbb{E}}_{w\sim W_G(k)} \left\| \{\mathbf{Z}'_w\} - \left\{\mathbf{Z}'_{w(1,k_1)}\right\}\left\{\mathbf{Z}'_{w(k_1+1,k_2)}\right\} \right\|_1 + \\ & \mathop{\mathbb{E}}_{w\sim W_G(k)} \left\| \left\{\mathbf{Z}'_{w(1,k_1)}\right\}\left\{\mathbf{Z}'_{w(k_1+1,k_2)}\right\} - \prod_{i=1}^{k_1} \left\{\mathbf{Z}'_{w_i}\right\}\left\{\mathbf{Z}'_{w(k_1+1,k_2)}\right\} \right\|_1 + \\ & \mathop{\mathbb{E}}_{w\sim W_G(k)} \left\| \prod_{i=1}^{k_1} \left\{\mathbf{Z}'_{w_i}\right\}\left\{\mathbf{Z}'_{w(k_1+1,k_2)}\right\} - \prod_{i=1}^{k} \left\{\mathbf{Z}'_{w_i}\right\} \right\|_1.\end{aligned}$$

Using the marginalization given by Claim 2.9.7 on the second and third terms and simplifying, we get

$$\begin{aligned}\mathop{\mathbb{E}}_{w\sim W_G(k)} \left\| \{\mathbf{Z}'_w\} - \prod_{i=1}^{k} \left\{\mathbf{Z}'_{w_i}\right\} \right\|_1 \leq & \mathop{\mathbb{E}}_{w\sim W_G(k)} \left\| \{\mathbf{Z}'_w\} - \left\{\mathbf{Z}'_{w(1,k_1)}\right\}\left\{\mathbf{Z}'_{w(k_1+1,k_2)}\right\} \right\|_1 + \\ & \mathop{\mathbb{E}}_{w\sim W_G(k_1)} \left\| \left\{\mathbf{Z}'_w\right\} - \prod_{i=1}^{k_1} \left\{\mathbf{Z}'_{w_i}\right\} \right\|_1 + \\ & \mathop{\mathbb{E}}_{w\sim W_G(k_2)} \left\| \left\{\mathbf{Z}'_w\right\} - \prod_{i=1}^{k_2} \left\{\mathbf{Z}'_{w_i}\right\} \right\|_1.\end{aligned}$$

Applying the induction hypothesis to the second term with tree $\mathcal{T}_1$ and to the third term with tree $\mathcal{T}_2$ finishes the proof. ∎

Swap Operators Arising from Expander Walks

We define the Swap operator associated to walks on a given graph G as follows.

Definition 2.9.10 (Graph Walk Swap Operator)**.** *Let $G = (V, E, w)$ be a weighted graph. Let $k_1, k_2 \in \mathbb{N}^+$ be such that $k = k_1 + k_2$. We define the graph walk Swap operator*

$$\mathsf{S}^{\circ}_{k_1,k_2} \colon \mathbb{R}^{W_G(k_2)} \to \mathbb{R}^{W_G(k_1)}$$

such that for every $f \in \mathbb{R}^{W_G(k_2)}$,

$$\left(\mathsf{S}^{\circ}_{k_1,k_2}(f)\right)(w) \coloneqq \mathbb{E}_{w' : ww' \in W(k)}[f(w')],$$

where ww' denotes the concatenation of the walks w and w'. The operator $\mathsf{S}^{\circ}_{k_1,k_2}$ can be defined more concretely in matrix form such that for every $w \in W_G(k_1)$ and $w' \in W_G(k_2)$,

$$\left(\mathsf{S}^{\circ}_{k_1,k_2}\right)_{w,w'} \coloneqq \frac{\Pi_k(ww')}{\Pi_{k_1}(w)}.$$

Remark 2.9.11. *Swap operators are Markov operators, so the largest singular value of a Swap operator is bounded by* 1*.*

Unlike the Swap operators for HDXs described in [AJT19], which are defined using unordered subsets of hyperedges, the Swap operators $\mathsf{S}^{\circ}_{k_1,k_2}$ use sub-walks and are thus directed operators. Instead of analyzing such an operator directly, we will examine the symmetrized version

$$\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2}) = \begin{pmatrix} 0 & \mathsf{S}^{\circ}_{k_1,k_2} \\ \left(\mathsf{S}^{\circ}_{k_1,k_2}\right)^{\dagger} & 0 \end{pmatrix}$$

and show that $\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})$ is the normalized random walk operator of an undirected graph. In particular, $\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})$ defines an undirected weighted bipartite graph on the vertices $W_G(k_1) \cup W_G(k_2)$, where each edge ww' in this graph is weighted according to the transition probability

from one walk to the other whenever one of w, w' is in $W_G(k_1)$ and the other is in $W_G(k_2)$. This becomes clear when taking a closer look at the adjoint operator $(\mathsf{S}^{\circ}_{k_1,k_2})^{\dagger}$.

Claim 2.9.12. *Let $k_1, k_2 \in \mathbb{N}$ and $k = k_1 + k_2$. Define the operator $\mathfrak{S}_{k_1,k_2,\cdot} \colon \mathbb{R}^{W_G(k_1)} \to \mathbb{R}^{W_G(k_2)}$ such that for every $f \in \mathbb{R}^{W_G(k_1)}$,*

$$\left(\mathfrak{S}_{k_1,k_2,(}f)\right)(w') \coloneqq \mathbb{E}_{w:ww' \in W(k)}[f(w)]$$

for every $w' \in W_G(k_2)$. Then

$$\left(\mathsf{S}^{\circ}_{k_1,k_2}\right)^{\dagger} = \mathfrak{S}_{k_1,k_2,\cdot}$$

Proof. Let $f \in C^{W_G(k_1)}$ and $g \in C^{W_G(k_2)}$. We show that $\left\langle f, \mathsf{S}^{\circ}_{k_1,k_2} g \right\rangle = \left\langle \mathfrak{S}_{k_1,k_2,f}, g \right\rangle$. On one hand we have

$$\begin{aligned}
\left\langle f, \mathsf{S}^{\circ}_{k_1,k_2} g \right\rangle &= \mathbb{E}_{w \in W_G(k_1)} \left[f(w) \mathbb{E}_{w' : ww' \in W_G(k)}[g(w')] \right] \\
&= \mathbb{E}_{w \in W_G(k_1)} \left[f(w) \sum_{w' \in W_G(k_2)} \frac{\Pi_k(ww')}{\Pi_{k_1}(w)} g(w') \right] \\
&= \sum_{w \in W_G(k_1)} \Pi_{k_1}(w) f(w) \sum_{w' \in W_G(k_2)} \frac{\Pi_k(ww')}{\Pi_{k_1}(w)} g(w') \\
&= \sum_{ww' \in W_G(k)} f(w) g(w') \Pi_k(ww').
\end{aligned}$$

On the other hand we have

$$\begin{aligned}\left\langle \mathfrak{S}_{k_1,k_2,f}, g \right\rangle &= \mathbb{E}_{w' \in W_G(k_2)} \left[\mathbb{E}_{w : ww' \in W_G(k)} [f(w)] g(w') \right] \\ &= \mathbb{E}_{w' \in W_G(k_2)} \left[\sum_{w \in W_G(k_1)} \frac{\Pi_k(ww')}{\Pi_{k_2}(w')} f(w) g(w') \right] \\ &= \sum_{w' \in W_G(k_2)} \Pi_{k_2}(w') \sum_{w \in W_G(k_1)} \frac{\Pi_k(ww')}{\Pi_{k_2}(w')} f(w) g(w') \\ &= \sum_{ww' \in W_G(k)} f(w) g(w') \Pi_k(ww').\end{aligned}$$

Hence, $\mathfrak{S}_{k_1,k_2,=} (\mathsf{S}^{\circ}_{k_1,k_2})^{\dagger}$ as claimed. ∎

Swap Operators are Splittable

At a high level, the expansion of a certain collection of Swap walks $\mathsf{S}^{\circ}_{k_1,k_2}$ ensures that we can round the SOS solution and this gives rise to the *splittable* notion, which we tailor to the $W_G(k)$ case after recalling some notation.

Remark 2.9.13. *We establish the definitions in slightly greater generality than needed for our coding application since this generality is useful for solving k-CSP instances on $W_G(k)$ for more general graphs G that are not necessarily expanders (c.f. Section 2.9.1). Solving these kinds of k-CSPs might be of independent interest. For the coding application, the threshold rank (Definition 2.9.14) will be one, i.e., we will be working with expander graphs.*

Definition 2.9.14 (Threshold Rank of Graphs (from [BRS11]))**.** *Let $G = (V, E, w)$ be a weighted graph on n vertices and A be its normalized random walk matrix. Suppose the eigenvalues of A are $1 = \lambda_1 \geq \cdots \geq \lambda_n$. Given a parameter $\tau \in (0, 1)$, we denote the threshold rank of G by $\mathrm{rank}_{\geq \tau}(\mathsf{A})$ (or $\mathrm{rank}_{\geq \tau}(G)$) and define it as*

$$\mathrm{rank}_{\geq \tau}(\mathsf{A}) \coloneqq |\{i \mid \lambda_i \geq \tau\}|.$$

Let $\mathrm{Swap}(\mathcal{T}, W_G(\leq k))$ be the set of all swap graphs over $W_G(\leq k)$ finding representation in the splitting tree $\mathcal{T}$, i.e., for each internal node with leaves labeled k_1 and k_2 we associate the undirected Swap operator $\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})$.

Given a threshold parameter $\tau \leq 1$ and a set of normalized adjacency matrices $\mathcal{A} = \{\mathsf{A}_1, \ldots, \mathsf{A}_s\}$, we define the threshold rank $\mathrm{rank}_{\geq\tau}(\mathcal{A})$ of $\mathcal{A}$ as

$$\mathrm{rank}_{\geq\tau}(\mathcal{A}) \coloneqq \max_{\mathsf{A}\in\mathcal{A}} \mathrm{rank}_{\geq\tau}(\mathsf{A}),$$

where $\mathrm{rank}_{\geq\tau}(\mathsf{A})$ denotes the usual threshold rank of A as in Definition 2.9.14.

Definition 2.9.15 ($(\mathcal{T},\tau,r)$-splittability [AJT19])**.** *A collection $W_G(\leq k)$ is said to be $(\mathcal{T},\tau,r)$-splittable if $\mathcal{T}$ is a k-splitting tree and*

$$\mathrm{rank}_{\geq\tau}(\mathrm{Swap}(\mathcal{T}, W_G)) \leq r.$$

If there exists some k-splitting tree $\mathcal{T}$ such that $W_G(\leq k)$ is $(\mathcal{T},\tau,r)$-splittable, the instance $W_G(\leq k)$ will be called a (τ,r)-splittable instance.

We show that the expansion of $\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})$ is inherited from the expansion of its defining graph G. To this end we will have to overcome the hurdle that $W_G(k) \subseteq V^k$ is not necessarily a natural product space, but it can be made so with the proper representation.

Lemma 2.9.16. *Let $G = (V = [n], E)$ be a d-regular graph with normalized random walk operator A_G. Then for every $k_1, k_2 \in \mathbb{N}^+$, there are representations of $\mathsf{S}^{\circ}_{k_1,k_2}$ and A_G as matrices such that*

$$\mathsf{S}^{\circ}_{k_1,k_2} = \mathsf{A}_G \otimes \mathsf{J}/d^{k_2-1},$$

where $\mathsf{J} \in \mathbb{R}^{[d]^{k_1-1}\times[d]^{k_2-1}}$ is the all ones matrix.

Proof. Partition the set of walks $W_G(k_1)$ into the sets $W_1, \ldots, W_n$, where $w \in W_i$ if the last vertex of the walk is $w_{k_1} = i$. Similarly, partition $W_G(k_2)$ into the sets $W'_1, \ldots, W'_n$, where

$w' \in W'_j$ if the first vertex of the walk is $w'_1 = j$. Note that $|W_i| = d^{k_1-1}$ for all i and $\left|W'_j\right| = d^{k_2-1}$ for all j.

Now order the rows of the matrix $\mathsf{S}^{\circ}_{k_1,k_2}$ so that all of the rows corresponding to walks in W_1 appear first, followed by those for walks in W_2, and so on, with an arbitrary order within each set. Do a similar re-ordering of the columns for the sets $W'_1, \ldots, W'_n$. Observe that

$$\left(\mathsf{S}^{\circ}_{k_1,k_2}\right)_{w,w'} = \frac{\Pi_{k_1+k_2}(ww')}{\Pi_{k_1}(w)} = \frac{\mathbf{1}\left[w_{k_1} \text{ is adjacent to } w'_1\right]}{d^{k_2-1}},$$

which only depends on the adjacency of the last vertex of w and the first vertex of w'. If the vertices i and j are adjacent, then $\left(\mathsf{S}^{\circ}_{k_1,k_2}\right)_{w,w'} = 1/d^{k_2-1}$ for every $w \in W_i$ and $w' \in W'_j$; otherwise, $\left(\mathsf{S}^{\circ}_{k_1,k_2}\right)_{w,w'} = 0$. Since the walks in the rows and columns are sorted according to their last and first vertices, respectively, the matrix $\mathsf{S}^{\circ}_{k_1,k_2}$ exactly matches the tensor product $\mathsf{A}_G \otimes \mathsf{J}/d^{k_2-1}$, where the rows and columns of A_G are sorted according to the usual ordering on $[n]$. ∎

Corollary 2.9.17. *Let $G = (V, E)$ be a γ-two-sided spectral expander with normalized random walk operator A_G. Then for every $k_1, k_2 \in \mathbb{N}^+$,*

$$\lambda_2(\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})) \leq \gamma.$$

Proof. To make the presentation reasonably self-contained, we include the proof of the well-known connection between the singular values of $\mathsf{S}^{\circ}_{k_1,k_2}$ and the eigenvalues of $\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})$. Using Lemma 2.9.16 and the fact that $\sigma_i(\mathsf{A}_G \otimes \mathsf{J}/d^{k_2-1}) = \sigma_i(\mathsf{A}_G)$, we have $\sigma_i(\mathsf{S}^{\circ}_{k_1,k_2}) = \sigma_i(\mathsf{A}_G)$. Since

$$\left(\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})^{\dagger}\right)\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2}) = \begin{pmatrix} \mathsf{S}^{\circ}_{k_1,k_2}\left(\mathsf{S}^{\circ}_{k_1,k_2}\right)^{\dagger} & 0 \\ 0 & \left(\mathsf{S}^{\circ}_{k_1,k_2}\right)^{\dagger}\mathsf{S}^{\circ}_{k_1,k_2} \end{pmatrix},$$

the nonzero singular values of $\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})$ are the same as the nonzero singular values of $\mathsf{S}^{\circ}_{k_1,k_2}$.

As $\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})$ is the random walk operator of a bipartite graph, the spectrum of $\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})$ is symmetric around 0 implying that its nonzero eigenvalues are

$$\pm\sigma_1(\mathsf{S}^{\circ}_{k_1,k_2}), \pm\sigma_2(\mathsf{S}^{\circ}_{k_1,k_2}), \ldots = \pm\sigma_1(\mathsf{A}_G), \pm\sigma_2(\mathsf{A}_G), \ldots$$

Hence, the second-largest of these is $\lambda_2(\mathcal{U}(\mathsf{S}^{\circ}_{k_1,k_2})) = \sigma_2(\mathsf{A}_G) \leq \gamma$. ∎

Applying this spectral bound on $\mathcal{U}(\mathsf{S}^{\circ}_{k,k})$ to each internal node of any splitting tree readily gives the splittability of $W_G(k)$.

Corollary 2.9.18. *If G is a γ-two-sided spectral expander, then for every $k \in \mathbb{N}^+$ the collection $W_G(k)$ endowed with Π_k is $(\gamma, 1)$-splittable (for all choices of splitting trees).*

Splittable Implies Tensorial

By a simple adaptation of an argument in [AJT19] for hypergraphs which built on [BRS11], we can use the splittable property to obtain tensorial properties for $W_G(k)$. More precisely, we can deduce Theorem 2.9.19.

Theorem 2.9.19 (Adapted from [AJT19])**.** *Suppose $W_G(\leq k)$ with $W_G(1) = [n]$ and an $(L+2k)$-local PSD ensemble $\mathbf{Z} = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ are given. There exist some universal constants $c_4 \geq 0$ and $C'' \geq 0$ satisfying the following: If $L \geq C'' \cdot (q^{4k} \cdot k^7 \cdot r/\mu^5)$, $\mathrm{Supp}(\mathbf{Z}_j) \leq q$ for all $j \in [n]$, and $W_G(\leq k)$ is $(c_4 \cdot (\mu/(4k \cdot q^k))^2, r)$-splittable, then*

$$\mathop{\mathbb{E}}_{\Omega} \mathop{\mathbb{E}}_{w \sim W_G(k)} \left\| \{\mathbf{Z}'_w\} - \left\{\mathbf{Z}'_{w_1}\right\} \cdots \left\{\mathbf{Z}'_{w_k}\right\} \right\|_1 \leq \mu, \tag{2.13}$$

where $\mathbf{Z}'$ is as defined in Algorithm 2.6.5 on the input of $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ and Π_k.

Using Theorem 2.9.19, we can establish conditions on a γ-two-sided expander graph $G = (V, E, w)$ in order to ensure that $W_G(k)$ is (μ, L)-two-step tensorial.

Lemma 2.9.20 (Expander walks are two-step tensorial)**.** *There exist some universal constants $c' \geq 0$ and $C' \geq 0$ satisfying the following: If $L \geq c' \cdot (q^{4k} \cdot k^7/\mu^5)$, $\mathrm{Supp}(\mathbf{Z}_j) \leq q$ for all $j \in [n]$, and G is a γ-two-sided expander for $\gamma \leq C' \cdot \mu^2/\left(k^2 \cdot q^{2k}\right)$ and size $\geq k$, then $W_G(k)$ is (μ, L)-two-step tensorial.*

Proof. The proof is similar to the proof of Lemma 2.7.4 for HDXs, so we omit it. ∎

Interlude: Approximating k-CSP on Walk Constraints

Now, we digress to show how using Theorem 2.9.19 it is possible to deduce parameters for approximating k-CSPs on $W_G(k)$. We believe this result might be of independent interest and note that it is not required in the list decoding application.

Corollary 2.9.21. *Suppose $\mathfrak{I}$ is a q-ary k-CSP instance with constraints on $W_G(k)$. There exist absolute constants $C'' \geq 0$ and $c_4 \geq 0$ satisfying the following:*

If $W_G(k)$ is $(c_4 \cdot (\mu/(4k \cdot q^k))^2, r)$-splittable, then there is an algorithm that runs in time $n^{O\left(q^{4k} \cdot k^7 \cdot r/\mu^5\right)}$ based on $(C'' \cdot k^5 \cdot q^k \cdot r/\mu^4)$-levels of SOS-hierarchy and Algorithm 2.6.5 that outputs a random assignment $\xi : [n] \to [q]$ that in expectation ensures $\mathsf{SAT}_{\mathfrak{I}}(\xi) = \mathsf{OPT}(\mathfrak{I}) - \mu$.

Proof. The algorithm will just run Algorithm 2.6.5 on the local PSD-ensemble $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ given by the SDP relaxation of $\mathfrak{I}$ strengthened by $L = (C'' \cdot k^5 \cdot q^{2k}/\mu^4)$-levels of SOS-hierarchy and Π_k, where $C'' \geq 0$ is the constant from Theorem 2.9.19. $\mathbf{Z}$ satisfies

$$\mathsf{SDP}(\mathfrak{I}) = \mathop{\mathbb{E}}_{w \sim \Pi_k} \left[\mathop{\mathbb{E}}_{\{\mathbf{Z}_w\}} [\mathbf{1}[\mathbf{Z}_w \in \mathcal{P}_w]] \right] \geq \mathsf{OPT}(\mathfrak{I}). \tag{2.14}$$

Since the conditioning done on $\{\mathbf{Z}'\}$ is consistent with the local distribution, by law of total expectation and Eq. (2.14) we have

$$\mathop{\mathbb{E}}_{\Omega} \mathop{\mathbb{E}}_{w \sim \Pi_k} \mathbf{1}[\mathbf{Z}'_w \in \mathcal{P}_w] = \mathsf{SDP}(\mathfrak{I}) \geq \mathsf{OPT}(\mathfrak{I}). \tag{2.15}$$

By Theorem 2.9.19 we know that

$$\mathop{\mathbb{E}}_{\Omega} \mathop{\mathbb{E}}_{w\sim\Pi_k} \left\| \{\mathbf{Z}'_w\} - \{\mathbf{Z}'_{w_1}\} \cdots \{\mathbf{Z}'_{w_k}\} \right\|_1 \leq \mu. \tag{2.16}$$

Now, the fraction of constraints satisfied by the algorithm in expectation is

$$\mathop{\mathbb{E}}_{\xi}[\mathsf{SAT}_{\mathfrak{J}}(\xi)] = \mathop{\mathbb{E}}_{\Omega} \mathop{\mathbb{E}}_{w\sim\Pi_k} \mathop{\mathbb{E}}_{(\xi_1,\ldots,\xi_n)\sim\{\mathbf{Z}'_1\}\cdots\{\mathbf{Z}'_n\}} [\mathbf{1}[\xi|_w \in \mathcal{P}_w]].$$

By using Eq. (2.16), we can obtain

$$\mathop{\mathbb{E}}_{\xi}[\mathsf{SAT}_{\mathfrak{J}}(\xi)] \geq \mathop{\mathbb{E}}_{\Omega} \left[\mathop{\mathbb{E}}_{\{\mathbf{Z}_w\}} \mathbf{1}[\mathbf{Z}'_w \text{ satisfies the constraint on } w] \right] - \mu.$$

Using Eq. (2.15), we conclude

$$\mathop{\mathbb{E}}_{\xi}[\mathsf{SAT}_{\mathfrak{J}}(\xi)] \geq \mathsf{SDP}(\mathfrak{I}) - \mu = \mathsf{OPT}(\mathfrak{I}) - \mu.$$

■

2.9.2 Instantiation to Linear Base Codes

We instantiate the list decoding framework to the direct sum lifting given by the collection $W_G(k)$ of length k walks on a sufficiently expanding graph $G = (V, E, w)$. For parity sampling of expander walks, we will rely on the following fact.

Theorem 2.9.22 (Walks on Expanders are Parity Samplers [TS17] (Restatement of Theorem 2.4.1))**.** *Suppose G is a graph with second-largest eigenvalue in absolute value at most λ, and let $X(k)$ be the set of all walks of length k on G. Then $X(k)$ is a $(\beta_0, (\beta_0 + 2\lambda)^{\lfloor k/2 \rfloor})$-parity sampler. In particular, for any $\beta > 0$, if $\beta_0 + 2\lambda < 1$ and k is sufficiently large, then $X(k)$ is a (β_0, β)-parity sampler.*

First, we instantiate the framework to linear codes which already encompasses most of the ideas need for general binary codes.

Lemma 2.9.23 (Direct sum lifting of linear biased codes II)**.** *Let $\varepsilon_0 < 1/2$ be a constant and $\varepsilon \in (0, \varepsilon_0)$. There exists a universal constant $C > 0$ such that for any d-regular γ-two-sided expander graph G on ground set $W_G(1) = [n]$, if $\gamma \leq \varepsilon^C$, then the following holds:*

For every binary $2\varepsilon_0$-biased linear code $\mathcal{C}_1$ on $W_G(1) = [n]$, there exists a 2ε-biased binary lifted linear code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ on $W_G(k)$ where $k = O(\log(1/\varepsilon))$ and

- *[Efficient List Decoding] If $\tilde{y}$ is $(1/2 - \sqrt{\varepsilon})$-close to $\mathcal{C}_k$, then we can compute the list $\mathcal{L}(\tilde{y}, \mathcal{C}_1, \mathcal{C}_k)$ (c.f. Definition 2.6.15) in time*

$$n^{\varepsilon^{-O(1)}} \cdot f(n),$$

  where $f(n)$ is the running time of a unique decoding algorithm for $\mathcal{C}_1$.

- *[Rate] The rate r_k of $\mathcal{C}_k$ satisfies $r_k = r_1/d^{k-1}$ where r_1 is the relative rate of $\mathcal{C}_1$.*

- *[Linear] The lifted code $\mathcal{C}_k$ is linear.*

Proof. The proof is analogous to the one given in Lemma 2.7.5. We want to define parameters for a γ-two-sided expander $G = (V, E, w)$ so that $W_G(k)$ satisfies strong enough *robust* and *tensorial* assumptions and we can apply Theorem 2.6.17. In this application, we will rely on parity sampling for robustness. If $\mathrm{dsum}_{W_G(k)}$ is a $(2\varepsilon_0, 2\varepsilon)$-parity sampler, using the linearity of $\mathcal{C}_1$, we obtain a lifted code $\mathcal{C}_k = \mathrm{dsum}_{X(k)}(\mathcal{C}_1)$ which is linear and has bias 2ε; thus the lifting is indeed $(1/2-\varepsilon_0, 1/2-\varepsilon)$-robust. If we want to fully rely on parity sampling in Theorem 2.6.17, the lifting must be a $(\beta_0 = 1/2+\varepsilon_0, \beta = 2\varepsilon)$-parity sampler, which is more stringent than the first parity sampling requirement.[21] To invoke Theorem 2.9.22 and obtain

21. Recall that this strengthening is used in our list decoding framework.

this (β_0, β)-parity sampler, we need to choose a parameter θ (where $0 < \theta < (1-\beta_0)/\beta_0$) such that

$$k \geq 2 \cdot \log_{(1+\theta)\beta_0}(\beta) + 2 \text{ and}$$
$$\gamma \leq \frac{\theta \cdot \beta_0}{2},$$

which will ensure that

$$(\beta_0 + 2\gamma)^{\lfloor k/2 \rfloor} \leq ((1+\theta)\beta_0)^{\lfloor k/2 \rfloor} \leq \beta.$$

To get a (μ, L)-tensorial collection of walks, Lemma 2.9.20 requires

$$L \geq \frac{c' \cdot 2^{4k} \cdot k^7}{\mu^5} \quad \text{and} \quad \gamma \leq \frac{C' \cdot \mu^2}{k^2 \cdot 2^{2k}}.$$

where we used that our alphabet is binary (i.e., $q = 2$) and $c', C' > 0$ are constants. Finally, Theorem 2.6.17 requires $\mu \leq \varepsilon^8/2^{22}$. The conceptual part of the proof is essentially complete and we are left to compute parameters. We choose $\theta = 1/2 - \varepsilon_0$, so that provided $\varepsilon_0 < 1/2$ we have $(1+\theta)\beta_0 = 3/4 + \varepsilon_0 - \varepsilon_0^2 < 1$. Combining the parity sampling and tensorial requirements and after some simplification, the expansion γ is constrained as

$$\gamma \leq C'' \cdot \min\left(\frac{\varepsilon^{16}}{k^2 \cdot 2^{2k}}, \left(1/4 - \varepsilon_0^2\right)\right),$$

where $C'' > 0$ is a constant. We deduce that taking γ as

$$\gamma \leq C'' \cdot \frac{\left(1/4 - \varepsilon_0^2\right) \cdot \varepsilon^{16}}{k^2 \cdot 2^{2k}}$$

is sufficient. Further simplifying the above bound gives γ as in the statement of the theorem.

Now, we turn to the SOS related parameter L which is constrained to be

$$L \geq c'' \cdot \frac{2^{4k} \cdot k^7}{\varepsilon^{40}},$$

where $c'' > 0$. Note that in this case the exponent $O(L + k)$ appearing in the running time of Theorem 2.6.17 becomes $O(L)$. Further simplification of the bound on L leads to a running time of $n^{\varepsilon^{-O(1)}} \cdot f(n)$ as in the statement of the theorem. ∎

2.9.3 Instantiation to General Base Codes

The proof of Theorem 2.9.2 follows from Lemma 2.9.23 in the same way that Theorem 2.7.7 follows from Lemma 2.7.5 in the case of HDXs.

2.10 Auxiliary Results

2.10.1 Basic Facts of Probability

In this section, we collect some basic facts of probability used in the text.

Fact 2.10.1 (First Moment Bound)**.** *Let* $\mathbf{R}$ *be a random variable in* $[0, 1]$ *with* $\mathbb{E}[\mathbf{R}] = \alpha$*. Let* $\beta \in (0, 1)$ *be an arbitrary approximation parameter. Then*

$$\mathbb{P}[\mathbf{R} \geq (1 - \beta) \cdot \alpha] \geq \beta \cdot \alpha.$$

In particular,

$$\mathbb{P}\left[\mathbf{R} \geq \frac{\alpha}{2}\right] \geq \frac{\alpha}{2}.$$

Fact 2.10.2 (Chernoff Bound [MU17])**.** *Let* $\mathbf{R}_1, \ldots, \mathbf{R}_n$ *be independent and identically dis-*

tributed random variables where $\mathbf{R}_i$ *is uniformly distributed on* $\{\pm 1\}$*. For every* $a > 0$*,*

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} \mathbf{R}_i\right| \geq a\right] \leq 2 \cdot \exp\left(-\frac{a^2}{2n}\right).$$

Fact 2.10.3 (Hoeffding Bound [MU17])**.** *Let* $\mathbf{R}_1, \ldots, \mathbf{R}_n$ *be independent random variables such that* $\mathbb{E}[\mathbf{R}_i] = \mu$ *and* $\mathbb{P}[a \leq \mathbf{R}_i \leq b] = 1$ *for* $i \in [n]$*. For every* $\beta > 0$*,*

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{i=1}^{n} \mathbf{R}_i - \mu\right| \geq \beta\right] \leq 2 \cdot \exp\left(-\frac{2 \cdot \beta^2 \cdot n}{(a-b)^2}\right).$$

2.10.2 Further Properties of Liftings

We show that a uniformly random odd function $g\colon \{\pm 1\}^k \to \{\pm 1\}$ yields a parity lifting w.v.h.p. in k. Thus, parity liftings abound and we are not restricted to k-XOR in the framework. In fact, SOS abstracts the specific combinatorial properties of the lifting function being able to handle them in a unified way.

Lemma 2.10.4. *Let* $k \in \mathbb{N}^+$ *be odd. For every* $p, \beta, \theta > 0$ *satisfying* $\theta \geq \sqrt{\log(2/\beta)}/\sqrt{pk}$*,*

$$\mathop{\mathbb{P}}_{g}\left[\left|\mathbb{E}_{x \sim \mathrm{Bern}(p)^{\otimes k}}\left[g(\chi^{\otimes k}(x))\right]\right| \geq \beta\right] \leq 2 \cdot k \cdot \exp\left(-\beta^2 \cdot \binom{k}{\lfloor (1-\theta)pk \rfloor}/8\right),$$

where $g\colon \{\pm 1\}^k \to \{\pm 1\}$ *is a uniformly random odd function and* $\chi\colon (\mathbb{F}_2, +) \to (\{\pm 1\}, \cdot)$ *is the non-trivial character.*

Proof. It is enough to consider $p \in (0, 1/2]$ since the case $p \in [1/2, 1)$ can be reduced to the current case by taking the complement of the bit strings appearing in this analysis. Applying the Hoeffding bound Fact 2.10.3 yields

$$\begin{aligned}\mathbb{E}_{x \sim \mathrm{Bern}(p)^{\otimes k}}[g(x)] &= \mathbb{E}_{w \sim \mathrm{Binom}(k,p)}\left[g(\chi^{\otimes k}(x))\mathbf{1}_{w \in [pk \pm C \cdot pk]}\right] \pm 2 \cdot \exp(-C^2) \\ &= \mathbb{E}_{w \sim \mathrm{Binom}(k,p)}\left[g(\chi^{\otimes k}(x))\mathbf{1}_{w \in [pk \pm C \cdot pk]}\right] \pm \frac{\beta}{2},\end{aligned}$$

where the last equality follows from choosing $C = \theta\sqrt{pk}$ and the assumption that $\theta \geq \sqrt{\log(2/\beta)}/\sqrt{pk}$.

Since $p \leq 1/2$, $\ell = \binom{k}{\lfloor(1-\theta)\cdot p\cdot k\rfloor}$ is a lower bound on the number of binary strings of the Boolean k-hypercube in a single layer of Hamming weight in the interval $[pk \pm C \cdot pk]$. A second application of the Hoeffding bound Fact 2.10.3 gives that the bias within this layer is

$$\mathop{\mathbb{P}}_{g}\left[\left|\mathbb{E}_{x\in\mathbb{F}_2^k\colon \|x\|=\ell}\left[g(\chi^{\otimes k}(x))\right]\right| \geq \beta/2\right] \leq 2\cdot\exp\left(\beta^2\cdot\ell/8\right).$$

By union bound over the layers the result follows. ∎

2.10.3 Derandomization

We show how to derandomize the list decoding framework (which amounts to derandomize Algorithm 2.6.29) when the lifting function is a parity sampler and it satisfies a bounded degree condition (cf Eq. (2.17)). We observe that this is the setting of our two concrete instantiations, namely, for HDXs and expander walks. In the former case, we work with D-flat distributions and in the latter case with walk length and graph degree that are both functions of ε. Roughly speaking, we show that replacing a random sample by the majority works as long as parity sampling is sufficiently strong.

Lemma 2.10.5 (Majority Word)**.** *Let $z^* \in \{\pm 1\}^{X(1)}$ where $X(1) = [n]$. Suppose that $y^* = \mathrm{dsum}_{X(k)}(z^*)$ satisfy*

$$\mathbb{E}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}}\left[\left|\mathbb{E}_{\mathfrak{s}\sim\Pi_k} y^*_{\mathfrak{s}}\cdot\mathrm{dsum}(z)_{\mathfrak{s}}\right|\right] \geq 3\cdot\varepsilon,$$

and

$$\mathop{\mathbb{P}}_{\mathfrak{s}\sim\Pi_k}[\mathfrak{s}\ni i] \leq \frac{g(\varepsilon)}{n}. \tag{2.17}$$

If also $\mathrm{dsum}_{X(k)}$ is a $(1-\xi, 2\varepsilon)$-parity sampler for some $\xi \in (0,1)$, $\xi \geq 2\exp\left(-C\cdot\varepsilon^2\cdot g(\varepsilon)^2\cdot n\right) =$

$o_n(1)$ *where* $C > 0$ *is an universal constant and* $\xi \geq 1/(n(1-\xi-o_n(1)))$*, then*

$$\left|\mathbb{E}_{i\in[n]} z_i^* \cdot z_i'\right| \geq 1 - 7\sqrt{\xi},$$

where $z' \in \{\pm 1\}^n$ *is the majority defined as* $z_i' = \operatorname{argmax}_{b\in\{\pm1\}} \Pr_{\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}}[\mathbf{Z}_i = b]$.

Proof. Define $f(z) \coloneqq \left|\mathbb{E}_{\mathfrak{s}\sim\Pi_k} y_{\mathfrak{s}}^* \cdot \operatorname{dsum}(z)_{\mathfrak{s}}\right|$. Then, using Eq. (2.17) we claim that $f(z)$ is $O(g(\varepsilon)/n)$-Lipschitz with respect to ℓ_1 since

$$|f(z) - f(\tilde{z})| \leq \sum_{i\in X(1)} 2 \cdot \mathop{\mathbb{P}}_{\mathfrak{s}\sim\Pi_k} [\mathfrak{s} \ni i] \cdot |z_i - \tilde{z}_i| \leq O\left(\frac{g(\varepsilon)}{n}\right) \cdot \|z - \tilde{z}\|_1 .$$

Since the underlying distribution of $\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}$ is a product distribution on $\{\pm1\}^n$ and f is $O(g(\varepsilon)/n)$-Lipschitz, applying Hoeffding's inequality yields

$$\mathop{\mathbb{P}}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} [f(z) \leq \varepsilon] \leq \mathop{\mathbb{P}}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} \left[\left|f(z) - \mathbb{E}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} f(z)\right| \geq \varepsilon\right] \leq \exp\left(-\Theta(g'(\varepsilon)\cdot n)\right),$$

where $g'(\varepsilon) = \varepsilon^2 \cdot g(\varepsilon)^2$.

Using the assumption that dsum is a $(1-\xi, 2\varepsilon)$-parity sampler, we obtain

$$\mathbb{E}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} [|\langle z^*, z\rangle|] \geq 1 - \xi - 2\exp\left(-\Theta(g'(\varepsilon)\cdot n)\right).$$

By Jensen's inequality,

$$\mathbb{E}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} \left[\langle z^*, z\rangle^2\right] \geq \left(\mathbb{E}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} [|\langle z^*, z\rangle|]\right)^2 \geq (1 - \xi - 2\exp\left(-\Theta(g'(\varepsilon)\cdot n)\right))^2.$$

Using indepdence, we get

$$\mathbb{E}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} \left[\mathbb{E}_{i,j\in[n]} z_i^* z_i z_j z_j^*\right] \leq \mathbb{E}_{i,j\in[n]} z_i^* \mathbb{E}[z_i] \mathbb{E}\left[z_j\right] z_j^* + \frac{1}{n} = \left(\mathbb{E}_{i\in[n]} z_i^* \mathbb{E}[z_i]\right)^2 + \frac{1}{n}.$$

Thus, in particular $\left|\mathbb{E}_{i\in[n]} z_i^* \mathbb{E}\left[z_i\right]\right| \geq (1-\xi-o_n(1)) - 1/((1-\xi-o_n(1))n) \geq 1-3\xi$ which implies

$$\begin{aligned} 1-3\xi &\leq \left|\mathbb{E}_{i\in[n]} z_i^* \left(\Pr_{|(S,\sigma)}\left[\mathbf{Z}_i = 1\right] - \Pr_{|(S,\sigma)}\left[\mathbf{Z}_i = -1\right]\right)\right| \\ &\leq \mathbb{E}_{i\in[n]} \left|\Pr_{|(S,\sigma)}\left[\mathbf{Z}_i = 1\right] - \Pr_{|(S,\sigma)}\left[\mathbf{Z}_i = -1\right]\right|. \end{aligned}$$

Since

$$\mathbb{E}_{i\in[n]} 1 - \left|\Pr_{|(S,\sigma)}\left[\mathbf{Z}_i = 1\right] - \Pr_{|(S,\sigma)}\left[\mathbf{Z}_i = -1\right]\right| \leq 3\xi,$$

Markov's inequality yields

$$\mathbb{P}_{i\in[n]}\left[1-\sqrt{\xi} \geq \left|\Pr_{|(S,\sigma)}\left[\mathbf{Z}_i = 1\right] - \Pr_{|(S,\sigma)}\left[\mathbf{Z}_i = -1\right]\right|\right] \leq 3\sqrt{\xi}.$$

Now, let $z' \in \{\pm 1\}^n$ be as in the statement of the lemma. Then,

$$1-3\xi-4\sqrt{\xi} \leq \left|\mathbb{E}_{i\in[n]} z_i^* \cdot z_i'\right|.$$

Hence, we conclude that $\left|\mathbb{E}_{i\in[n]} z_i^* \cdot z_i'\right| \geq 1-7\sqrt{\xi}$. ∎

Remark 2.10.6. *The parity sampling requirement might be slightly stronger with this derandomized version but it does not change the asymptotic nature of our results. More precisely, we are only asking for $(1-\xi, 2\varepsilon)$-parity sampler for a different constant value $\xi > 0$.*

CHAPTER 3

UNIQUE DECODING OF EXPLICIT ε-BALANCED CODES NEAR THE GILBERT–VARSHAMOV BOUND

This chapter is joint work with Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani that originally appeared in [JQST20].

3.1 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 β vanishes as ε vanishes.[1] Actually, Ta-Shma obtained ε-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

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

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

Theorem 3.1.1 (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 ε-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 3.1.1).

Theorem 3.1.2 (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 ε-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 $N^{O_{\varepsilon,\beta}(1)}$.

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.

We observe that the exponent in the running time $N^{O_{\varepsilon,\beta}(1)}$ appearing in Theorem 3.1.1 and Theorem 3.1.2 depends on ε. This dependence is no worse than $O(\log\log(1/\varepsilon))$, and if $\beta > 0$ is taken to be an arbitrarily constant (independent of ε), 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 3.1.2 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].

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 ε_0-balanced for a constant ε_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 ε-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$^+$92] that the resulting code admits a unique decoding algorithm if the incidence graph corresponding to the collection W is a good sampler. Very recently, it was proved by Dinur et al. [DHK$^+$19] that such a direct product construction admits list decoding if the incidence graph is a "double sampler". The results of [DHK$^+$19] also apply to direct sum, but the use of double samplers pushes the rate away from near optimality.

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 3.3.

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 (e.g., Table 3.1). 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). We summarize the results discussed so far in Table 3.1.

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

Binary Code Results near the Gilbert–Varshamov bound						
Who?	**Construction**	**GV**	**Explicit**	**Concatenated**	**Decoding**	**Local**
[Gil52, Var57]	existential	yes	no	no	no	n/a
[Tho83]	Reed–Solomon + random binary	yes	no	yes	no	n/a
[GI04]	Thommesen [Tho83]	yes	no	yes	unique decoding	n/a
[GKO$^+$17]	Thommesen-like	yes	no	yes	unique decoding	LTC/LCC
[HRW17]	Thommesen-like	yes	no	yes	near linear time unique decoding	n/a
[TS17]	Expander-based	$\Omega(\varepsilon^{2+\beta})$	yes	no	no	n/a
this paper	Ta-Shma [TS17]	$\Omega(\varepsilon^{2+\beta})$	yes	no	gentle list decoding	n/a

Table 3.1: Binary codes near the GV bound.

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.

3.2 Preliminaries and Notation

3.2.1 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 $\Delta(\mathcal{C}) \coloneqq \min_{z \neq z'} \Delta(z, z')$ 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 3.2.1 (Bias)**.** *The* bias *of a word $z \in \mathbb{F}_2^n$ is defined as* $\mathrm{bias}(z) \coloneqq \left|\mathbb{E}_{i \in [n]}(-1)^{z_i}\right|$. *The bias of a code $\mathcal{C}$ is the maximum bias of any non-zero codeword in $\mathcal{C}$.*

Definition 3.2.2 (ε-balanced Code)**.** *A binary code $\mathcal{C}$ is* ε-balanced *if* $\mathrm{bias}(z + z') \leq \varepsilon$ *for every pair of distinct $z, z' \in \mathcal{C}$.*

Remark 3.2.3. *For linear binary code $\mathcal{C}$, the condition* $\mathrm{bias}(\mathcal{C}) \leq \varepsilon$ *is equivalent to $\mathcal{C}$ being an ε-balanced code.*

3.2.2 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 3.2.4 (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 3.2.5. *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 3.2.6 (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* $\mathrm{bias}(\mathrm{dsum}_{W(k)}(z)) \leq \varepsilon$*.*

3.2.3 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.

3.3 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 ε-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 ε-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 A_G being λ. 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

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

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 ε-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. 3.1). 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 $\Delta(z, \overline{\tilde{z}}) < (1-\varepsilon_0)/4$) and we can use a unique decoding algorithm for $\mathcal{C}$ to find z given $\tilde{z}$.

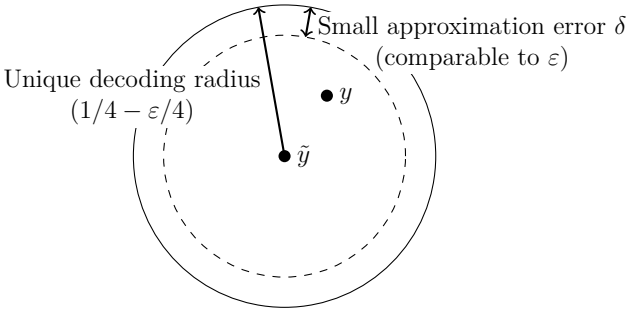

Figure 3.1: Unique decoding ball along with error from approximation.

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 3.3.1 (Splittability (informal))**.** *A collection $W(t) \subseteq [n]^t$ is said to be τ-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 τ (where ww' denotes the concatenated tuple).*

2. *The collections $W[1, t']$ and $W[t'+1, t]$ are τ-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 S as $\mathsf{A}_G \otimes \mathsf{J}_d/d$ where A_G is normalized adjacency matrix of G and 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 τ-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* up to a radius of $1/2 - \sqrt{\varepsilon} - \delta$ under a similar condition on τ, which will be very useful for our application.[3]

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.

While the above can yield decoding algorithms for suitably expanding G, the requirement on τ (and hence on λ) 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. 3.2). However, this alone does not improve the rate as we still need the splittability (and hence λ) to be $2^{-\Omega(t)}$ with $t = O(\log(1/\varepsilon))$.

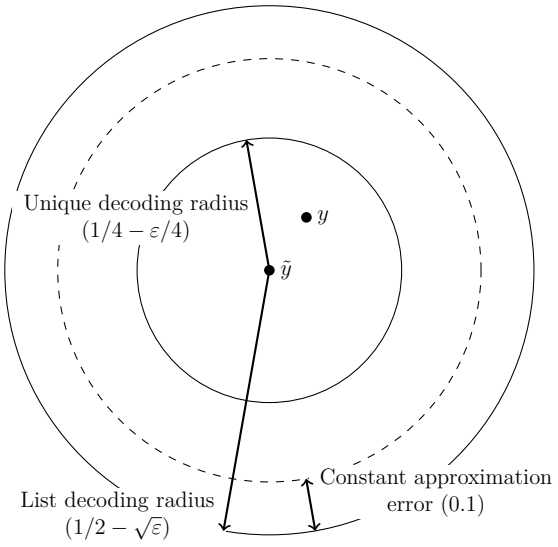

Figure 3.2: Unique decoding and list decoding balls along with error from approximation. Note that the list decoding ball contains the unique decoding ball even after allowing for a relatively large amount of error.

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 ε), 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 Section 3.5) as *code cascades* (see Fig. 3.3).

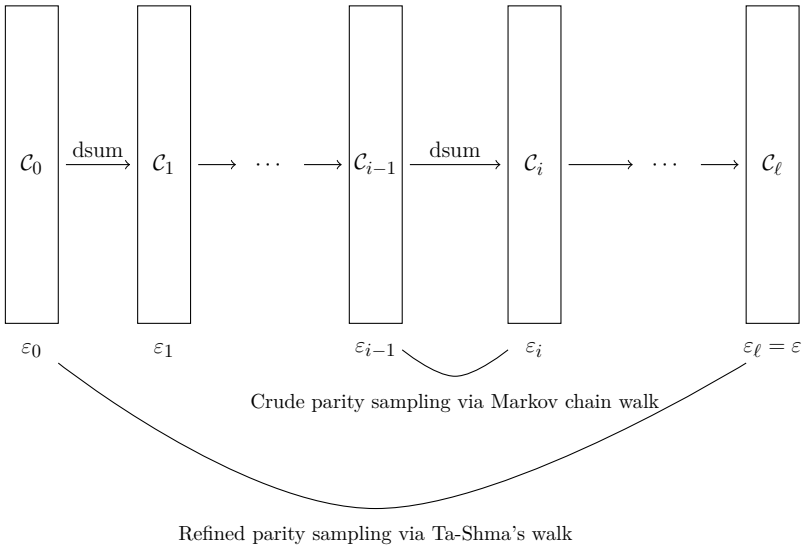

Figure 3.3: Code cascading.

Instead of applying the decoding framework 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 (η_0, η)-parity sampler, the list decoding algorithm at radius

$1/2 - \sqrt{\eta}$ suffices for unique decoding even if η is a (sufficiently small) constant independent of ε. This implies that we can take k to be a (suitably large) constant. This also allows the splittability (and hence λ) to be $2^{-O(k)} = \Omega(1)$, yielding polynomial rates. We present the reduction using cascades in Section 3.6 and the parameter choices in Section 3.8. The specific versions of the list decoding results from [AJQ+20] needed here are instantiated in Section 3.9.

While the above allows for polynomial rate, the *running time* of the algorithm is still exponential in the number of levels ℓ (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 Section 3.6.

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 Section 3.4.

We also verify that the tuples given by Ta-Shma's construction satisfy the conditions for applying the list decoding framework, in Section 3.7. 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 Section 3.7 that this is indeed implied by splittability, and also prove splittability for (the modified version of) Ta-Shma's construction.

3.4 Ta-Shma's Construction: A Summary and Some Tweaks

In this section, we first discuss the s-wide replacement product that is central to Ta-Shma's construction of optimal ε-balanced codes, and then we describe the construction itself (we refer the reader to [TS17] for formal details beyond those we actually need here).

As mentioned before, we will also need to modify Ta-Shma's construction [TS17] a little to get *splittability* which is a notion of expansion of a collection $W(k) \subseteq [n]^k$ (and it is formally defined in Definition 3.7.9). The reason for this simple modification is that this *splittability* property is required by the list decoding framework. Note that we are not improving the Ta-Shma code parameters; in fact, we need to argue why with this modification we can still achieve Ta-Shma's parameters. Fortunately, this modification is simple enough that we will be able to essentially reuse Ta-Shma's original analysis. In Section 3.4.3, we will also have the opportunity to discuss, at an informal level, the intuition behind some parameter trade-offs in Ta-Shma codes which should provide enough motivation when we instantiate these codes in Section 3.8.

3.4.1 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 Definition 3.4.2 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 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 A_H is the normalized adjacency matrix of H. The inter-cloud substeps are all deterministic, with the first moving according to G_0, the second according to G_1, and so on, returning to 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 nd_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).

To formally define the s-wide replacement product, we must consider the labeling of neighbors in G more carefully.

Definition 3.4.1 (Rotation Map)**.** *Suppose G is a d_1-regular graph on $[n]$. For each $v \in [n]$ and $j \in [d_1]$, let $v_G[j]$ be the j-th neighbor of v in G. Based on the indexing of the neighbors of each vertex, we define the rotation map* $\mathrm{rot}_G \colon [n] \times [d_1] \to [n] \times [d_1]$ *such that for every* $(v, j) \in [n] \times [d_1]$,

$$\mathrm{rot}_G((v, j)) = (v', j') \Leftrightarrow v_G[j] = v' \text{ and } v'_G[j'] = v.$$

Furthermore, if there exists a bijection $\varphi \colon [d_1] \to [d_1]$ *such that for every* $(v, j) \in [n] \times [d_1]$,

$$\mathrm{rot}_G((v, j)) = (v_G[j], \varphi(j)),$$

then we call rot_G locally invertible.

If G has a locally invertible rotation map, the cloud label after applying the rotation map only depends on the current cloud label, not the vertex of G. In the s-wide replacement product, this corresponds to the H component of the rotation map only depending on a vertex's H component, not its G component. We define the s-wide replacement product as described before, with the inter-cloud operator G_i using the i-th coordinate of the H component, which is a value in $[d_1]$, to determine the inter-cloud step.

Definition 3.4.2 (s-wide replacement product)**.** *Suppose we are given the following:*

- *A d_1-regular graph $G = ([n], E)$ together with a locally invertible rotation map* $\mathrm{rot}_G \colon [n] \times [d_1] \to [n] \times [d_1]$.
- *A d_2-regular graph $H = ([d_1]^s, E')$.*

And we define:

- *For $i \in \{0, 1, \ldots, s-1\}$, we define* $\mathrm{Rot}_i \colon [n] \times [d_1]^s \to [n] \times [d_1]^s$ *as, for every $v \in [n]$*

and $(a_0, \ldots, a_{s-1}) \in [d_1]^s$,

$$\mathrm{Rot}_i((v, (a_0, \ldots, a_{s-1}))) \coloneqq (v', (a_0, \ldots, a_{i-1}, a_i', a_{i+1}, \ldots, a_{s-1})),$$

where $(v', a_i') = \mathrm{rot}_G(v, a_i)$.

- *Denote by* G_i *the operator realizing* Rot_i *and let* A_H *be the normalized random walk operator of* H. *Note that* G_i *is a permutation operator corresponding to a product of transpositions.*

Then $t-1$ *steps of the* s*-wide replacement product are given by the operator*

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

Ta-Shma instantiates the s-wide replacement product with an outer graph G that is a Cayley graph, for which locally invertible rotation maps exist generically.

Remark 3.4.3. *Let* R *be a group and* $A \subseteq R$ *where the set* A *is closed under inversion. For every Cayley graph* $\mathrm{Cay}(R, A)$, *the map* $\varphi\colon A \to A$ *defined as* $\varphi(g) = g^{-1}$ *gives rise to the locally invertible rotation map*

$$\mathrm{rot}_{\mathrm{Cay}(R,A)}((r, a)) = (r \cdot a, a^{-1}),$$

for every $r \in R$, $a \in A$.

3.4.2 The Construction

Ta-Shma's code construction works by starting with a constant bias code $\mathcal{C}_0$ in $\mathbb{F}_2^n$ and boosting to arbitrarily small bias using direct sum liftings. Recall that the direct sum lifting is based on a collection $W(t) \subseteq [n]^t$, which Ta-Shma obtains using $t-1$ steps of random walk

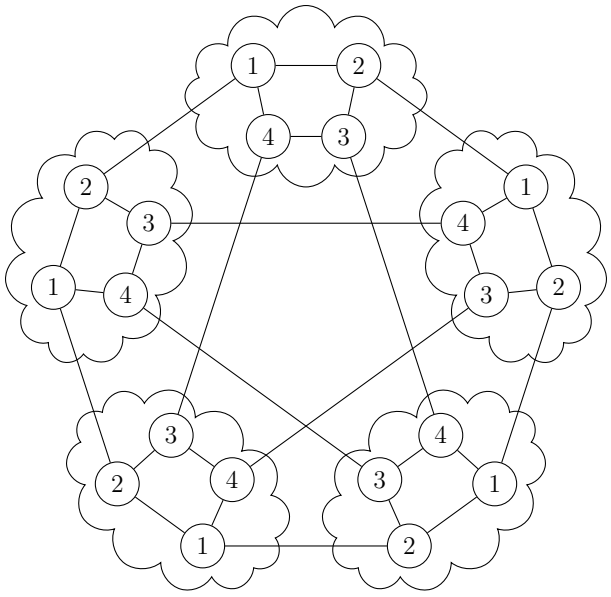

Figure 3.4: An example of the 1-wide replacement product with outer graph $G = K_5$ and inner graph $H = C_4$. Vertices are labeled by their H components. Note that the rotation map is locally invertible, with $\varphi(1) = 2$, $\varphi(2) = 1$, $\varphi(3) = 4$, and $\varphi(4) = 3$.

on the s-wide replacement product of two regular expander graphs G and H. The graph G is on n vertices (same as blocklength of the base code) and other parameters like degrees d_1 and d_2 of G and H respectively are chosen based on target code parameters.

To elaborate, every $t-1$ length walk on the replacement product gives a sequence of t outer vertices or G-vertices, which can be seen as an element of $[n]^t$. This gives the collection $W(t)$ with $|W(t)| = n \cdot d_1^s \cdot d_2^{t-1}$ which means the rate of lifted code is smaller than the rate of $\mathcal{C}_0$ by a factor of $d_1^s d_2^{t-1}$. However, the collection $W(t)$ is a parity sampler and this means that the bias decreases (or the distance increases). The relationship between this decrease in bias and decrease in rate with some careful parameter choices allows Ta-Shma to obtain nearly optimal ε-balanced codes.

3.4.3 Tweaking the Construction

Recall the first s steps in Ta-Shma's construction are given by the operator

$$\mathsf{G}_{s-1}(\mathsf{I}\otimes\mathsf{A}_H)\mathsf{G}_{s-2}\cdots G_1(\mathsf{I}\otimes\mathsf{A}_H)\mathsf{G}_0(\mathsf{I}\otimes\mathsf{A}_H).$$

Naively decomposing the above operator into the product of operators $\prod_{i=0}^{s-1}\mathsf{G}_i(\mathsf{I}\otimes\mathsf{A}_H)$ is not good enough to obtain the *splittability* property which would hold provided $\sigma_2(\mathsf{G}_i(\mathsf{I}\otimes\mathsf{A}_H))$ was small for every i in $\{0,\ldots,s-1\}$. However, each $\mathsf{G}_i(\mathsf{I}\otimes\mathsf{A}_H)$ has $|V(G)|$ singular values equal to 1 since G_i is an orthogonal operator and $(\mathsf{I}\otimes\mathsf{A}_H)$ has $|V(G)|$ singular values equal to 1. To avoid this issue we will tweak the construction to be the following product

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

The operator $(\mathsf{I}\otimes\mathsf{A}_H)\mathsf{G}_i(\mathsf{I}\otimes\mathsf{A}_H)$ is exactly the walk operator of the zig-zag product $G\textcircled{z}H$ of G and H with a rotation map given by the (rotation map) operator G_i. This tweaked construction is slightly simpler in the sense that $G\textcircled{z}H$ is an undirected graph. We know by the zig-zag analysis that $(\mathsf{I}\otimes\mathsf{A}_H)\mathsf{G}_i(\mathsf{I}\otimes\mathsf{A}_H)$ is expanding as long G and H are themselves expanders. More precisely, we have a bound that follows from [RVW00].

Fact 3.4.4. *Let G be an outer graph and H be an inner graph used in the s-wide replacement product. For any integer $0\leq i\leq s-1$,*

$$\sigma_2((\mathsf{I}\otimes\mathsf{A}_H)\mathsf{G}_i(\mathsf{I}\otimes\mathsf{A}_H))\leq\sigma_2(G)+2\cdot\sigma_2(H)+\sigma_2(H)^2.$$

This bound will imply *splittability* as shown in Section 3.7.2. We will need to argue that this modification still preserves the correctness of the parity sampling and that it can be achieved with similar parameter trade-offs.

The formal definition of a length-t walk on this slightly modified construction is given

below.

Definition 3.4.5. *Let $t \in \mathbb{N}$, G be a d_1-regular graph and H be a d_2-regular graph on d_1^s vertices. Given a starting vertex $(v, h) \in V(G) \times V(H)$, a $(t-1)$-step walk on the tweaked s-wide replacement product of G and H is a tuple $((v_0, h_0), \ldots, (v_{t-1}, h_{t-1})) \in (V(G) \times V(H))^t$ such that*

- *$(v_0, h_0) = (v, h)$, and*
- *for every $0 \leq i < t-1$, we have (v_i, h_i) adjacent to (v_{i+1}, h_{i+1}) in $(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{i \bmod s}(\mathsf{I} \otimes \mathsf{A}_H)$.*

Note that each $(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{i \bmod s}(\mathsf{I} \otimes \mathsf{A}_H)$ is a walk operator of a d_2^2-regular graph. Therefore, the starting vertex (v, h) together with a degree sequence $(m_1, \ldots, m_t) \in [d_2^2]^{t-1}$ uniquely defines a $(t-1)$-step walk.

Parity Sampling

We argue informally why parity sampling still holds with similar parameter trade-offs. Later in Section 3.4.3, we formalize a key result underlying parity sampling and, in Section 3.8, we compute the new trade-off between bias and rate in some regimes. In Section 3.4.1, the definition of the original s-wide replacement product as a purely graph theoretic operation was given. Now, we explain how Ta-Shma used this construction for parity sampling obtaining codes near the GV bound.

For a word $z \in \mathbb{F}_2^{V(G)}$ in the base code, let P_z be the diagonal matrix, whose rows and columns are indexed by $V(G) \times V(H)$, with $(\mathsf{P}_z)_{(v,h),(v,h)} = (-1)^{z_v}$. Proving parity sampling requires analyzing the operator norm of the following product

$$\mathsf{P}_z \prod_{i=0}^{s-1} (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i\mathsf{P}_z(\mathsf{I} \otimes \mathsf{A}_H), \tag{3.1}$$

when $\text{bias}(z) \leq \varepsilon_0$. Let $\mathbf{1} \in \mathbb{R}^{V(G)\times V(H)}$ be the all-ones vector and W be the collection of all $(t-1)$-step walks on the tweaked s-wide replacement product. Ta-Shma showed (and it is not difficult to verify) that

$$\text{bias}\left(\text{dsum}_W(z)\right) = \left|\left\langle \mathbf{1}, \mathsf{P}_z \prod_{i=0}^{t-2} (\mathsf{I}\otimes \mathsf{A}_H)\mathsf{G}_{i \bmod s}\mathsf{P}_z(\mathsf{I}\otimes \mathsf{A}_H)\mathbf{1}\right\rangle\right|.$$

From the previous equation, one readily deduces that

$$\text{bias}\left(\text{dsum}_W(z)\right) \leq \sigma_1 \left(\mathsf{P}_z \prod_{i=0}^{s-1} (\mathsf{I}\otimes \mathsf{A}_H)\mathsf{G}_i\mathsf{P}_z(\mathsf{I}\otimes \mathsf{A}_H)\right)^{\lfloor (t-1)/s\rfloor}.$$

Set $\mathsf{B} \coloneqq \mathsf{P}_z \prod_{i=0}^{s-1} (\mathsf{I}\otimes \mathsf{A}_H)\mathsf{G}_i\mathsf{P}_z(\mathsf{I}\otimes \mathsf{A}_H)$. To analyze the operator norm of B, we will first need some notation. Note that B is an operator acting on the space $\mathcal{V} = \mathbb{R}^{V(G)} \otimes \mathbb{R}^{V(H)}$. Two of its subspaces play an important role in the analysis, namely,

$$\mathcal{W}^{\|} = \text{span}\{a \otimes b \in \mathbb{R}^{V(G)} \otimes \mathbb{R}^{V(H)} \mid b = \mathbf{1}\} \text{ and } \mathcal{W}^{\perp} = (\mathcal{W}^{\|})^{\perp}.$$

Note that the complement subspace is with respect to the standard inner product. Observe that $\mathcal{V} = \mathcal{W}^{\|} \oplus \mathcal{W}^{\perp}$. Given arbitrary unit vectors $v, w \in \mathcal{V}$, Ta-Shma considers the inner product

$$\left\langle v, \prod_{i=0}^{s-1} (\mathsf{I}\otimes \mathsf{A}_H)\mathsf{G}_i\mathsf{P}_z(\mathsf{I}\otimes \mathsf{A}_H)w\right\rangle. \tag{3.2}$$

Each time an operator $(\mathsf{I}\otimes \mathsf{A}_H)$ appears in the above expression, the next step of the walk can take one out of d_2 possibilities and thus the rate suffers a multiplicative decrease of $1/d_2$. We think that we are "paying" d_2 for this step of the walk. The whole problem lies in the trade-off between rate and distance, so the crucial question now is how much the norm decreases as we pay d_2. For a moment, suppose that the norm always decreases by a factor of $\lambda_2 \coloneqq \sigma_2(H)$ per occurrence of $(\mathsf{I}\otimes \mathsf{A}_H)$. If in this hypothetical case we could further assume $\lambda_2 = 1/\sqrt{d_2}$, then if B was a product containing $\lceil \log_{\lambda_2}(\varepsilon)\rceil$ factors of $(\mathsf{I}\otimes \mathsf{A}_H)$, the

final bias would be at most ε and the rate would have suffered a multiplicative decrease of (essentially) ε^2 and we would be done.

Of course, this was an oversimplification. The general strategy is roughly the above, but a beautiful non-trivial step is needed. Going back to the bilinear form Eq. (3.2), if $w \in \mathcal{W}^{\perp}$ (or $v \in \mathcal{W}^{\perp}$), we pay d_2 and we do obtain a norm decrease of λ_2. More generally, note that can decompose $w = w^{\|} + w^{\perp}$ with $w^{\|} \in \mathcal{W}^{\|}$ and $w^{\perp} \in \mathcal{W}^{\perp}$ (decompose $v = v^{\|} + v^{\perp}$ similarly) and we can carry this process iteratively collecting factors of λ_2. However, we are stuck with several terms of the form for $0 \leq k_1 \leq k_2 < s$,

$$\left\langle v_{k_1}^{\|}, \prod_{i=k_1}^{k_2} (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i\mathsf{P}_z(\mathsf{I} \otimes \mathsf{A}_H) w_{k_2}^{\|} \right\rangle,$$

with $v_{k_1}^{\|}, w_{k_2}^{\|} \in \mathcal{W}^{\|}$, and for which the preceding naive norm decrease argument fails. This is the point in the analysis where the structure of the s-wide replacement product is used. Since $v_{k_1}^{\|}, w_{k_2}^{\|} \in \mathcal{W}^{\|}$, these vectors are uniform on each "cloud", i.e., copy of H. Recall that a vertex in H is an s-tuple $(m_1, \ldots, m_s) \in [d_1]^s$. Ta-Shma leverages the fact of having a uniform such tuple to implement $k_2 - k_1 + 1$ (up to s) steps of random walk on G. More precisely, Ta-Shma obtains the following beautiful result:

Theorem 3.4.6 (Adapted from Ta-Shma [TS17])**.** *Let G be a locally invertible graph of degree d_1, H be a Cayley graph on $\mathbb{F}_2^{s \log d_1}$, and $0 \leq k_1 \leq k_2 < s$ be integers. If $v^{\|} = v \otimes 1$ and $w^{\|} = w \otimes 1$, then*

$$\left\langle v^{\|}, \prod_{i=k_1}^{k_2} \mathsf{G}_i(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{P}_z w^{\|} \right\rangle = \left\langle v, (\mathsf{A}_G\mathsf{M}_z)^{k_2-k_1+1}\, w \right\rangle$$

where $\mathsf{M}_z \in \mathbb{R}^{V(G)\times V(G)}$ is the diagonal matrix defined as $(\mathsf{M}_z)_{v,v} \coloneqq (-1)^{z_v}$ for $v \in V(G)$.

Remark 3.4.7. *Note that the walk operator in this theorem corresponds to the original construction. Theorem 3.4.6 was used by Ta-Shma to obtain Fact 3.4.9 whose Corollary 3.4.10*

corresponds to the modified construction.

Ta-Shma proved Theorem 3.4.6 under the more general condition that H is 0-pseudorandom. Roughly speaking, this property means that if we start with a distribution that is uniform over the clouds, and walk according to fixed H-steps $j_0, j_1, \cdots, j_{s-1}$, then the distribution of G-vertices obtained will be identical to the distribution obtained if we were doing the usual random walk on G. We will always choose H to be a Cayley graph on $\mathbb{F}_2^{s \log d_1}$, which will imply that H is also 0-pseudorandom. The proof of Theorem 3.4.6 crucially uses the product structure of $\mathbb{F}_2^{s \log d_1}$: every vertex of H can be represented by s registers of $\log d_1$ bits each, and both inter-cloud and intra-cloud steps can be seen as applying register-wise bijections using some canonical mapping between $[d_1]$ and $\mathbb{F}_2^{\log d_1}$.

Ta-Shma's original parity sampling proof required $\varepsilon_0 + 2\theta + 2\sigma_2(G) \leq \sigma_2(H)^2$, where ε_0 is the initial bias and θ is an error parameter arising from a number theoretic construction of Ramanujan graphs for the outer graph G. This is because $\varepsilon_0 + 2\theta + 2\sigma_2(G)$ is the reduction of bias in every two steps while taking a walk on G (see Theorem 3.5.2). Having $\varepsilon_0 + 2\theta + 2\sigma_2(G) \leq \sigma_2(H)^2$ ensured that after establishing Theorem 3.4.6, we were collecting enough reduction for d_2^2 price we paid for two steps. In the modified construction, we now have d_2^2 possibilities for each step in $(\mathsf{I} \otimes \mathsf{A}_H^2)$ (so d_2^4 price for two steps), and so if instead we have $\varepsilon_0 + 2\theta + 2\sigma_2(G) \leq \sigma_2(H)^4$ in the modified construction, we claim that the correctness of the parity sampling analysis is preserved as well as (essentially) the trade-off between walk length and norm decay. Fortunately, Ta-Shma's parameters decouple and we can choose parameters to satisfy the above requirement.

Remark 3.4.8. *This modification on the s-replacement product of G and H essentially amounts to taking a different inner graph H which can be factored as $H = \sqrt{H}\sqrt{H}$ (and is still 0-pseudorandom).*[4]

4. The only differences occur at the first and last factors in the product of operators.

Spectral Analysis of the Modified Construction

We formally show that we don't loose much by going from Ta-Shma's original s-wide product construction to its tweaked version. The key technical result obtained by Ta-Shma is the following, which is used to analyze the bias reduction as a function of the total number walk steps $t-1$.

Fact 3.4.9 (Theorem 24 abridged [TS17])**.** *If H is a Cayley graph on $\mathbb{F}_2^{s \log d_1}$ and $\varepsilon_0 + 2 \cdot \theta + 2 \cdot \sigma_2(G) \leq \sigma_2(H)^2$, then*

$$\left\| \prod_{i=0}^{s-1} \mathsf{P}_z \mathsf{G}_i (\mathsf{I} \otimes \mathsf{A}_H) \right\|_{\text{op}} \leq \sigma_2(H)^s + s \cdot \sigma_2(H)^{s-1} + s^2 \cdot \sigma_2(H)^{s-3},$$

where $\mathsf{P}_z \in \mathbb{R}^{(V(G) \times V(H)) \times (V(G) \times V(H))}$ is the sign operator *of a ε_0 biased word $z \in \mathbb{F}_2^{V(G)}$ defined as a diagonal matrix with $(P_z)_{(v,h),(v,h)} = (-1)^{z_v}$ for every $(v,h) \in V(G) \times V(H)$.*

We reduce the analysis of Ta-Shma's tweaked construction to Fact 3.4.9. In doing so, we only lose one extra step as shown below.

Corollary 3.4.10. *If H^2 is a Cayley graph on $\mathbb{F}_2^{s \log d_1}$ and $\varepsilon_0 + 2 \cdot \theta + 2 \cdot \sigma_2(G) \leq \sigma_2(H)^4$, then*

$$\left\| \prod_{i=0}^{s-1} (\mathsf{I} \otimes \mathsf{A}_H) \mathsf{P}_z \mathsf{G}_i (\mathsf{I} \otimes \mathsf{A}_H) \right\|_{\text{op}} \leq \sigma_2(H^2)^{s-1} + (s-1) \cdot \sigma_2(H^2)^{s-2} + (s-1)^2 \cdot \sigma_2(H^2)^{s-4},$$

where P_z is the sign operator *of an ε_0-biased word $z \in \mathbb{F}_2^{V(G)}$ as in Fact 3.4.9.*

Proof. We have

$$\begin{aligned}\left\|\prod_{i=0}^{s-1}(\mathsf{I}\otimes\mathsf{A}_H)\mathsf{P}_z\mathsf{G}_i(\mathsf{I}\otimes\mathsf{A}_H)\right\|_{\mathrm{op}} &\le \|(\mathsf{I}\otimes\mathsf{A}_H)\|_{\mathrm{op}}\left\|\prod_{i=1}^{s-1}\mathsf{P}_z\mathsf{G}_i(\mathsf{I}\otimes\mathsf{A}_H^2)\right\|_{\mathrm{op}}\|\mathsf{P}_z\mathsf{G}_0(\mathsf{I}\otimes\mathsf{A}_H)\|_{\mathrm{op}}\\ &\le \left\|\prod_{i=1}^{s-1}\mathsf{P}_z\mathsf{G}_i(\mathsf{I}\otimes\mathsf{A}_H^2)\right\|_{\mathrm{op}}\\ &\le \sigma_2(H^2)^{s-1}+(s-1)\cdot\sigma_2(H^2)^{s-2}+(s-1)^2\cdot\sigma_2(H^2)^{s-4},\end{aligned}$$

where the last inequality follows from Fact 3.4.9. ∎

Remark 3.4.11. *We know that in the modified construction H^2 is a Cayley graph since H is a Cayley graph.*

From this point onward, we will be working exclusively with the modified construction instead of using it in its original form. Any references to Ta-Shma's construction or the s-wide replacement product will actually refer to the modified versions described in this section.

3.5 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 3.5.1. *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 ε. 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 ℓ depends on ε). 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 ℓ-level recursive procedure, the gain from replacing t by t_i will outweigh this loss, as we discuss below.

3.5.1 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 ε_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 3.5.2 (Adapted from Ta-Shma [TS17])**.** *Let $\mathcal{C}$ be an ε_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 ε-balanced. Observe that the required expansion of the graph G only depends on the constant initial bias ε_0, not on the desired final bias ε. It will be important for being able to decode with better parameters that both $\sigma_2(G)$ and m are constant with respect to ε; only ℓ depends on the final bias (with more care we can make $\sigma_2(G)$ depend on ε, 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 Theorem 3.6.1 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.

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

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

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 ε_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 over the vertex set $W_i(s)$.[5] 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 (see Definition 3.7.2) to Section 3.7.2 where it is actually used.

- $\mathcal{C}_\ell$ denotes the final code in the sequence, which will later be chosen so that its bias is at most ε.

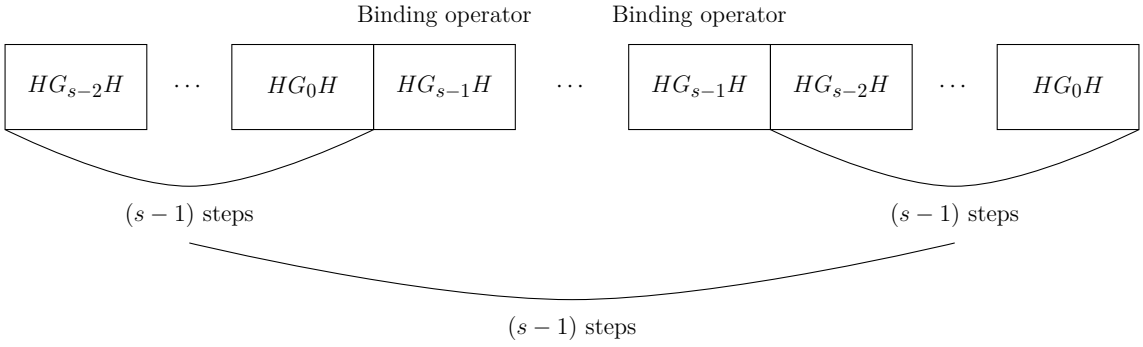

Figure 3.5: Two levels of code cascading for Ta-Shma's construction involving codes $\mathcal{C}_0$, $\mathcal{C}_1$, and $\mathcal{C}_2$ (to make the notation compact we use H to denote $\mathsf{I} \otimes \mathsf{A}_H$).

3.6 Unique Decoding of Ta-Shma Codes

We show how code cascading together with list decoding for each level of the cascade allow us to obtain an efficient unique decoding algorithm for Ta-Shma's construction. We obtain a sequence of results of increasing strength culminating in Theorem 3.1.1 (which we recall below for convenience). The approach is as follows: we use several different instantiations

5. 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.

of Ta-Shma's construction, each yielding a value of s (for the s-wide replacement product) and expansion parameters for the family of outer and inner graphs, and show how the list decoding framework can be invoked in the associated cascade for each one.

Theorem 3.1.1 (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 ε-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).

In this section, we will fit these objects and tools together assuming the parameters are chosen to achieve the required rates and the conditions for applying the list decoding results are satisfied. The concrete instantiations of Ta-Shma codes are done in Section 3.8. Establishing that the list decoding framework can be applied to this construction is done in Section 3.7 after which the framework is finally instantiated in Section 3.9.

Ta-Shma uses the direct sum lifting on an s-wide replacement product graph to construct a family of ε-balanced codes $\mathcal{C}_{N,\varepsilon,\beta}$ with rate $\Omega(\varepsilon^{2+\beta})$ and finds parameters for such codes to have the required bias and rate. We will discuss unique decoding results for several versions of these codes. Throughout this section, we will use collections $W(k)$ which will always be either the set of walks with $k = s$ vertices on an s-wide replacement product graph (corresponding to the first level of the code cascade), which we denote $W[0, s-1]$, or a set of walks where the vertices are walks on a lower level of the code cascade.

3.6.1 Unique Decoding via Code Cascading

To perform unique decoding we will use the machinery of list decoding from Theorem 3.6.1 (proven later in Section 3.9), which relies on the list decoding framework of [AJQ+20]. Proving that this framework can be applied to Ta-Shma's construction is one of our technical contributions.

Theorem 3.6.1 (List decoding direct sum lifting)**.** *Let $\eta_0 \in (0, 1/4)$ be a constant, $\eta \in (0, \eta_0)$, and*

$$k \geq k_0(\eta) \coloneqq \Theta(\log(1/\eta)).$$

Suppose $\mathcal{C} \subseteq \mathbb{F}_2^n$ is an η_0-balanced linear code and $\mathcal{C}' = \mathrm{dsum}_{W(k)}(\mathcal{C})$ is the direct sum lifting of $\mathcal{C}$ on a τ-splittable collection of walks $W(k)$. There exists an absolute constant $K > 0$ such that if

$$\tau \leq \tau_0(\eta, k) \coloneqq \frac{\eta^8}{K \cdot k \cdot 2^{4k}},$$

then the code $\mathcal{C}'$ is η-balanced and can be efficiently list decoded in the following sense:

If $\tilde{y}$ is $(1/2 - \sqrt{\eta})$-close to $\mathcal{C}'$, then we can compute the list

$$\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}') \coloneqq \left\{(z, \mathrm{dsum}_{W(k)}(z)) \mid z \in \mathcal{C}, \Delta\Big(\mathrm{dsum}_{W(k)}(z), \tilde{y}\Big) \leq \frac{1}{2} - \sqrt{\eta}\right\}$$

in time

$$n^{O(1/\tau_0(\eta,k)^4)} \cdot f(n),$$

where $f(n)$ is the running time of a unique decoding algorithm for $\mathcal{C}$. Otherwise, we return $\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}') = \emptyset$ with the same running time of the preceding case.

Note that the requirement on k in the above theorem is necessary for the lifted code $\mathcal{C}'$ to be η-balanced. Splittability will imply that the walk collection $W(k)$ is expanding, which gives us parity sampling for large k. Specifically, k must be large enough for $W(k)$ to be a $(1/2 + \eta_0/2, \eta)$-parity sampler.

Applying the list decoding tool above, we can perform unique decoding in the regime of η_0, η, and k being constant. With these choices the expansion required for splittability and the parity sampling strength are only required to be constants.

Lemma 3.6.2 (Decoding Step)**.** *Let $\eta_0 \in (0, 1/4)$ and $\eta < \min\{\eta_0, 1/16\}$. If $W(k)$ is a walk collection on vertex set $[n]$ with $k \geq k_0(\eta)$ and splittability $\tau \leq \tau_0(\eta, k)$, where k_0 and τ_0 are as in Theorem 3.6.1, we have the following unique decoding property:*

If $\mathcal{C} \subseteq \mathbb{F}_2^n$ is an η_0-balanced linear code that can be uniquely decoded in time $f(n)$, then $\mathcal{C}' = \mathrm{dsum}_{W(k)}(\mathcal{C})$ is an η-balanced code that can be uniquely decoded in time $n^{O(1/\tau_0(\eta,k)^4)} \cdot f(n)$.

Proof. Using Theorem 3.6.1, we can list decode $\mathcal{C}'$ up to a radius of $1/2 - \sqrt{\eta}$ for any η if we have the appropriate parameters k and τ. Let $\tilde{y} \in \mathbb{F}_2^{W(k)}$ be a received word within the unique decoding radius of $\mathcal{C}'$. To perform unique decoding, we simply run the list decoding algorithm on $\tilde{y}$ and return the codeword on the resulting list which is closest to $\tilde{y}$; this will yield the correct result as long as the list decoding radius is larger than the unique decoding radius. It suffices to have $1/2 - \sqrt{\eta} > 1/4 \geq \Delta(\mathcal{C}')/2$. We choose parameters as follows:

1. Take $\eta < 1/16$ to ensure $1/2 - \sqrt{\eta} > 1/4$.
2. Let $k_0 = \Theta(\log(1/\eta))$ be the smallest integer satisfying the assumption in Theorem 3.6.1 with the chosen η. Take $k \geq k_0$.
3. Take $\tau \leq \tau_0(\eta, k) = \eta^8/(K \cdot k \cdot 2^{4k})$.

Note that k and τ satisfy the conditions of Theorem 3.6.1, so we can use this theorem to list decode a received word $\tilde{y}$ in time $n^{O(1/\tau_0(\eta,k)^4)} \cdot f(n)$. To unique decode, we return the closest y on the list to $\tilde{y}$ (or failure if the list is empty). ∎

Iteratively using the decoding step given by Lemma 3.6.2 above, we obtain unique decodability of all codes in a cascade (under suitable assumptions).

Lemma 3.6.3 (Code Cascade Decoding)**.** *Let $\eta_0 \in (0, 1/4)$ and $\eta < \min\{\eta_0, 1/16\}$. Suppose $\mathcal{C}_0 \subseteq \mathbb{F}_2^{n_0}, \mathcal{C}_1 \subseteq \mathbb{F}_2^{n_1}, \ldots, \mathcal{C}_\ell \subseteq \mathbb{F}_2^{n_\ell}$ is a code cascade where $\mathcal{C}_0$ is an η_0-balanced linear code that can be uniquely decoded in time $g(n_0)$.*

If for every $i \in [\ell]$ we have that $\mathcal{C}_i$ is obtained from $\mathcal{C}_{i-1}$ by a τ_i-splittable walk collection $W_i(k_i)$ on vertex set $[n_{i-1}]$ with $k_i \geq k_0(\eta)$ and $\tau_i \leq \tau_0(\eta, k_i)$, where k_0 and τ_0 are as in Theorem 3.6.1, then $\mathcal{C}_\ell$ is uniquely decodable in time

$$g(n_0) \cdot \prod_{i=1}^{\ell} n_{i-1}^{O(1/\tau_0(\eta,k_i)^4)}.$$

Proof. Induct on $i \in [\ell]$ applying Lemma 3.6.2 as the induction step. The code $\mathcal{C}_i$ produced during each step will have bias at most $\eta < \eta_0$, so the hypothesis of Lemma 3.6.2 will be met at each level of the cascade. ∎

We are almost ready to prove our first main theorem establishing decodability close to the Gilbert–Varshamov bound. We will need parameters for an instantiation of Ta-Shma's code that achieves the desired distance and rate (which will be developed in Section 3.8.1) and a lemma relating splittability to the spectral properties of the graphs used in the construction (to be proven in Section 3.7.2).

Lemma 3.6.4 (Ta-Shma Codes I)**.** *For any $\beta > 0$, there are infinitely many values of $\varepsilon \in (0, 1/2)$ (with 0 as an accumulation point) such that for infinitely many values of $N \in \mathbb{N}$, there are explicit binary Ta-Shma codes $\mathcal{C}_{N,\varepsilon,\beta} \subseteq \mathbb{F}_2^N$ with*

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

(ii) rate $\Omega(\varepsilon^{2+\beta})$.

Furthermore, $\mathcal{C}_{N,\varepsilon,\beta}$ is the direct sum lifting of a base code $\mathcal{C}_0 \subseteq \mathbb{F}_2^{n_0}$ using the collection of walks $W[0, t-1]$ on the s-wide replacement product of two graphs G and H, with the following parameters:

- $s \geq s_0 \coloneqq \max\{128, 26/\beta\}$.
- *The inner graph H is a regular graph with $\sigma_2(H) \leq \lambda_2$, where $\lambda_2 = (16s^3 \log s)/s^{2s^2}$.*
- *The outer graph G is a regular graph with $\sigma_2(G) \leq \lambda_1$, where $\lambda_1 = \lambda_2^4/6$.*
- *The base code $\mathcal{C}_0$ is unique decodable in time $n_0^{O(1)}$ and has bias $\varepsilon_0 \leq \lambda_2^4/3$.*
- *The number of vertices t in the walks satisfies $\lambda_2^{2(1-5/s)(1-1/s)(t-1)} \leq \varepsilon$.*

Lemma 3.6.5. *Let $W(k)$ be either the collection $W[0, s-1]$ of walks of length s on the s-wide replacement product with outer graph G and inner graph H or the collection of walks over the vertex set $W[0, r]$, where $r \equiv -1 \pmod{s}$. Then $W(k)$ is τ-splittable with $\tau = \sigma_2(G) + 2\sigma_2(H) + \sigma_2(H)^2$.*

The statement of this first decoding theorem is more technical than Theorem 3.1.1, but it will be easier to prove while the latter will build on this theorem with a more careful tuning of parameters.

Theorem 3.6.6 (Main I)**.** *For every $\beta > 0$, there are infinitely many values $\varepsilon \in (0, 1/2)$ (with 0 an accumulation point) such that for infinitely many values of $N \in \mathbb{N}$ there are explicit binary linear Ta-Shma codes $\mathcal{C}_{N,\varepsilon,\beta} \subseteq \mathbb{F}_2^N$ with*

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

(ii) rate $\Omega(\varepsilon^{2+\beta})$, and

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

Proof. We proceed as follows:

1. Let $\eta_0 = 1/10$ and $\eta = 1/100$ (these choices are arbitrary; we only need $\eta_0 < 1/4$, $\eta < 1/16$, and $\eta < \eta_0$). Let $k_0 = k_0(\eta)$ be the constant from Theorem 3.6.1 with this value of η.

2. Given $\beta > 0$, Lemma 3.6.4 provides a value s_0 such that the direct sum lifting on the s-wide replacement product with $s \geq s_0$ can achieve a rate of $\Omega(\varepsilon^{2+\beta})$ for infinitely many $\varepsilon \in (0, 1/2)$. Choose s to be an integer larger than both k_0 and s_0 that also satisfies

$$s^2 \cdot \left(\frac{s}{16}\right)^{-s^2} \leq \frac{\eta^8}{4K}, \tag{3.3}$$

where K is the constant from Theorem 3.6.1.

3. Use Lemma 3.6.4 with this value of s to obtain graphs G and H and a base code $\mathcal{C}_0$ having the specified parameters λ_1, λ_2, ε_0, and t, with the additional requirement that $t = s^\ell$ for some integer ℓ. These parameter choices ensure that the resulting code $\mathcal{C}_{N,\varepsilon,\beta}$ has the desired distance and rate. Since $s \geq 128$, we have $\lambda_2 = (16s^3 \log s)/s^{2s^2} \leq s^{-s^2}$. From the choice of t satisfying $\lambda_2^{2(1-5/s)(1-1/s)(t-1)} \leq \varepsilon$, we deduce that $\ell = O(\log(\log(1/\varepsilon)))$. Note also that the bias ε_0 of the code $\mathcal{C}_0$ is smaller than η_0.

4. Create a code cascade with ℓ levels using the s-wide replacement product of the graphs G and H as in Section 3.5.2, starting with $\mathcal{C}_0$ and ending with the final code $\mathcal{C}_\ell = \mathcal{C}_{N,\varepsilon,\beta}$. As the total number of vertices in a walk is $t = s^\ell$, each level of the code cascade will use walks with s vertices. Let $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_\ell$ be the sequence of codes in this cascade.

5. In order to satisfy the splittability requirement of Lemma 3.6.3, the walk collection $W_i(s)$ at each level of the code cascade must be τ-splittable, where $\tau \leq \tau_0(\eta, s^2)$. (We use $k = s^2$ instead of $k = s$ in the requirement for a technical reason that will be clear in Section 3.8.2.) The bounds on the singular values of G and H and Lemma 3.6.5 ensure that

$$\tau = \sigma_2(G) + 2\sigma_2(H) + \sigma_2(H)^2 \leq 4\lambda_2 \leq 4s^{-s^2},$$

which is smaller than $\tau_0(\eta, s^2) = \eta^8/(K \cdot s^2 \cdot 2^{4s^2})$ by Eq. (3.3)

6. As all hypotheses of Lemma 3.6.3 are satisfied by the code cascade, we apply it to

conclude that $\mathcal{C}_{N,\varepsilon,\beta}$ is uniquely decodable in time

$$g(n_0) \cdot \prod_{i=1}^{\ell} n_{i-1}^{O(1/\tau_0(\eta,s)^4)} \leq N^{O(1)} \cdot \prod_{i=1}^{\ell} N^{O_\beta(1)} = N^{O_\beta(\log(\log(1/\varepsilon)))},$$

where we use that $\mathcal{C}_0$ is uniquely decodable in time $n_0^{O(1)}$, $1/\tau_0(\eta, s) = 2^{O(1/\beta)}$, $n_{i-1} < n_\ell = N$ for every $i \in [\ell]$, and $\ell = O(\log(\log(1/\varepsilon)))$.

■

In the code cascade constructed in Theorem 3.6.6, the final number of vertices in a walk is $t = s^\ell$, where s is a sufficiently large constant that does not depend on ε. The limited choices for t place some restrictions on the values of the final bias ε that can be achieved. To achieve any bias ε for $\mathcal{C}_\ell$ we need to choose the parameters more carefully, which will be done in Section 3.8.2 to yield our next main result.

Theorem 3.6.7 (Main II)**.** *For every $\beta > 0$ and every $\varepsilon > 0$ with β and ε 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 ε-balanced),

(ii) rate $\Omega(\varepsilon^{2+\beta})$, and

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

Ta-Shma obtained codes of rate $\Omega(\varepsilon^{2+\beta})$ with vanishing β as ε goes to zero. We obtain a unique decoding algorithm for this regime (with slightly slower decreasing β as ε vanishes). More precisely, using the parameters described in Section 3.8.3 and the running time analysis in Section 3.6.2, we obtain the following theorem which is our main result for unique decoding.

Theorem 3.6.8 (Main Unique Decoding (restatement of Theorem 3.1.1))**.** *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* ε*-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).*

Theorem 3.1.2 about gentle list decoding is proved in Section 3.8.4 after instantiating Ta-Shma codes in some parameter regimes in the preceding parts of Section 3.8.

3.6.2 Fixed Polynomial Time

In Theorem 3.6.7, a running time of $N^{O_\beta(\log(\log(1/\varepsilon)))}$ was obtained to decode Ta-Shma codes $\mathcal{C}_{N,\varepsilon,\beta}$ of distance $1/2 - \varepsilon/2$ and rate $\Omega(\varepsilon^{2+\beta})$ for constant $\beta > 0$ and block length N. The running time contains an exponent which depends on the bias ε and is therefore not fixed polynomial time. We show how to remove this dependence in this regime of $\beta > 0$ being an arbitrary constant. More precisely, we show the following.

Theorem 3.6.9 (Fixed PolyTime Unique Decoding)**.** *Let* $\beta > 0$ *be an arbitrary constant. 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* ε*-balanced),*

(ii) rate $\Omega(\varepsilon^{2+\beta})$*, and*

(iii) a unique decoding algorithm with fixed polynomial running time $(\log(1/\varepsilon))^{O(1)} \cdot N^{O_\beta(1)}$*.*

The list decoding framework finds a list of pairs $(z, y = \mathrm{dsum}(z))$ of size at most $N^{(1/\tau_0(\eta,k))^{O(1)}}$ at each level of the code cascade and recursively issues decoding calls to

all z in this list. Since the number of lifts in the cascade is $\Omega(\log(\log(1/\varepsilon)))$, we end up with an overall running time of $N^{O_\beta(\log(\log(1/\varepsilon)))}$.

We will describe a method of pruning these lists which will lead to fixed polynomial running time. Let $1/2 - \sqrt{\eta}$, where $\eta > 0$ is a constant, be the list decoding radius used for a unique decoding step in the cascade. To achieve fixed polynomial time we will prune this polynomially large list of words to a constant size at each inductive step in Lemma 3.6.3. As we are working with parameters within the Johnson bound, the actual list of codewords has constant size $(1/\eta)^{O(1)}$. At every step, we will be able to find a small sublist whose size only depends on η that has a certain covering property, and then issue decoding calls to this much smaller list.

Definition 3.6.10 (ζ-cover)**.** *Let $W(k) \subseteq [n]^k$, $\mathcal{C} \subseteq \mathbb{F}_2^n$ be a code, $A \subseteq \mathcal{C}$, and $\mathcal{L} = \{(z, \mathrm{dsum}_{W(k)}(z)) \mid z \in A\}$. We say that*

$$\mathcal{L}' = \{(z^{(1)}, \mathrm{dsum}_{W(k)}(z^{(1)})), \ldots, (z^{(m)}, \mathrm{dsum}_{W(k)}(z^{(m)}))\}$$

is a ζ-cover *of $\mathcal{L}$ if for every $(z, y) \in \mathcal{L}$, there exists some $(z', y') \in \mathcal{L}'$ with $\mathrm{bias}(z - z') > 1 - 2\zeta$ (that is, either $\Delta(z, z') < \zeta$ or $\Delta(z, z') > 1 - \zeta$).*

Lemma 3.6.11 (Cover Compactness)**.** *Let $W(k) \subseteq [n]^k$, $\mathcal{C} \subseteq \mathbb{F}_2^n$ be a linear η_0-balanced code, $\mathcal{C}' = \mathrm{dsum}_{W(k)}(\mathcal{C})$ be an η-balanced code, and $\tilde{y} \in \mathbb{F}_2^{W(k)}$. Define*

$$\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}') \coloneqq \left\{(z, \mathrm{dsum}_{W(k)}(z)) \mid z \in \mathcal{C}, \Delta\Big(\mathrm{dsum}_{W(k)}(z), \tilde{y}\Big) \leq \frac{1}{2} - \sqrt{\eta}\right\}.$$

Suppose $\mathcal{L}'$ is a ζ-cover of $\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}')$ for some $\zeta < 1/2$. Further, suppose that for every $(z', y') \in \mathcal{L}'$, we have $\Delta\Big(y', \tilde{y}\Big) \leq 1/2 - \sqrt{\eta}$. If $W(k)$ is a $(1 - 2\zeta, \eta)$-parity sampler, then there exists $\mathcal{L}'' \subseteq \mathcal{L}'$ with $\Big|\mathcal{L}''\Big| \leq 1/\eta$ which is a (2ζ)-cover of $\mathcal{L}$.

Proof. Build a graph where the vertices are pairs $(z', y') \in \mathcal{L}'$ and two vertices $(z^{(i)}, y^{(i)})$, $(z^{(j)}, y^{(j)})$ are connected iff $\mathrm{bias}(z^{(i)} - z^{(j)}) > 1 - 2\zeta$. Let $\mathcal{L}''$ be any maximal independent

set of this graph. Any two vertices $(z^{(i)}, y^{(i)}), (z^{(j)}, y^{(j)}) \in \mathcal{L}''$ have $\text{bias}(z^{(i)} - z^{(j)}) \leq 1 - 2\zeta$ and thus $\text{bias}(y^{(i)} - y^{(j)}) \leq \eta$ since $W(k)$ is a $(1 - 2\zeta, \eta)$-parity sampler. This means that $\{y'' \mid (z'', y'') \in \mathcal{L}''\}$ is a code of distance at least $1/2 - \eta/2$. By the condition that $\Delta(y'', \tilde{y}) \leq 1/2 - \sqrt{\eta}$ for all $(z'', y'') \in \mathcal{L}''$ and the Johnson bound, we have $\left|\mathcal{L}''\right| \leq 1/\eta$.

Finally, we will show that $\mathcal{L}''$ is a (2ζ)-cover of $\mathcal{L}$. Let $(z, y) \in \mathcal{L}$. As $\mathcal{L}'$ is a ζ-cover of $\mathcal{L}$, there exists a pair $(z', y') \in \mathcal{L}$ with $\text{bias}(z - z') > 1 - 2\zeta$, so z is within distance ζ of either z' or its complement $\overline{z'}$. The construction of $\mathcal{L}''$ as a maximal independent set ensures that there is some $(z'', y'') \in \mathcal{L}''$ with $\text{bias}(z' - z'') > 1 - 2\zeta$, so z'' is also within distance ζ of either z' or its complement $\overline{z'}$. Applying the triangle inequality in all of the possible cases, we see that either $\Delta(z, z'') < 2\zeta$ or $\Delta(z, z'') > 1 - 2\zeta$, which implies $\mathcal{L}''$ is a (2ζ)-cover of $\mathcal{L}$. ∎

To decode in fixed polynomial time, we use a modification of the list decoding result Theorem 3.6.1 that outputs a ζ-cover $\mathcal{L}'$ of the list of codewords $\mathcal{L}$ instead of the list itself. Theorem 3.6.1 recovers the list $\mathcal{L}$ by finding $\mathcal{L}'$ and unique decoding every element of it. To get $\mathcal{L}'$, we use the same algorithm, but stop before the final decoding step. This removes the unique decoding time $f(n)$ of the base code from the running time of the list decoding algorithm. We will apply Lemma 3.6.11 after each time we call this ζ-cover algorithm to pare the list down to a constant size before unique decoding; note that this loses a factor of 2 in the strength of the cover. To compensate for this, we will use a collection $W(k)$ with stronger parity sampling than required for Theorem 3.6.1. In that theorem, $W(k)$ was a $(1/2 + \eta_0/2, \eta)$-parity sampler to ensure that we obtained words within the list decoding radius $(1/4 - \eta_0/4)$ of the base code. By using a stronger parity sampler, the words in the pruned list $\mathcal{L}''$ will still be within the unique decoding radius even after accounting for the loss in the bias from cover compactness, which means decoding will still be possible at every level of the cascade. Fortunately, improving the parity sampling only requires increasing the walk length k by a constant multiplicative factor. The cover retrieval algorithm below will

be proven in Section 3.9.

Theorem 3.6.12 (Cover retrieval for direct sum lifting)**.** *Let $\eta_0 \in (0, 1/4)$ be a constant, $\eta \in (0, \eta_0)$, $\zeta = 1/8 - \eta_0/8$, and*

$$k \geq k_0'(\eta) \coloneqq \Theta(\log(1/\eta)).$$

Suppose $\mathcal{C} \subseteq \mathbb{F}_2^n$ is an η_0-balanced linear code and $\mathcal{C}' = \mathrm{dsum}_{W(k)}(\mathcal{C})$ is the direct sum lifting of $\mathcal{C}$ on a τ-splittable collection of walks $W(k)$. There exists an absolute constant $K > 0$ such that if

$$\tau \leq \tau_0(\eta, k) \coloneqq \frac{\eta^8}{K \cdot k \cdot 2^{4k}},$$

then the code $\mathcal{C}'$ is η-balanced, $W(k)$ is a $(1-2\zeta, \eta)$-parity sampler, and we have the following:

If $\tilde{y}$ is $(1/2 - \sqrt{\eta})$-close to $\mathcal{C}'$, then we can compute a ζ-cover $\mathcal{L}'$ of the list

$$\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}') \coloneqq \left\{(z, \mathrm{dsum}_{W(k)}(z)) \mid z \in \mathcal{C}, \Delta\Big(\mathrm{dsum}_{W(k)}(z), \tilde{y}\Big) \leq \frac{1}{2} - \sqrt{\eta}\right\}$$

in which $\Delta(y', \tilde{y}) \leq 1/2 - \sqrt{\eta}$ for every $(z', y') \in \mathcal{L}'$, in time

$$n^{O(1/\tau_0(\eta,k)^4)}.$$

Otherwise, we return $\mathcal{L}' = \emptyset$ with the same running time of the preceding case.

We now have all of the pieces necessary to prove Theorem 3.6.9. The process is essentially the same as our earlier unique decoding algorithm, except we use the cover retrieval algorithm from Theorem 3.6.12 instead of the full list decoding from Theorem 3.6.1. This allows us to insert a list pruning step in between obtaining the ζ-cover and calling the unique decoding algorithm for the previous level of the cascade.

Proof of Theorem 3.6.9. We use the code $\mathcal{C}_{N,\varepsilon,\beta}$ from Theorem 3.6.7 to get the desired dis-

tance and rate, with the slight modification of ensuring s is larger than k_0' from Theorem 3.6.12 rather than k_0 from Theorem 3.6.1.

Each level of the code cascade between $\mathcal{C}_{i-1}$ and $\mathcal{C}_i$ uses constant $\eta_0 < 1/4$ and $\eta < \min\{\eta_0, 1/16\}$, which allows for decoding in a similar fashion to Lemma 3.6.2 and Lemma 3.6.3. The difference is that we use Theorem 3.6.12 as the decoding step to obtain a ζ-cover $\mathcal{L}'$ of the list $\mathcal{L}(\tilde{y}, \mathcal{C}_{i-1}, \mathcal{C}_i)$ for $\tilde{y} \in \mathbb{F}_2^{n_i}$, where $\zeta = 1/8 - \eta_0/8$. By Lemma 3.6.11 and the fact that the walk collection is a $(1-2\zeta, \eta)$-parity sampler, $\mathcal{L}$ has a (2ζ)-cover $\mathcal{L}'' \subseteq \mathcal{L}'$ of size at most $1/\eta$. The covering property says that for every $(z, y) \in \mathcal{L}$, there exists some $(z'', y'') \in \mathcal{L}''$ such that z is within distance $2\zeta = 1/4 - \eta_0/4$ of either z'' or its complement $\overline{z''}$. This is the unique decoding radius of the η_0-balanced code $\mathcal{C}_{i-1}$, so we can recursively decode the list

$$\mathcal{L}'' \cup \{(\overline{z''}, \mathrm{dsum}(\overline{z''})) \mid (z'', \mathrm{dsum}(z'')) \in \mathcal{L}''\}$$

to obtain the complete list of codewords in $\mathcal{C}_{i-1}$.

Now we analyze the running time. On each level of the code cascade, we run the cover retrieval algorithm once to get $\mathcal{L}'$, prune the cover to get $\mathcal{L}''$, and then feed the union of $\mathcal{L}''$ and its complement (which has size at most $2/\eta$) into the unique decoding algorithm for the next level of the cascade. Letting $T_i(n_i)$ be the running time of unique decoding a single word in the code $\mathcal{C}_i \subseteq \mathbb{F}_2^{n_i}$, we have the following recurrence:

$$T_i(n_i) \leq n_i^{O(1/\tau_0(\eta,k)^4)} + \frac{2}{\eta} \cdot T_{i-1}(n_{i-1}) \quad \text{and} \quad T_0(n_0) = n_0^{O(1)}.$$

Note that the base code $\mathcal{C}_0$ has constant bias ε_0 and thus it has a fixed polynomial time decoding algorithm (e.g. Theorem 3.6.7). The height of the recursive call tree is the number of levels in the code cascade, which is $\ell = O(\log(\log(1/\varepsilon)))$, as in the proof of Theorem 3.6.6. Each node of this tree has a constant branching factor of $2/\eta$. Thus, the tree has $(\log(1/\varepsilon))^{O(1)}$ nodes, each of which costs at most $n_i^{O(1/\tau_0(\eta,k)^4)} \leq N^{O(1/\tau_0(\eta,k)^4)}$ time.

Furthermore, in this regime of $\beta > 0$ being a constant, k is constant as well as η, so we have $N^{O(1/\tau_0(\eta,k)^4)} = N^{O_\beta(1)}$ and the total running time is $(\log(1/\varepsilon))^{O(1)} \cdot N^{O_\beta(1)}$. ∎

3.7 Satisfying the List Decoding Framework Requirements

The list decoding framework of [AJQ+20] is capable of decoding codes obtained from direct sum liftings, provided they satisfy a few requisite properties. The framework was originally shown to work for expander walks; we need to adapt it to our case of a code cascade based on walks on the s-wide replacement product. We will start with a broad overview of the list decoding algorithm and point out where various requirements arise.

The problem of finding a list of codewords in a direct sum lifting close to a received word can be viewed as finding approximate solutions to a k-XOR instance. This is done by solving a particular SOS program and rounding the resulting solution. The algorithm is unable to perform rounding if the k-XOR instance is based on an arbitrary collection of walks $W(k)$; it can only handle liftings in which $W(k)$ satisfies a property called *tensoriality*. If $W(k)$ is tensorial, the SOS local variables in the solution can be approximated by product distributions, which will allow us to obtain a list of solutions by independent rounding. Tensoriality for expander walks is a consequence of a simpler property known as *splittability*, which is a certain measure of the expansion of a walk collection.

Unfortunately, the list returned by the rounding process will not contain codewords directly—instead, we only get a guarantee that all of the codewords we are looking for have a weak agreement (just over 1/2) with something on this list. We will find the desired codewords by relying on the parity sampling of $W(k)$. If $W(k)$ is a sufficiently strong parity sampler, weak agreement in the lifted space corresponds to a much stronger agreement in the ground space. This will allow us to recover the codewords using the unique decoding algorithm of the base code.

To recap, applying the list decoding framework in our setting requires doing the following:

1. Proving parity sampling for the walks used in the code cascade (Section 3.7.1).

2. Showing that the walk collection of the s-wide replacement product is splittable (Section 3.7.2).

3. Making Ta-Shma's construction compatible with the Sum-of-Squares machinery (Section 3.7.3) and then obtaining tensoriality from splittability (Section 3.7.4).

An additional complication is introduced by using a code cascade instead of a single decoding step: the above requirements need to be satisfied at every level of the cascade. The details of the proofs will often differ between the first level of the cascade, which is constructed using walks on the s-wide replacement product, and higher levels, which are walks on a directed graph whose vertices are walks themselves. Once we have established all of the necessary properties, we will instantiate the list decoding framework in Section 3.9.

We will first define some convenient notation which will be used throughout this section.

Notation 3.7.1. *Let G be a d_1-regular outer graph and H be a d_2-regular inner graph used in Ta-Shma's s-wide replacement product.*

Let $0 \leq k_1 \leq k_2$ be integers. We define $W[k_1, k_2]$ to be the set of all walks starting at time k_1 and ending at time k_2 in Ta-Shma's construction. More precisely, since G and H are regular graphs, the collection $W[k_1, k_2]$ contains all walks obtained by sampling a uniform vertex $(v, h) \in V(G) \times V(H)$ and applying the operator

$$(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2-1}(\mathsf{I} \otimes \mathsf{A}_H) \cdots (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_1}(\mathsf{I} \otimes \mathsf{A}_H),$$

where the index i of each G_i is taken modulo s. Observe that when $k_1 = k_2$, we have $W[k_1, k_2] = V(G) \times V(H)$.

We define a family of Markov operators which will play a similar role to the graphs $\widehat{G}_i$ from the cascade described in Section 3.5.1, but for Ta-Shma's construction rather than expander walks.

Definition 3.7.2 (Split Operator)**.** *Let $0 \leq k_1 \leq k_2 < k_3$. We define the graph walk split operator*

$$\mathsf{S}_{k_1,k_2,k_3} \colon \mathbb{R}^{W[k_2+1,k_3]} \to \mathbb{R}^{W[k_1,k_2]}$$

such that for every $f \in \mathbb{R}^{W[k_2+1,k_3]}$,

$$\left(\mathsf{S}_{k_1,k_2,k_3}(f)\right)(w) \coloneqq \mathbb{E}_{w' : ww' \in W[k_1,k_3]}[f(w')],$$

where ww' denotes the concatenation of the walks w and w'. The operator S_{k_1,k_2,k_3} can be defined more concretely in matrix form such that for every $w \in W[k_1,k_2]$ and $w' \in W[k_2+1,k_3]$,

$$\left(\mathsf{S}_{k_1,k_2,k_3}\right)_{w,w'} = \frac{\mathbb{1}_{ww' \in W[k_1,k_3]}}{|\{\tilde{w} : w\tilde{w} \in W[k_1,k_3]\}|} = \frac{\mathbb{1}_{ww' \in W[k_1,k_3]}}{d_2^{2(k_3-k_2)}}.$$

3.7.1 Parity Sampling for the Code Cascade

To be able to apply the list decoding machinery to the code cascade $\mathcal{C}_0 \subseteq \mathbb{F}_2^{n_0}, \mathcal{C}_1 \subseteq \mathbb{F}_2^{n_1}, \ldots, \mathcal{C}_\ell \subseteq \mathbb{F}_2^{n_\ell}$, we need the direct sum lifting at every level to be a parity sampler. The first level in the cascade uses walks directly on the s-wide replacement product, which we can show is a good parity sampler using the spectral properties proven in Section 3.4.3. However, it will be more convenient for calculating parameters later on to prove a weaker result, which will suffice for our purposes since we only need to obtain constant bias for every level of the cascade. We analyze the parity sampling of these walks with the same strategy Ta-Shma employed to show parity sampling for walks on expander graphs (which resulted in Theorem 3.5.2).

Claim 3.7.3. *Let $W[0,s-1]$ be the collection of walks on the s-wide replacement product of the graphs G and H and $z \in \mathbb{F}_2^{V(G)}$ be a word with $\mathrm{bias}(z) \leq \eta_0$. Let P_z be the diagonal matrix with entries $(\mathsf{P}_z)_{(v,h),(v,h)} = (-1)^{z_v}$ for $(v,h) \in V(G) \times V(H)$. If $\sigma_2((\mathsf{I} \otimes \mathsf{A}_H)G_i(\mathsf{I} \otimes \mathsf{A}_H)) \leq \gamma$*

for all $0 \leq i \leq s-2$*, then*

$$\left\| \prod_{i=0}^{s-2} (\mathsf{I} \otimes \mathsf{A}_H) \mathsf{G}_i (\mathsf{I} \otimes \mathsf{A}_H) \mathsf{P}_z \right\|_2 \leq (\eta_0 + 2\gamma)^{\lfloor (s-1)/2 \rfloor}.$$

Proof. Let $0 \leq j < s-2$ be even. Take a vector $v \in \mathbb{R}^{V(G) \times V(H)}$ with $\|v\|_2 = 1$ and let $v^{\|}$ and $v^{\perp}$ be its parallel and orthogonal components to the all ones vector. For $0 \leq i \leq s-2$, let $\mathsf{A}_i = (\mathsf{I} \otimes \mathsf{A}_H) \mathsf{G}_i (\mathsf{I} \otimes \mathsf{A}_H)$. Consider two terms $\mathsf{A}_{j+1} \mathsf{P}_z \mathsf{A}_j \mathsf{P}_z$ of the product appearing in the claim. Since P_z is unitary, $\left\| \mathsf{A}_{j+1} \mathsf{P}_z \mathsf{A}_j \mathsf{P}_z \right\|_2 = \left\| \mathsf{A}_{j+1} \mathsf{P}_z \mathsf{A}_j \right\|_2$. We have

$$\begin{aligned}
\left\| \mathsf{A}_{j+1} \mathsf{P}_z \mathsf{A}_j v \right\|_2 &\leq \left\| \mathsf{A}_{j+1} \mathsf{P}_z \mathsf{A}_j v^{\|} \right\|_2 + \left\| \mathsf{A}_{j+1} \mathsf{P}_z \mathsf{A}_j v^{\perp} \right\|_2 \\
&\leq \left\| \mathsf{A}_{j+1} \mathsf{P}_z \mathsf{A}_j v^{\|} \right\|_2 + \left\| \mathsf{A}_j v^{\perp} \right\|_2 \\
&\leq \left\| \mathsf{A}_{j+1} \mathsf{P}_z v^{\|} \right\|_2 + \sigma_2(\mathsf{A}_j) \\
&\leq \left\| \mathsf{A}_{j+1} (\mathsf{P}_z v^{\|})^{\|} \right\|_2 + \left\| \mathsf{A}_{j+1} (\mathsf{P}_z v^{\|})^{\perp} \right\|_2 + \sigma_2(\mathsf{A}_j) \\
&\leq \left\| (\mathsf{P}_z v^{\|})^{\|} \right\|_2 + \sigma_2(\mathsf{A}_{j+1}) + \sigma_2(\mathsf{A}_j) \\
&\leq \eta_0 + 2\gamma.
\end{aligned}$$

Applying this inequality to every two terms of the product, the result follows. ∎

Corollary 3.7.4. *Let* $W[0, s-1]$ *be the collection of walks on the* s*-wide replacement product of the graphs* G *and* H *and* $\eta_0 > 0$*. If* $\sigma_2((\mathsf{I} \otimes \mathsf{A}_H) \mathsf{G}_i (\mathsf{I} \otimes \mathsf{A}_H)) \leq \gamma$ *for all* $0 \leq i \leq s-2$*, then* $W[0, s-1]$ *is an* (η_0, η)*-parity sampler, where* $\eta = (\eta_0 + 2\gamma)^{\lfloor (s-1)/2 \rfloor}$*.*

Proof. Let $z \in \mathbb{F}_2^n$ have bias at most η_0. The bias of $\mathrm{dsum}_{W[0,s-1]}(z)$ is given by

$$\mathrm{bias}(\mathrm{dsum}_{W[0,s-1]}(z)) = \left| \left\langle \mathbf{1}, \mathsf{P}_z \left(\prod_{i=0}^{s-2} (\mathsf{I} \otimes \mathsf{A}_H) \mathsf{G}_i (\mathsf{I} \otimes \mathsf{A}_H) \mathsf{P}_z \right) \mathbf{1} \right\rangle \right|,$$

where P_z is the diagonal matrix with entries $(\mathsf{P}_z)_{(v,h),(v,h)} = (-1)^{z_v}$ for $(v, h) \in V(G) \times V(H)$

and $\mathbf{1}$ is the all-ones vector.[6] Since P_z is unitary, we have

$$\text{bias}(\text{dsum}_{W[0,s-1]}(z)) \leq \left\| \prod_{i=0}^{s-2} (\mathsf{I} \otimes \mathsf{A}_H) \mathsf{G}_i (\mathsf{I} \otimes \mathsf{A}_H) \mathsf{P}_z \right\|_2 \leq (\eta_0 + 2\gamma)^{\lfloor (s-1)/2 \rfloor} = \eta$$

by Claim 3.7.3. Hence $W[0, s-1]$ is an (η_0, η)-parity sampler. ∎

For higher levels of the cascade, we need to prove parity sampling for collections of walks over walks. Since the walks on the first level contain s vertices, when we take walks on higher levels, the operator linking different walks together will always use G_{s-1} as the walk operator for the G step. Thus we can consider a more specific form of the split operator where we split at a time parameter that is one less than a multiple of s.

Definition 3.7.5. *Let $r \equiv -1 \pmod{s}$ be a positive integer. We define the operator $\mathsf{S}^{\triangle}_{r,r}$ as*

$$\mathsf{S}^{\triangle}_{r,r} = \mathsf{S}_{k_1,k_2,k_3},$$

where $k_1 = 0$, $k_2 = r$, and $k_3 = 2r+1$. In this case, $W[k_1, k_2] = W[k_2 + 1, k_3]$.

All levels of the code cascade beyond the first use walks generated by the directed operator $\mathsf{S}^{\triangle}_{r,r}$. Proving parity sampling for these walks is analogous to the proof of Corollary 3.7.4, but slightly simpler since the walk operator doesn't change with each step.

Claim 3.7.6. *Let $r \equiv -1 \pmod{s}$ be a positive integer and $z \in \mathbb{F}_2^{W[0,r]}$ be a word with $\text{bias}(z) \leq \eta_0$. Let $\widetilde{\mathsf{P}}_z$ be the diagonal matrix with entries $(\widetilde{\mathsf{P}}_z)_{w,w} = (-1)^{z_w}$ for $w \in W[0,r]$. For every integer $k \geq 1$, we have*

$$\left\| \left(\mathsf{S}^{\triangle}_{r,r} \widetilde{\mathsf{P}}_z \right)^{k-1} \right\|_2 \leq \left(\eta_0 + 2 \cdot \sigma_2 \left(\mathsf{S}^{\triangle}_{r,r} \right) \right)^{\lfloor (k-1)/2 \rfloor}.$$

6. This is slightly different from the expression for the bias given in Section 3.4.3, but both are equal since moving on the H component of the graph doesn't affect the bit assigned to a vertex.

Proof. Take a vector $v \in \mathbb{R}^{W[0,r]}$ with $\|v\|_2 = 1$ and let $v^{\parallel}$ and $v^{\perp}$ be its parallel and orthogonal components to the all ones vector. Since $\widetilde{\mathsf{P}}_z$ is unitary, $\left\|\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z\right\|_2 = \left\|\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z\mathsf{S}^{\triangle}_{r,r}\right\|_2$. We have

$$\begin{aligned}
\left\|\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z\mathsf{S}^{\triangle}_{r,r}v\right\|_2 &\leq \left\|\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z\mathsf{S}^{\triangle}_{r,r}v^{\parallel}\right\|_2 + \left\|\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z\mathsf{S}^{\triangle}_{r,r}v^{\perp}\right\|_2 \\
&\leq \left\|\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z\mathsf{S}^{\triangle}_{r,r}v^{\parallel}\right\|_2 + \left\|\mathsf{S}^{\triangle}_{r,r}v^{\perp}\right\|_2 \\
&\leq \left\|\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z v^{\parallel}\right\|_2 + \sigma_2(\mathsf{S}^{\triangle}_{r,r}) \\
&\leq \left\|\mathsf{S}^{\triangle}_{r,r}(\widetilde{\mathsf{P}}_z v^{\parallel})^{\parallel}\right\|_2 + \left\|\mathsf{S}^{\triangle}_{r,r}(\widetilde{\mathsf{P}}_z v^{\parallel})^{\perp}\right\|_2 + \sigma_2(\mathsf{S}^{\triangle}_{r,r}) \\
&\leq \left\|(\widetilde{\mathsf{P}}_z v^{\parallel})^{\parallel}\right\|_2 + \sigma_2(\mathsf{S}^{\triangle}_{r,r}) + \sigma_2(\mathsf{S}^{\triangle}_{r,r}) \\
&\leq \eta_0 + 2\cdot\sigma_2(\mathsf{S}^{\triangle}_{r,r}).
\end{aligned}$$

As $\left\|(\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z)^{k-1}\right\|_2 \leq \left\|(\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z)^2\right\|^{\lfloor (k-1)/2\rfloor}$, the result follows. ∎

Corollary 3.7.7. *Let $r \equiv -1 \pmod{s}$ be a positive integer and $\eta_0 > 0$. The collection of walks $W(k)$ with k vertices over the vertex set $W[0,r]$ using random walk operator $\mathsf{S}^{\triangle}_{r,r}$ is an (η_0, η)-parity sampler, where $\eta = (\eta_0 + 2\cdot\sigma_2(\mathsf{S}^{\triangle}_{r,r}))^{\lfloor (k-1)/2\rfloor}$.*

Proof. Let $z \in \mathbb{F}_2^{W[0,r]}$ have bias at most η_0. The bias of the direct sum lifting of z is given by

$$\mathrm{bias}(\mathrm{dsum}_{W(k)}(z)) = \left|\left\langle \mathbf{1}, \widetilde{\mathsf{P}}_z(\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z)^{k-1}\mathbf{1}\right\rangle\right|,$$

where $\widetilde{\mathsf{P}}_z$ is the diagonal matrix with entries $(\widetilde{\mathsf{P}}_z)_{w,w} = (-1)^{z_w}$ for $w \in W[0,r]$ and $\mathbf{1}$ is the all-ones vector. Since $\widetilde{\mathsf{P}}_z$ is unitary, we have

$$\left|\left\langle \mathbf{1}, \widetilde{\mathsf{P}}_z(\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z)^{k-1}\mathbf{1}\right\rangle\right| \leq \left\|\left(\mathsf{S}^{\triangle}_{r,r}\widetilde{\mathsf{P}}_z\right)^{k-1}\right\|_2 \leq \left(\eta_0 + 2\cdot\sigma_2\left(\mathsf{S}^{\triangle}_{r,r}\right)\right)^{\lfloor (k-1)/2\rfloor} = \eta$$

by Claim 3.7.6. Hence $W(k)$ is an (η_0, η)-parity sampler. ∎

3.7.2 Splittability of Ta-Shma's Construction

We investigate the splittability of the collection of walks generated by Ta-Shma's construction. In order to formally define this property, we will need the concept of an interval splitting tree, which describes how a walk is split into smaller and smaller pieces.

Definition 3.7.8 (Interval Splitting Tree)**.** *We say that a binary rooted tree $\mathcal{T}$ is a k-interval splitting tree if it has exactly k leaves and*

- *the root of $\mathcal{T}$ is labeled with $(0, m, k-1)$ for some $m \in \{0, 1, \ldots, k-2\}$, and*
- *each non-leaf non-root vertex v of $\mathcal{T}$ is labeled with (k_1, k_2, k_3) for some integer $k_2 \in [k_1, k_3 - 1]$. Suppose (k_1', k_2', k_3') is the label assigned to the parent of v. If v is a left child, we must have $k_1 = k_1'$ and $k_3 = k_2'$; otherwise, we must have $k_1 = k_2' + 1$ and $k_3 = k_3'$.*

Given an interval splitting tree $\mathcal{T}$, we can naturally associate a split operator S_{k_1,k_2,k_3} to each internal node (k_1, k_2, k_3). The splittability of a collection $W[0, k-1]$ of k-tuples is a notion of expansion at every node in the splitting tree.

Definition 3.7.9 ($(\mathcal{T}, \tau)$-splittability)**.** *The collection $W[0, k-1]$ is said to be $(\mathcal{T}, \tau)$-splittable if $\mathcal{T}$ is a k-interval splitting tree and*

$$\sigma_2(\mathsf{S}_{k_1,k_2,k_3}) \leq \tau$$

for every internal node (k_1, k_2, k_3) of $\mathcal{T}$.

If there exists some k-interval splitting tree $\mathcal{T}$ such that $W[0, k-1]$ is $(\mathcal{T}, \tau)$-splittable, then $W[0, k-1]$ will be called τ-splittable.

In order to prove that the collection of walks in Ta-Shma's construction is splittable, a split operator S_{k_1,k_2,k_3} can be related to the walk operator $(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)$ as shown

below. This structural property will allow us to deduce spectral properties of S_{k_1,k_2,k_3} from the spectrum of $(\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)$.

Lemma 3.7.10. *Let $0 \leq k_1 \leq k_2 < k_3$. Suppose G is a d_1-regular outer graph on vertex set $[n]$ with walk operator G_{k_2} used at step k_2 of a walk on the s-wide replacement product and H is a d_2-regular inner graph on vertex set $[m]$ with normalized random walk operator A_H. Then there are orderings of the rows and columns of the representations of S_{k_1,k_2,k_3} and A_H as matrices such that*

$$\mathsf{S}_{k_1,k_2,k_3} = \Big((\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)\Big) \otimes \mathsf{J}/d_2^{2(k_3-k_2-1)},$$

where $\mathsf{J} \in \mathbb{R}^{[d_2]^{2(k_2-k_1)} \times [d_2]^{2(k_3-k_2-1)}}$ is the all ones matrix.

Proof. Partition the set of walks $W[k_1, k_2]$ into the sets $W_{1,1}, \ldots, W_{n,m}$, where $w \in W_{i,j}$ if the last vertex of the walk $w_{k_2} = (v_{k_2}, h_{k_2})$ satisfies $v_{k_2} = i$ and $h_{k_2} = j$. Similarly, partition $W[k_2 + 1, k_3]$ into the sets $W'_{1,1}, \ldots, W'_{n,m}$, where $w' \in W'_{i,j}$ if the first vertex of the walk $w'_1 = (v_1, h_1)$ satisfies $v_1 = i$ and $h_1 = j$. Note that $\left|W_{i,j}\right| = d_2^{2(k_2-k_1)}$ and $\left|W'_{i,j}\right| = d_2^{2(k_3-k_2-1)}$ for all $(i, j) \in [n] \times [m]$, since there are d_2^2 choices for each step of the walk.

Now order the rows of the matrix S_{k_1,k_2,k_3} so that all of the rows corresponding to walks in $W_{1,1}$ appear first, followed by those for walks in $W_{1,2}$, and so on in lexicographic order of the indices (i, j) of $W_{i,j}$, with an arbitrary order within each set. Do a similar re-ordering of the columns for the sets $W'_{1,1}, \ldots, W'_{1,m}$. Observe that

$$\begin{aligned}\left(\mathsf{S}_{k_1,k_2,k_3}\right)_{w,w'} &= \frac{\mathbb{1}_{ww' \in W[k_1,k_3]}}{d_2^{2(k_3-k_2)}} \\ &= \frac{d_2^2 \cdot (\text{weight of transition from } (v_{k_2}, h_{k_2}) \text{ to } (v'_1, h'_1) \text{ in } (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H))}{d_2^{2(k_3-k_2)}},\end{aligned}$$

which only depends on the adjacency of the last vertex of w and the first vertex of w'. If

the vertices $w_{k_2} = (v_{k_2}, h_{k_2})$ and $w'_1 = (v_1, h_1)$ are adjacent, then

$$\left(\mathsf{S}_{k_1,k_2,k_3}\right)_{w,w'} = \left((\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)\right)_{(v_{k_2},h_{k_2}),(v'_1,h'_1)} / d_2^{2(k_3-k_2-1)},$$

for every $w \in W_{w_{k_2}}$ and $w' \in W'_{w_{k_1}}$; otherwise, $\left(\mathsf{S}_{k_1,k_2,k_3}\right)_{w,w'} = 0$. Since the walks in the rows and columns are sorted according to their last and first vertices, respectively, the matrix S_{k_1,k_2,k_3} exactly matches the tensor product $((\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)) \otimes \mathsf{J}/d_2^{2(k_3-k_2-1)}$. ∎

Corollary 3.7.11. *Let $0 \le k_1 \le k_2 < k_3$. Suppose G is a d_1-regular outer graph with walk operator G_{k_2} used at step k_2 of a walk on the s-wide replacement product and H is a d_2-regular inner graph with normalized random walk operator A_H. Then*

$$\sigma_2(\mathsf{S}_{k_1,k_2,k_3}) = \sigma_2((\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)).$$

Proof. Using Lemma 3.7.10 and the fact that

$$\sigma_2(((\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)) \otimes \mathsf{J}/d_2^{2(k_3-k_2-1)}) = \sigma_2((\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)),$$

the result follows. ∎

Remark 3.7.12. *Corollary 3.7.11 is what causes the splittability argument to break down for Ta-Shma's original construction, as $\sigma_2(\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)) = 1$.*

By combining this result with the spectral bound from Fact 3.4.4, we find that the collection of walks of length s on the s-wide replacement product is $(\mathcal{T}, \tau)$-splittable for any splitting tree $\mathcal{T}$, where τ is controlled by the second singular values of the graphs G and H. This analysis can also be applied to walks on higher levels of the cascade where the vertex set is $W[0, r]$.

Corollary 3.7.13 (Restatement of Lemma 3.6.5)**.** *The collection of walks $W[0, s-1]$ on the s-wide replacement product with outer graph G and inner graph H and the collection of*

walks $W(k)$ on the vertex set $W[0,r]$ with random walk operator $\mathsf{S}^{\triangle}_{r,r}$ and $r \equiv -1 \pmod{s}$ are both τ-splittable with $\tau = \sigma_2(G) + 2\sigma_2(H) + \sigma_2(H)^2$.

Proof. By Corollary 3.7.11 and Fact 3.4.4, the split operator S_{k_1,k_2,k_3} for any $0 \le k_1 \le k_2 < k_3$ satisfies

$$\sigma_2(\mathsf{S}_{k_1,k_2,k_3}) = \sigma_2((\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_{k_2}(\mathsf{I} \otimes \mathsf{A}_H)) \le \sigma_2(G) + 2\sigma_2(H) + \sigma_2(H)^2,$$

so $W[0, s-1]$ is τ-splittable with $\tau = \sigma_2(G) + 2\sigma_2(H) + \sigma_2(H)^2$, as any internal node (k_1, k_2, k_3) of any s-interval splitting tree will have $\sigma_2(\mathsf{S}_{k_1,k_2,k_3}) \le \tau$. The split operators of any k-interval splitting tree for the collection $W(k)$ are of the form S_{k_1,k_2,k_3} with $k_1 \equiv 0 \pmod{s}$ and $k_2, k_3 \equiv -1 \pmod{s}$, which means $W(k)$ is τ-splittable as well. ∎

3.7.3 Integration with Sum-of-Squares

Before defining tensoriality and obtaining it in our setting, we examine how the Sum-of-Squares hierarchy is used in the list decoding algorithm in more detail.

SOS Preliminaries: p-local PSD Ensembles

The SOS hierarchy gives a sequence of increasingly tight semidefinite programming relaxations for several optimization problems, including CSPs. Since we will use relatively few facts about the SOS hierarchy, already developed in the analysis of Barak, Raghavendra and Steurer [BRS11], we will adapt their notation of *p-local distributions* to describe the relaxations.

Solutions to a semidefinite relaxation of a CSP on n boolean variables using p levels of the SOS hierarchy induce probability distributions μ_S over $\mathbb{F}_2^S$ for any set $S \subseteq [n]$ with $|S| \le p$. These distributions are consistent on intersections: for $T \subseteq S \subseteq [n]$, we have $\mu_{S|T} = \mu_T$, where $\mu_{S|T}$ denotes the restriction of the distribution μ_S to the set T. We use

these distributions to define a collection of random variables $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$ taking values in $\mathbb{F}_2$ such that for any set S with $|S| \leq p$, the collection of variables $\{\mathbf{Z}_i\}_{i \in S}$ has joint distribution μ_S. Note that the entire collection $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ *may not* have a joint distribution: this property is only true for sub-collections of size at most p. We will refer to the collection $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ as a *p-local ensemble* of random variables.

For any $T \subseteq [n]$ with $|T| \leq p-2$ and any $\xi \in \mathbb{F}_2^T$, we can define a $(p-|T|)$-local ensemble $\{\mathbf{Z}'_1, \ldots, \mathbf{Z}'_n\}$ by "conditioning" the local distributions on the event $\mathbf{Z}_T = \xi$, where $\mathbf{Z}_T$ is shorthand for the collection $\{\mathbf{Z}_i\}_{i \in T}$. For any S with $|S| \leq p-|T|$, we define the distribution of $\mathbf{Z}'_S$ as $\mu'_S := \mu_{S \cup T} | \{\mathbf{Z}_T = \xi\}$.

Finally, the semidefinite program also ensures that for any such conditioning, the conditional covariance matrix

$$\mathsf{M}_{(S_1,\alpha_1)(S_2,\alpha_2)} \;=\; \mathrm{Cov}\left(\mathbb{1}_{[\mathbf{Z}'_{S_1}=\alpha_1]}, \mathbb{1}_{[\mathbf{Z}'_{S_2}=\alpha_2]}\right)$$

is positive semidefinite, where $|S_1|, |S_2| \leq (p-|T|)/2$. Here, for each pair S_1, S_2 the covariance is computed using the joint distribution $\mu'_{S_1 \cup S_2}$. In this paper, we will only consider p-local ensembles such that for every conditioning on a set of size at most $(p-2)$, the conditional covariance matrix is PSD. We will refer to these as *p-local PSD ensembles*. We will also need a simple corollary of the above definitions.

Fact 3.7.14. *Let $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ be a p-local PSD ensemble and $W(k) \subseteq [n]^k$ For $1 \leq i < k$, define $W(i) \subseteq [n]^i$ to be the collection of tuples of size i appearing in elements of $W(k)$. For all $p' \leq p/2$, the collection $\left\{\mathbf{Z}_{\mathrm{set}(w)}\right\}_{w \in W(\leq p')}$ is a (p/p')-local PSD ensemble, where $W(\leq p') = \bigcup_{i=1}^{p'} W(i)$.*

For random variables $\mathbf{Z}_S$ in a p-local PSD ensemble, we use the notation $\{\mathbf{Z}_S\}$ to denote the distribution of $\mathbf{Z}_S$ (which exists when $|S| \leq p$). As we will work with ordered tuples of variables instead of sets, we define $\mathbf{Z}_w$ for $w \in [n]^k$ based on the set $S_w = \mathrm{set}(w)$, taking

care that repeated elements of w are always assigned the same value.

Definition 3.7.15 (Plausible assignment)**.** *Given $w = (w_1, \dots, w_k) \in [n]^k$ and an assignment $\alpha \in \mathbb{F}_2^w$, we say that α is plausible for w if there are no distinct $i, j \in [k]$ such that $w_i = w_j$ but $\alpha_i \neq \alpha_j$.*

The distribution $\{\mathbf{Z}_w\} = \mu_w$ is defined as $\mu_w(\alpha) = \mu_{S_w}(\alpha|_{S_w})$ if $\alpha \in \mathbb{F}_2^w$ is plausible for w, and $\mu_w(\alpha) = 0$ otherwise.

Tensoriality

A key algorithm in the list decoding framework is propagation rounding (Algorithm 3.7.16), which solves a CSP to find solutions close to a codeword. Suppose $W(k) \subseteq [n]^k$ is a collection of walks, or more generally, a collection of any k-tuples. The algorithm starts with a local PSD ensemble $\{\mathbf{Z}_1, \dots, \mathbf{Z}_n\}$ which is the solution to an SOS program for list decoding. Propagation rounding takes this solution and conditions some of the variables according to a random assignment to these variables to yield another local PSD ensemble $\mathbf{Z}'$.

Algorithm 3.7.16 (Propagation Rounding Algorithm, adapted from [AJQ+20]).

Input *An $(L+2k)$-local PSD ensemble $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_n\}$ and collection $W(k) \subseteq [n]^k$.*

Output *A random assignment $(\sigma_1, \ldots, \sigma_n) \in \mathbb{F}_2^n$ and $2k$-local PSD ensemble $\mathbf{Z}'$.*

1. *Choose $m \in \{1, \ldots, L/k\}$ uniformly at random.*
2. *For $j = 1, \ldots, m$, sample a walk w_j independently and uniformly from $W(k)$.*
3. *Write $S = \bigcup_{j=1}^{m} \mathrm{set}(w_j)$ for the set of the seed vertices.*
4. *Sample an assignment $\sigma : S \to \mathbb{F}_2$ according to the local distribution $\{\mathbf{Z}_S\}$.*
5. *Set $\mathbf{Z}' = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_n | \mathbf{Z}_S = \sigma\}$, i.e. the local ensemble $\mathbf{Z}$ conditioned on agreeing with σ.*
6. *For all $i \in [n]$, sample independently $\sigma_i \sim \{\mathbf{Z}'_i\}$.*
7. *Output $(\sigma_1, \ldots, \sigma_n)$ and $\mathbf{Z}'$.*

If the collection $W(k) \subseteq [n]^k$ used in the direct sum lifting is amenable to SOS rounding, the conditioned ensemble $\mathbf{Z}'$ will be able to recover a word close to some codeword on the list. This is quantified by the following *tensorial* properties. We will see shortly how splittability will be used to obtain tensoriality in our setting.

Definition 3.7.17 (Tensorial Walk Collection). *Let $W(k) \subseteq [n]^k$, $\mu \in [0,1]$, and $L \in \mathbb{N}$. Define Ω to be the set of all tuples (m, S, σ) obtainable in propagation rounding (Algorithm 3.7.16) on $W(k)$ with SOS degree parameter L. We say that $W(k)$ is (μ, L)-tensorial if the local PSD ensemble $\mathbf{Z}'$ returned by propagation rounding satisfies*

$$\mathop{\mathbb{E}}_{\Omega} \mathop{\mathbb{E}}_{w \in W(k)} \left\| \{\mathbf{Z}'_w\} - \left\{\mathbf{Z}'_{w(1)}\right\} \cdots \left\{\mathbf{Z}'_{w(k)}\right\} \right\|_1 \leq \mu. \tag{3.4}$$

The framework actually uses a strengthening of the above property, in which variables

for pairs of walks chosen independently approximately behave as a product.

Definition 3.7.18 (Two-Step Tensorial Walk Collection)**.** *Let $W(k) \subseteq [n]^k$, $\mu \in [0,1]$, and $L \in \mathbb{N}$. Define Ω to be the set of all tuples (m, S, σ) obtainable in propagation rounding (Algorithm 3.7.16) on $W(k)$ with SOS degree parameter L. We say that $W(k)$ is (μ, L)-two-step tensorial if it is (μ, L)-tensorial and the local PSD ensemble $\mathbf{Z}'$ returned by propagation rounding satisfies the additional condition*

$$\mathop{\mathbb{E}}_{\Omega} \mathop{\mathbb{E}}_{w,w' \in W(k)} \left\| \{\mathbf{Z}'_w \mathbf{Z}'_{w'}\} - \left\{\mathbf{Z}'_w\right\}\left\{\mathbf{Z}'_{w'}\right\} \right\|_1 \leq \mu.$$

From Directed to Undirected

In order to apply the list decoding framework using the directed split operator S_{k_1,k_2,k_3}, we will replace it with the symmetrized version

$$\mathcal{U}(\mathsf{S}_{k_1,k_2,k_3}) = \begin{pmatrix} 0 & \mathsf{S}_{k_1,k_2,k_3} \\ \left(\mathsf{S}_{k_1,k_2,k_3}\right)^\dagger & 0 \end{pmatrix}$$

and show how $\mathcal{U}(\mathsf{S}_{k_1,k_2,k_3})$ corresponds to a particular undirected graph.

Definition 3.7.19. *Let $0 \leq k_1 \leq k_2 < k_3$. We define the operator $\mathfrak{S}_{k_2,k_3,k_1} \colon \mathbb{R}^{W[k_1,k_2]} \to \mathbb{R}^{W[k_2+1,k_3]}$ such that for every $f \in \mathbb{R}^{W[k_1,k_2]}$,*

$$\left(\mathfrak{S}_{k_2,k_3,k_1}(f)\right)(w') \coloneqq \mathbb{E}_{w : ww' \in W[k_1,k_3]}[f(w)],$$

for every $w' \in W[k_2+1, k_3]$.

The operator $\mathcal{U}(\mathsf{S}_{k_1,k_2,k_3})$ defines an undirected weighted bipartite graph on the vertices $W[k_1,k_2] \cup W[k_2+1,k_3]$. We can see that $\mathfrak{S}_{k_2,k_3,k_1}$ is the adjoint of S_{k_1,k_2,k_3}, which means that each edge ww' in this graph is weighted according to the transition probability from one walk to the other whenever one of w, w' is in $W[k_1,k_2]$ and the other is in $W[k_2+1,k_3]$.

Claim 3.7.20.

$$\left(\mathsf{S}_{k_1,k_2,k_3}\right)^\dagger = \mathfrak{S}_{k_2,k_3,k_1}.$$

Proof. Let $f \in C^{W[k_1,k_2]}$ and $g \in C^{W[k_2+1,k_3]}$. For $i \le j$, define $\Pi_{i,j}$ to be the uniform distribution on $W[i,j]$. We show that $\left\langle f, \mathsf{S}_{k_1,k_2,k_3} g \right\rangle = \left\langle \mathfrak{S}_{k_2,k_3,k_1} f, g \right\rangle$. On one hand we have

$$\begin{aligned}
\left\langle f, \mathsf{S}_{k_1,k_2,k_3} g \right\rangle &= \mathbb{E}_{w \in W[k_1,k_2]} \left[f(w) \mathbb{E}_{w' : ww' \in W[k_1,k_3]} [g(w')] \right] \\
&= \mathbb{E}_{w \in W[k_1,k_2]} \left[f(w) \sum_{w' \in W[k_2+1,k_3]} \frac{\Pi_{k_1,k_3}(ww')}{\Pi_{k_1,k_2}(w)} g(w') \right] \\
&= \sum_{w \in W[k_1,k_2]} \Pi_{k_1,k_2}(w) f(w) \sum_{w' \in W[k_2+1,k_3]} \frac{\Pi_{k_1,k_3}(ww')}{\Pi_{k_1,k_2}(w)} g(w') \\
&= \sum_{ww' \in W[k_1,k_3]} f(w) g(w') \Pi_{k_1,k_3}(ww').
\end{aligned}$$

On the other hand we have

$$\begin{aligned}
\left\langle \mathfrak{S}_{k_2,k_3,k_1} f, g \right\rangle &= \mathbb{E}_{w' \in W[k_2+1,k_3]} \left[\mathbb{E}_{w : ww' \in W[k_1,k_3]} [f(w)] g(w') \right] \\
&= \mathbb{E}_{w' \in W[k_2+1,k_3]} \left[\sum_{w \in W[k_1,k_2]} \frac{\Pi_{k_1,k_3}(ww')}{\Pi_{k_2+1,k_3}(w')} f(w) g(w') \right] \\
&= \sum_{w' \in W[k_2+1,k_3]} \Pi_{k_2+1,k_3}(w') \sum_{w \in W[k_1,k_2]} \frac{\Pi_{k_1,k_3}(ww')}{\Pi_{k_2+1,k_3}(w')} f(w) g(w') \\
&= \sum_{ww' \in W[k_1,k_3]} f(w) g(w') \Pi_{k_1,k_3}(ww').
\end{aligned}$$

Hence, $\mathfrak{S}_{k_2,k_3,k_1} = (\mathsf{S}_{k_1,k_2,k_3})^\dagger$ as claimed. ∎

Variables for Walks on the s-wide Replacement Product

When analyzing walks on the s-wide replacement product, we actually need to use two separate, but related, local PSD ensembles. In Ta-Shma's construction, the vertices of the outer graph G correspond to positions in the base code $\mathcal{C}_0 \subseteq \mathbb{F}_2^n$, where $n = |V(G)|$. Given a vertex $(v, h) \in V(G) \times V(H)$ in the s-wide replacement product and codeword $z \in \mathcal{C}_0$, (v, h) is assigned bit z_v, regardless of the vertex h of the inner graph. We will enforce this property by working with variables in $V(G)$ rather than the full $V(G) \times V(H)$. The local PSD ensemble $\mathbf{Z} = \{\mathbf{Z}_v\}_{v \in V(G)}$ contains one variable for every vertex of G, with local distributions for sets of variables up to a given size. For a walk w on the s-wide replacement product, we will use $\mathbf{Z}_w$ as an abbreviation for $\mathbf{Z}_{S_w}$, where S_w is the set of all G-components of vertices visited on the walk.

The constraints of the CSP are placed on walks on the s-wide replacement product that do care about the H-component of the vertices, so we define a second local PSD ensemble $\mathbf{Y} = \{\mathbf{Y}_{(v,h)}\}_{(v,h) \in V(G) \times V(H)}$ with a variable for each vertex of the s-wide replacement product of G and H. It is this collection $\mathbf{Y}$ for which we need to prove tensoriality in order to use the list decoding framework. When we perform propagation rounding, we condition the ensemble $\mathbf{Z}$ on a random assignment σ to a subset $S \subseteq V(G)$, rather than conditioning $\mathbf{Y}$ on a random assignment to a subset of $V(G) \times V(H)$. Working with $\mathbf{Z}$ ensures that the rounded assignments will be consistent on each cloud of the s-wide replacement product. Since the bit assigned to a vertex (v, h) only depends on v, independent rounding of $\{\mathbf{Z} \mid \mathbf{Z}_S = \sigma\}$ will also yield the desired rounding of $\{\mathbf{Y} \mid \mathbf{Z}_S = \sigma\}$.

We can define $\mathbf{Y}$ based on the ensemble $\mathbf{Z}$ more concretely. Suppose $S' \subseteq V(G) \times V(H)$ is a subset of size at most p, where p is the locality of the ensemble, and define $T = \{v \mid (v, h) \in S'\}$. The distribution $\mu_{S'}$ of $\mathbf{Y}_{S'}$ is defined based on the distribution μ_T of $\mathbf{Z}_T$ by $\mu_{S'}(\alpha) = \mu_T(\alpha|_T)$, where $\alpha \in \mathbb{F}_2^{S'}$ is an assignment to S' whose value on each vertex (v, h) only depends on v.

Observe that the introduction of the ensemble $\mathbf{Y}$ is only necessary on the first level of the Ta-Shma code cascade between the codes $\mathcal{C}_0$ and $\mathcal{C}_1$, which takes place on the s-wide replacement product. Higher levels of the cascade use walks on graphs whose vertices are the walks from the level below. The association of the bits of a codeword to the vertices of this graph has no consistency requirement, so we simply use a single local ensemble $\mathbf{Z}$ with a variable for each vertex.

3.7.4 Splittability Implies Tensoriality

The connection between splittability and tensoriality will be made with the help of a version of the triangle inequality.

Claim 3.7.21 (Triangle inequality, adapted from [AJQ+20])**.** *Let $s \in \mathbb{N}^+$ and $\mathcal{T}$ be an s-interval splitting tree. Then*

$$\mathop{\mathbb{E}}_{w \in W[0,s-1]} \left\| \{\mathbf{Z}_w\} - \prod_{i=0}^{s-1} \left\{\mathbf{Z}_{w(i)}\right\} \right\|_1 \leq \sum_{(k_1,k_2,k_3) \in \mathcal{T}} \mathop{\mathbb{E}}_{w \in W[k_1,k_3]} \left\| \{\mathbf{Z}_w\} - \left\{\mathbf{Z}_{w(k_1,k_2)}\right\}\left\{\mathbf{Z}_{w(k_2+1,k_3)}\right\} \right\|_1,$$

where the sum is taken over the labels of the internal nodes of $\mathcal{T}$.

To prove tensoriality, we will use the method of [BRS11] and [AJT19] to show that we can break correlations over expanding collections of tuples arising in the s-wide replacement product of the form

$$\mathop{\mathbb{E}}_{\substack{ww' \in W[k_1,k_3] \\ w \in W[k_1,k_2], w' \in W[k_2+1,k_3]}} \left\| \{\mathbf{Z}_{ww'}\} - \{\mathbf{Z}_w\}\{\mathbf{Z}_{w'}\} \right\|_1$$

appearing on the right-hand side of the triangle inequality.

The First Level of the Cascade

We now check the technical details to obtain tensoriality for the first lifting in the code cascade between the codes $\mathcal{C}_0$ and $\mathcal{C}_1$, which corresponds to taking s steps in Ta-Shma's construction. Recall that in order to obtain an assignment $z' \in \mathbb{F}_2^n$ whose lifting is consistent on vertices with the same G-component, we need to prove tensoriality for the ensemble $\mathbf{Y}$ with a variable for each vertex in $V(G) \times V(H)$.

The proof of tensoriality will make use of a specific entropic potential function. For an arbitrary random variable $\mathbf{X}$ taking values in a finite set $[q]$, define the function $\mathcal{H}(\mathbf{X})$ as

$$\mathcal{H}(\mathbf{X}) \;\coloneqq\; \frac{1}{q} \sum_{a \in [q]} \mathrm{H}(\mathbb{1}_{[\mathbf{X}=a]}) \;=\; \mathbb{E}_{a \in [q]} \mathrm{H}(\mathbb{1}_{[\mathbf{X}=a]}),$$

where H is the binary entropy function. Using this, we define a potential function for a weighted undirected graph G.

Definition 3.7.22 (Graph Potential)**.** *Let $G = (V, E)$ be a weighted graph with edge distribution Π_E. Let Π_V be the marginal distribution on V. Suppose that $\{\mathbf{Y}_i\}_{i \in V}$ is a p-local PSD ensemble for some $p \geq 1$. We define Φ^G to be*

$$\Phi^G \;\coloneqq\; \mathop{\mathbb{E}}_{i \sim \Pi_V} \left[\mathcal{H}(\mathbf{Y}_i)\right].$$

Let $\mathcal{T}$ be an s-interval splitting tree associated with the s-wide replacement product of graphs G and H. We define

$$\Phi^{\mathcal{T}} \coloneqq \sum_{(k_1,k_2,k_3) \in \mathcal{T}} \Phi^{\mathcal{U}(\mathsf{S}_{k_1,k_2,k_3})},$$

where $\mathcal{U}(\mathsf{S}_{k_1,k_2,k_3})$ is the associated bipartite undirected graph of the operator S_{k_1,k_2,k_3}.

Lemma 3.7.23 (Splittability Implies Tensoriality)**.** *Let $W[0, s-1]$ be the walk collection*

of the s-wide replacement product of two graphs G and H. If $L \geq 128 \cdot (s^4 \cdot 2^{4s}/\mu^4)$ and $W[0, s-1]$ is τ-splittable with $\tau \leq \mu/(4s \cdot 2^{4s})$, then $W[0, s-1]$ is (μ, L)-tensorial.

Proof. We need to show that

$$\mathop{\mathbb{E}}_{w \in W[0,s-1]} \left\| \{\mathbf{Y}'_w\} - \prod_{i=0}^{s-1} \left\{\mathbf{Y}'_{w(i)}\right\} \right\|_1 \leq \mu,$$

which can be proven by adapting a potential argument technique from [BRS11]. First, set the potential

$$\Phi_m = \mathop{\mathbb{E}}_{S \sim \Pi_m} \mathop{\mathbb{E}}_{\sigma \sim \{\mathbf{Z}_S\}} \Phi^{\mathcal{T}}_{|\mathbf{Z}_S = \sigma}, \tag{3.5}$$

where the distribution Π_m on $S \subseteq V(G)$ is obtained from the process of choosing S in propagation rounding (Algorithm 3.7.16) once m has been fixed. Consider the error term

$$\mu_m \coloneqq \mathop{\mathbb{E}}_{S \sim \Pi_m} \mathop{\mathbb{E}}_{\sigma \sim \{\mathbf{Z}_S\}} D(S, \sigma), \tag{3.6}$$

where $D(S, \sigma) \coloneqq \mathbb{E}_{w \in W[0,s-1]} \left\| \{\mathbf{Y}_w \mid \mathbf{Z}_S = \sigma\} - \prod_{i=0}^{s-1} \left\{\mathbf{Y}_{w(i)} \mid \mathbf{Z}_S = \sigma\right\} \right\|_1$. If $\mu_m \geq \mu/2$, then

$$\mathop{\mathbb{P}}_{S \sim \Pi_m, \sigma \sim \{\mathbf{Z}_S\}} [D(S, \sigma) \geq \mu_m/2] \geq \frac{\mu}{4}.$$

For each choice of S and σ such that $D(S, \sigma) \geq \mu/2$, applying the triangle inequality from Claim 3.7.21 to the conditioned variables gives us

$$\begin{aligned} \frac{\mu}{2} &\leq \mathop{\mathbb{E}}_{w \in W[0,s-1]} \left\| \{\mathbf{Y}_w \mid \mathbf{Z}_S = \sigma\} - \prod_{i=0}^{s-1} \left\{\mathbf{Y}_{w(i)} \mid \mathbf{Z}_S = \sigma\right\} \right\|_1 \\ &\leq \sum_{(k_1,k_2,k_3) \in \mathcal{T}} \mathop{\mathbb{E}}_{w \in W[k_1,k_3]} \left\| \{\mathbf{Y}_w \mid \mathbf{Z}_S = \sigma\} - \left\{\mathbf{Y}_{w(k_1,k_2)} \mid \mathbf{Z}_S = \sigma\right\} \left\{\mathbf{Y}_{w(k_2+1,k_3)} \mid \mathbf{Z}_S = \sigma\right\} \right\|_1. \end{aligned}$$

Hence, there exists (k_1, k_2, k_3) such that

$$\frac{\mu}{2s} \leq \mathop{\mathbb{E}}_{w \in W[k_1,k_3]} \Big\| \{\mathbf{Y}_w \mid \mathbf{Z}_S = \sigma\} - \Big\{\mathbf{Y}_{w(k_1,k_2)} \mid \mathbf{Z}_S = \sigma\Big\}\Big\{\mathbf{Y}_{w(k_2+1,k_3)} \mid \mathbf{Z}_S = \sigma\Big\}\Big\|_1.$$

Note that choosing $w \in W[0, s-1]$ uniformly and restricting to $w(k_1, k_3)$ gives a uniformly random element of $W[k_1, k_3]$. If we choose $w(k_1, k_2)$ or $w(k_2 + 1, k_3)$ with equal probability, then the final walk is distributed according to the stationary measure of $\mathcal{U}(\mathsf{S}_{k_1,k_2,k_3})$. Let w' denote the chosen walk. Observe that $\mathbf{Y}_{w'}$ is a deterministic function of $\mathbf{Z}_{w'} \mid \mathbf{Z}_S = \sigma$. Now, we sample $\mathbf{Z}_{w'} \mid \mathbf{Z}_S = \sigma$, which gives us a sample of $\mathbf{Y}_{w'}$. Applying Lemma 3.10.1, we have

$$\Phi^{\mathcal{U}(\mathsf{S}_{k_1,k_2,k_3})}_{|\{\mathbf{Y}_{w'}|\mathbf{Z}_S=\sigma\}} \leq \Phi^{\mathcal{U}(\mathsf{S}_{k_1,k_2,k_3})}_{\mathbf{Z}_S=\sigma} - \frac{\mu^2}{16s^2 \cdot 2^{4s}}.$$

This conditioning on an assignment to $\mathbf{Z}_{\mathrm{set}(w')} \mid \mathbf{Z}_S = \sigma$ does not increase the other terms of $\Phi^{\mathcal{T}}$ associated to split operators other than $\mathcal{U}(\mathsf{S}_{k_1,k_2,k_3})$ since entropy is non-increasing under conditioning. Similarly, conditioning on the remaining variables that are part of w but not w' does not increase $\Phi^{\mathcal{T}}$. Then, we obtain

$$\Phi_m - \Phi_{m+1} \geq \mathop{\mathbb{P}}_{S \sim \Pi_m, \sigma \sim \{\mathbf{Z}_S\}} [D(S, \sigma) \geq \mu_m/2] \cdot \frac{\mu^2}{16s^2 \cdot 2^{4s}}.$$

Since $s \geq \Phi_1 \geq \cdots \geq \Phi_{L/(s+1)} \geq 0$, there can be at most $32s^3 \cdot 2^{4s}/\mu^3$ indices $m \in [L/s]$ such that $\mu_m \geq \mu/2$. In particular, since the total number of indices is L/s, we have

$$\mathop{\mathbb{E}}_{m \in [L/s]}[\mu_m] \leq \frac{\mu}{2} + \frac{s}{L} \cdot \frac{32s^3 \cdot 2^{4s}}{\mu^3}.$$

Our choice of L is more than enough to ensure $\mathbb{E}_{m \in [L/s]}[\mu_m] \leq \mu$. ∎

Applying the list decoding framework will require the stronger property of two-step tensoriality, which we can obtain under the same assumptions.

Lemma 3.7.24 (Splittability Implies Two-step Tensoriality)**.** *Let $W[0, s-1]$ be the walk collection of the s-wide replacement product of two graphs G and H. If $L \geq 128 \cdot (s^4 \cdot 2^{4s}/\mu^4)$ and $W[0, s-1]$ is τ-splittable with $\tau \leq \mu/(4s \cdot 2^{4s})$, then $W[0, s-1]$ is (μ, L)-two-step tensorial.*

Proof. Under our assumptions the (μ, L)-tensorial property follows from Lemma 3.7.23 (which is the only place where the assumption on τ is used), so we only need to show

$$\mathop{\mathbb{E}}_{w,w' \in W[0,s-1]} \left\| \{\mathbf{Y}'_w \mathbf{Y}'_{w'}\} - \left\{\mathbf{Y}'_w\right\}\left\{\mathbf{Y}'_{w'}\right\} \right\|_1 \leq \mu,$$

which can be proven by adapting a potential argument technique from [BRS11]. First, set the potential

$$\Phi_m = \mathop{\mathbb{E}}_{S \sim \Pi_m} \mathop{\mathbb{E}}_{\sigma \sim \{\mathbf{Z}_S\}} \mathop{\mathbb{E}}_{w \in W[0,s-1]} \mathcal{H}(\mathbf{Y}_w \mid \mathbf{Z}_S = \sigma), \tag{3.7}$$

where the distribution Π_m on $S \subseteq V(G)$ is obtained from the process of choosing S in propagation rounding (Algorithm 3.7.16) once m has been fixed. Consider the error term

$$\mu_m \coloneqq \mathop{\mathbb{E}}_{S \sim \Pi_m} \mathop{\mathbb{E}}_{\sigma \sim \{\mathbf{Z}_S\}} D(S, \sigma), \tag{3.8}$$

where $D(S, \sigma) \coloneqq \mathbb{E}_{w,w' \in W[0,s-1]}[\|\{\mathbf{Y}_w \mathbf{Y}_{w'} \mid \mathbf{Z}_S = \sigma\} - \{\mathbf{Y}_w | \mathbf{Z}_S = \sigma\}\{\mathbf{Y}_{w'} | \mathbf{Z}_S = \sigma\}\|_1]$. If $\mu_m \geq \mu/2$, then

$$\mathop{\mathbb{P}}_{S \sim \Pi_m, \sigma \sim \{\mathbf{Z}_S\}} [D(S, \sigma) \geq \mu_m/2] \geq \frac{\mu}{4}.$$

Let $G' = (V = W[0, s-1], E)$ be the graph with edges $E = \{\{w, w'\} \mid w, w' \in W[0, s-1]\}$. Local correlation (expectation over the edges) on this graph G' is the same as global correlation (expectation over two independent copies of vertices). Then, we obtain[7]

$$\Phi_m - \Phi_{m+1} \geq \mathop{\mathbb{P}}_{S \sim \Pi_m, \sigma \sim \{\mathbf{Z}_S\}} [D(S, \sigma) \geq \mu_m/2] \cdot \frac{\mu^2}{2 \cdot 2^{2s}}.$$

7. See [AJT19] or [BRS11] for the details.

Since $1 \geq \Phi_1 \geq \cdots \geq \Phi_{L/(s+1)} \geq 0$, there can be at most $8 \cdot 2^{2s}/\mu^3$ indices $m \in [L/s]$ such that $\mu_m \geq \mu/2$. In particular, since the total number of indices is L/s, we have

$$\mathop{\mathbb{E}}_{m \in [L/s]} \mu_m \leq \frac{\mu}{2} + \frac{k}{L} \cdot \frac{8 \cdot 2^{2s}}{\mu^3}.$$

Our choice of L is more than enough to ensure $\mathbb{E}_{m \in [L/s]}[\mu_m] \leq \mu$. ∎

We have already established that $W[0, s-1]$ is τ-splittable with $\tau = \sigma_2(G) + 2\sigma_2(H) + \sigma_2(H)^2$ in Corollary 3.7.13, so we can obtain (μ, L)-two-step tensoriality for any μ if this quantity is small enough.

Higher Levels of the Cascade

We now discuss tensoriality of the other levels of the cascade between $\mathcal{C}_{i-1}$ and $\mathcal{C}_i$ for $i \geq 2$. Tensorial properties are simpler to establish here than on the first level of the cascade. The relevant split operators are special cases of S_{k_1,k_2,k_3} where $k_1 \equiv 0 \pmod{s}$ and $k_2, k_3 \equiv -1 \pmod{s}$. The main difference now is that we can associate the parity bits of $\mathcal{C}_{i-1}$ with the vertices of $\mathcal{U}(\mathsf{S}^{\triangle}_{r,r})$, which themselves represent walks. As this association of parity bits doesn't need to satisfy a consistency condition, we only need to work with a single ensemble $\mathbf{Z}$ instead of working with two different ensembles as in the previous case. The proofs of Lemma 3.7.23 and Lemma 3.7.24 with these slight modifications give us two-step tensoriality.

Lemma 3.7.25 (Two-step Tensoriality for Higher Levels)**.** *Let $W(k)$ be the set of walks defined using $(k-1)$ steps of the operator $\mathsf{S}^{\triangle}_{r,r}$. If $L \geq 128 \cdot (k^4 \cdot 2^{4k}/\mu^4)$ and $W(k)$ is τ-splittable with $\tau \leq \mu/(4k \cdot 2^{4k})$, then $W(k)$ is (μ, L)-two-step tensorial.*

We know from Corollary 3.7.13 that the collection of walks obtained from $\sigma_2(\mathsf{S}^{\triangle}_{r,r})$ is $(\sigma_2(G) + 2 \cdot \sigma_2(H) + \sigma_2(H)^2)$-splittable, so the parameters necessary to obtain two-step tensoriality are the same as in the first level of the cascade.

3.8 Choosing Parameters for Ta-Shma's Construction

We explore how some choices of parameters for Ta-Shma's construction interact with the requirements of our decoding algorithm. The analysis is divided into rounds of increasingly stronger decoding guarantees with later rounds relying on the codes obtained in previous rounds. Naturally, the stronger guarantees come with more delicate and technical considerations. We briefly summarize the goals of each round and some key parameters.

1. Round I: For any constant $\beta > 0$, we obtain *efficient unique decodable* codes $\mathcal{C}_\ell$ with distance at least $1/2 - \varepsilon$ and rate $\Omega(\varepsilon^{2+\beta})$ for infinitely many *discrete* values of $\varepsilon > 0$ (with ε as close to 0 as desired). In this regime, it suffices for the expansion of H to be constant. This round leads to Theorem 3.6.6.

2. Round II: Similar to Round I, but now ε can be any value in an interval $(0, b)$ with $b < 1/2$ being a function of β. Again the expansion of H can be constant. This round leads to Theorem 3.6.7.

3. Round III: We want β to vanish as ε vanishes (this is qualitatively similar to Ta-Shma's result). In this regime, we make the expansion of H be a function of ε, and we rely on the uniquely decodable codes of Round II. This round leads to Theorem 3.1.1.

4. Round IV: For any constant $\beta_0 > 0$, we obtain *efficient list decodable* codes $\mathcal{C}_\ell$ with list decoding radius $1/2 - \beta_0$ and rate $\Omega(\varepsilon^{2+\beta})$ with $\beta \to 0$ as $\varepsilon \to 0$. In this regime, we make the expansion of H be a function of ε, and we rely on the uniquely decodable codes of Round III. This round leads to Theorem 3.1.2.

The way we choose parameters for Ta-Shma's construction borrows heavily from Ta-Shma's arguments in [TS17]. We fix some notation common to all rounds. A graph is said to be an (n, d, λ)-graph provided it has n vertices, is d-regular, and has second largest singular value of its normalized adjacency matrix at most λ.

Notation 3.8.1. *We use the following notation for the graphs G and H used in the s-wide replacement product.*

- *The outer graph G will be an (n', d_1, λ_1)-graph.*
- *The inner graph H will be a (d_1^s, d_2, λ_2)-graph.*

The parameters $n', d_1, d_2, \lambda_1, \lambda_2$ and s will be chosen in the subsequent sections.

3.8.1 Round I: Initial Analysis

We are given the dimension D of the desired code and $\varepsilon \in (0, 1/2)$. We set a parameter $\alpha \leq 1/128$ such that (for convenience) $1/\alpha$ is a power of 2 and

$$\frac{\alpha^5}{4\log_2(1/\alpha)} \geq \frac{1}{\log_2(1/\varepsilon)}. \tag{3.9}$$

We can assume that $\alpha \leq 1/128$ satisfy Eq. (3.9) since otherwise ε is a constant and we can use the list decodable codes from [AJQ+20]. The use of Eq. (3.9) will be clear shortly. It becomes a necessity from round III onward. For rounds I and II, the parameter α will be a constant, but it will be useful to establish the analysis in more generality now so that subsequent rounds can reuse it.

The inner graph H. The choice of H is similar to Ta-Shma's choice. More precisely, we set $s = 1/\alpha$ and $d_2 = s^{4s^2}$ (Ta-Shma took $d_2 = s^{4s}$). We obtain a Cayley graph $H = \mathrm{Cay}(\mathbb{F}_2^{4s\log_2(d_2)}, A)$ such that H is an $(n_2 = d_2^{4s}, d_2, \lambda_2)$ graph where $\lambda_2 = b_2/\sqrt{d_2}$ and $b_2 = 4s\log_2(d_2)$. (The set of generators, A, comes from a small bias code derived from a construction of Alon et al. [AGHP92], but we will rely on Ta-Shma's analysis embodied in Lemma 3.10.11 and not discuss it further.)

The base code $\mathcal{C}_0$. Set $\varepsilon_0 = 1/d_2^2 = \lambda_2^4/b_2^4 \leq \lambda_2^4/3$ (this choice differs from Ta-Shma's and it appears because we are essentially working with H^2 rather than H). We will choose a base code $\mathcal{C}_0$ such that the desired code will be obtained as a direct sum lifting of $\mathcal{C}_0$, and

because this lifting preserves the dimension, the dimension of $\mathcal{C}_0$ should be D. We choose $\mathcal{C}_0$ to be an ε_0-balanced code with dimension D and block length $n = O_{\varepsilon_0}(D)$. For instance, we can start with any good (constant rate and relative distance) linear base code $\mathcal{C}_0$ that has an efficient unique decoding algorithm and obtain a ε_0-balanced lifted code that can be efficiently unique decoded (as long as ε_0 is constant) using the framework in [AJQ+20].

The outer graph G. Set $d_1 = d_2^4$ so that $n_2 = d_1^s$ as required by the s-wide replacement product. We apply Ta-Shma's explicit Ramanujan graph Lemma 3.10.10 with parameters n, d_1 and θ to obtain an (n', d_1, λ_1) Ramanujan graph G with $\lambda_1 \leq 2\sqrt{2}/\sqrt{d_1}$ and $n' \in [(1-\theta)n, n]$ or $n' \in [(1-\theta)2n, 2n]$. Here, θ is an error parameter that we set as $\theta = \lambda_2^4/6$ (this choice of θ differs from Ta-Shma). Because we can construct words with block length $2n$ (if needed) by duplicating each codeword, we may assume w.l.o.g. that n' is close to n and $(n - n') \leq \theta n \leq 2\theta n'$. See Section 3.10.2 for a more formal description of this graph.

Note that $\lambda_1 \leq \lambda_2^4/6$ since $\lambda_1 \leq 3/\sqrt{d_1} = 3/d_2^2 = 3 \cdot \lambda_2^4/b_2^4 \leq \lambda_2^4/6$. Hence, $\varepsilon_0 + 2\theta + 2\lambda_1 \leq \lambda_2^4$.

The walk length. Set the walk length $t-1$ to be the smallest integer such that

$$(\lambda_2^2)^{(1-5\alpha)(1-\alpha)(t-1)} \leq \varepsilon.$$

This will imply using Ta-Shma's analysis that the bias of the final code is at most ε as shown later.

$s = 1/\alpha$, such that $\frac{\alpha^5}{4\log_2(1/\alpha)} \geq \frac{1}{\log_2(1/\varepsilon)}$
$H : (n_2, d_2, \lambda_2), \quad n_2 = d_1^s, \quad d_2 = s^{4s^2}, \quad \lambda_2 = \frac{b_2}{\sqrt{d_2}}, \quad b_2 = 4s \log d_2$
$G : (n', d_1, \lambda_1), \quad n' \approx n = O(D/\varepsilon_0^c), \quad d_1 = d_2^4, \quad \lambda_1 \leq \frac{2\sqrt{2}}{d_1}$
$t :$ smallest integer such that $(\lambda_2^2)^{(1-5\alpha)(1-\alpha)(t-1)} \leq \varepsilon$

Claim 3.8.2. *We have $t - 1 \geq s/\alpha = s^2$.*

Proof. Using $d_2 = s^{4s^2}$ and Eq. (3.9), we have

$$\begin{aligned}\left(\frac{1}{\lambda_2^2}\right)^{(1-5\alpha)(1-\alpha)s/\alpha} &\leq \left(\frac{1}{\lambda_2^2}\right)^{s/\alpha} = \left(\frac{d_2}{b_2^2}\right)^{s/\alpha} \leq (d_2)^{s/\alpha} = s^{4s^3/\alpha} \\ &= 2^{4s^3\log_2(s)/\alpha} = 2^{4\log_2(1/\alpha)/\alpha^4} \leq 2^{\log_2(1/\varepsilon)} = \frac{1}{\varepsilon}.\end{aligned}$$

Hence, $\varepsilon \leq (\lambda_2^2)^{(1-5\alpha)(1-\alpha)s/\alpha}$ and thus $t-1$ must be at least s/α. ∎

Remark 3.8.3. *By our choice of t, we have $(\lambda_2^2)^{(1-5\alpha)(1-\alpha)(t-2)} \geq \varepsilon$. Since $1/(t-1) \leq \alpha$, we get $(\lambda_2^2)^{(1-5\alpha)(1-\alpha)^2(t-1)} \geq \varepsilon$.*

Final Bias. We denote by $\mathcal{C}_\ell$ the final code obtained by t steps of the s-wide replacement product. The bias of $\mathcal{C}_\ell$ is given by Corollary 3.4.10 (which in turn is a simple corollary of Ta-Shma's Fact 3.4.9) as shown next.

Corollary 3.8.4. *The code $\mathcal{C}_\ell$ is ε-balanced.*

Proof. Using Corollary 3.4.10, we have that the final bias

$$b \coloneqq \left(\sigma_2(H^2)^{s-1} + (s-1)\cdot\sigma_2(H^2)^{s-2} + (s-1)^2\cdot\sigma_2(H^2)^{s-4}\right)^{\lfloor(t-1)/s\rfloor}$$

is bounded by

$$\begin{aligned}b &\leq (3(s-1)^2\sigma_2(H^2)^{s-4})^{((t-1)/s)-1} && \text{(Using } \sigma_2(H^2) \leq 1/3s^2\text{)} \\ &\leq ((\sigma_2(H^2)^{s-5})^{(t-1-s)/s} \\ &= \sigma_2(H^2)^{(1-5/s)(1-s/(t-1))(t-1)} \\ &\leq \sigma_2(H^2)^{(1-5\alpha)(1-\alpha)(t-1)} \\ &= \left(\lambda_2^2\right)^{(1-5\alpha)(1-\alpha)(t-1)} \leq \varepsilon,\end{aligned}$$

where the last inequality follows from $s = 1/\alpha$ and $t-1 \geq s/\alpha$, the latter from Claim 3.8.2.

∎

Rate. The proof of the rate follows a similar structure of Ta-Shma's original argument except that we take s to be a constant independent of ε so that ε_0, λ_1, and λ_2 are also constants independent of ε. Note that we previously said $\alpha = 1/s$ needs to satisfy Equation 3.9, but that implies only an upper bound for s, and smaller (even constant) values for s are still permissible.

Claim 3.8.5. *$\mathcal{C}_\ell$ has rate $\Omega(\varepsilon^{2+26\cdot\alpha})$ provided $\varepsilon_0 > 0$ is constant.*

Proof. The support size is the number of walks of length t on the s-wide replacement product of G and H (each step of the walk has d_2^2 options), which is

$$\begin{aligned}
|V(G)||V(H)|d_2^{2(t-1)} = n' \cdot d_1^s \cdot d_2^{2(t-1)} &= n' \cdot d_2^{2(t-1)+4s} \leq n \cdot d_2^{2(t-1)+4s} \\
&= \Theta_{\varepsilon_0}\left(D \cdot d_2^{2(t-1)+4s}\right) \\
&= \Theta\left(D \cdot (d_2^2)^{t-1+2s}\right) \\
&= O\left(D \cdot (d_2^2)^{(1+2\alpha)(t-1)}\right),
\end{aligned}$$

where the penultimate equality follows from the assumption that ε_0 is a constant.

Note that $d_2^\alpha = d_2^{1/s} = s^{4s} \geq b_2$ since $b_2 = 4s\log_2(d_2) = 16s^3\log_2(s) \leq s^4$ (recall that $s = 1/\alpha \geq 128$). Thus,

$$d_2^{1-2\alpha} = \frac{d_2}{d_2^{2\alpha}} \leq \frac{d_2}{b_2^2} = \frac{1}{\sigma_2(H^2)}.$$

We obtain

$$\begin{aligned}
(d_2^2)^{(t-1)} &\leq \left(\frac{1}{\sigma_2(H^2)}\right)^{\frac{2(t-1)}{1-2\alpha}} \\
&\leq \left(\frac{1}{\varepsilon}\right)^{\frac{2}{(1-2\alpha)(1-5\alpha)(1-\alpha)^2}} && \text{(Using Remark 3.8.3)} \\
&\leq \left(\frac{1}{\varepsilon}\right)^{2(1+10\alpha)},
\end{aligned}$$

which implies a block length of

$$O\left(D \cdot (d_2^2)^{(1+2\alpha)(t-1)}\right) = O\left(D\left(\frac{1}{\varepsilon}\right)^{2(1+10\alpha)(1+2\alpha)}\right) = O\left(D\left(\frac{1}{\varepsilon}\right)^{2(1+13\alpha)}\right).$$

■

Lemma 3.8.6 (Codes Near the GV bound I)**.** *For every constant $\beta > 0$, there exists a sufficiently large constant s in the above analysis so that for any dimension value $D \in \mathbb{N}^+$ (sufficiently large) and $\varepsilon > 0$ (sufficiently small) the final code $\mathcal{C}_{N,\varepsilon,\beta}$, where N is the block length, satisfies*

- *$\mathcal{C}_{N,\varepsilon,\beta}$ is ε-balanced,*
- *$\mathcal{C}_{N,\varepsilon,\beta}$ has rate $\Omega(\varepsilon^{2+\beta})$, and*
- *$\mathcal{C}_{N,\varepsilon,\beta}$ is a linear code of dimension D.*

Remark 3.8.7. *As a consequence of code cascading, the final attainable walk lengths have the form $s^\ell - 1$ where ℓ is a positive integer. Given $\beta > 0$, we have infinitely many values of ε attainable by such walk lengths which gives infinitely many codes $\mathcal{C}_{N,\varepsilon,\beta}$. This means that although the bias ε cannot be arbitrary, we have an infinite sequence of values of ε for which the rates of the codes $\mathcal{C}_{N,\varepsilon,\beta}$ are near the GV bound. In Section 3.8.2, we show how to bypass this artificial limitation. These codes are used in the proof of Theorem 3.6.6.*

We can view the above analysis as defining a function Γ that receives

- the dimension $D \in \mathbb{N}^+$,
- the final bias $\varepsilon > 0$,
- the approximating error $\alpha \in (0, 1/128]$ with $s \coloneqq 1/\alpha$ being a power of two, and
- a multiplying factor $Q \in \mathbb{N}^+$ such that $d_2 = s^{4s^2 \cdot Q}$ (in the above Q was 1).

and outputs a tuple of parameters $(t, \varepsilon_0, \theta, d_1, \lambda_1, n')$, graphs G and H (as above) where, in particular, the number of steps $t \in \mathbb{N}^+$ is such that the final code $\mathcal{C}_\ell$ has bias at most ε and rate $\Omega(\varepsilon^{2+26\cdot\alpha})$.

In future rounds, Γ may be called with $Q = s$ instead of $Q = 1$. This will cause d_2 to increase from s^{4s^2} to $s^{4s^2\cdot Q}$, and so in the proof of Claim 3.8.2, $2^{4\log_2(1/\alpha)/\alpha^4}$ will be replaced by $2^{4\log_2(1/\alpha)/\alpha^5}$. This explains why Eq. (3.9) has a stricter requirement than needed in the $Q = 1$ case above.

3.8.2 Round II: A More Careful Analysis

We are given the dimension of the code D and $\varepsilon \in (0, 1/2)$. As before, we set a parameter $\alpha \leq 1/128$ such that (for convenience) $1/\alpha$ is a power of 2. Set $s = 1/\alpha$ and $Q = s$.

Apply Γ to $(D, \varepsilon, \alpha, Q)$ to obtain all parameters except t. Choose t to be the smallest integer satisfying

$$(\lambda_2^2)^{(1-5\alpha)(1-2\alpha)(1-\alpha)(t-1)} \leq \varepsilon,$$

where observe that an extra $(1 - 2\alpha)$ factor appears in the exponent. This change in t will worsen the rate but by losing a factor of $\frac{1}{1-2\alpha}$ in the exponent, we can lower bound the rate. That is, $(d_2^2)^{-(t-1)} = \Omega(\varepsilon^{\frac{2+26\cdot\alpha}{1-2\alpha}})$.

Set $\ell \in \mathbb{N}^+$ to be the smallest value such that $s^\ell \geq t$ (here we are implicitly assuming that $t > s$). If $s^\ell = t$, we are done since we can use all the parameters returned by Γ for the construction of $\mathcal{C}_\ell$. Now assume $s^\ell > t$ and let $\zeta = t/s^{\ell-1}$. Note that $\zeta \in (1, s)$. Choose P to be the integer in the interval $[Q, s \cdot Q]$ such that

$$0 \leq \frac{P}{Q} - \zeta \leq \frac{1}{Q}.$$

Because $s^\ell > t$, and only powers of s may be chosen for walk length, we might overshoot in walk length by a multiplicative factor of s. This will cause a corresponding decay in

rate computation that we cannot afford. To overcome this, in the last level of the cascade between codes $\mathcal{C}_{\ell-1}$ and $\mathcal{C}_\ell$, perform the direct sum over walks of length $(P-1)$ instead of length $(s-1)$. The new total number of vertices is $t' = Ps^{\ell-1}$. Note that P can be as large as s^2, so our splittability guarantee of $W(P)$ (the walk collection from the lift between $\mathcal{C}_{\ell-1}$ and $\mathcal{C}_\ell$) has to be strong enough to accommodate this larger arity and not only arity s.

Claim 3.8.8. *We have* $t - 1 \leq \frac{t'-1}{Q} \leq (1+2\alpha)(t-1)$.

Proof. By construction, we have the sequence of implications

$$\begin{aligned} 0 \leq \frac{P}{Q}s^{\ell-1} - \zeta s^{\ell-1} &\leq \frac{s^{\ell-1}}{Q} \\ \Rightarrow 0 \leq \frac{t'}{Q} - t &\leq \frac{s^{\ell-1}}{Q} \leq \frac{t}{Q} \\ \Rightarrow t - \frac{1}{Q} &\leq \frac{t'-1}{Q} \leq (t-1)\left(1+\frac{1}{Q}\right) + 1, \end{aligned}$$

from which we obtain

$$t - 1 \leq t - \frac{1}{Q} \leq \frac{t'-1}{Q}$$

and

$$\frac{t'-1}{Q} \leq (t-1)\left(1+\frac{1}{Q}\right) + 1 = (1+\alpha)(t-1) + 1 < (1+2\alpha)(t-1),$$

the latter using $Q = s = 1/\alpha$. ∎

We apply Γ again but this time to $(D, \varepsilon, \alpha, 1)$ to obtain new parameters $(t'', \varepsilon_0', \theta', d_1', \lambda_1', n'')$, and graphs G' and H'.

Claim 3.8.9. *The code* $\mathcal{C}_\ell'$ *obtained by* t' *walk steps on the* s*-wide replacement product of* G' *and* H' *from the second application of* Γ *has bias at most* ε *and rate* $\Omega(\varepsilon^{2+40\alpha})$.

Proof. Let $d_2 = s^{4s^2 \cdot Q}$, $b_2 = 4s\log_2(d_2)$ and $\lambda_2 = b_2/\sqrt{d_2}$ be the parameters of the first

invocation of Γ. Recall that t was chosen to be the smallest integer satisfying

$$(\lambda_2^2)^{(1-5\alpha)^2(1-\alpha)(t-1)} \leq \varepsilon.$$

Let $d_2' = s^{4s^2}$, $b_2' = 4s\log_2(d_2')$ and $\lambda_2' = b_2'/\sqrt{d_2'}$ be the parameters of the second invocation of Γ. Observe that

$$\begin{aligned}(\lambda_2')^Q &= \frac{(b_2')^Q}{\sqrt{(d_2')^Q}} = \frac{(b_2'^Q)}{\sqrt{d_2}} = \frac{(16s^3\log_2(s))^Q}{s^{2s^2\cdot Q}} \\ &\leq \frac{s^{4Q}}{s^{2s^2\cdot Q}} = \frac{1}{s^{2s^2\cdot Q(1-\frac{2}{s^2})}} = \left(\frac{1}{s^{2s^2\cdot Q}}\right)^{1-2\alpha} \leq \left(\frac{b_2}{\sqrt{d_2}}\right)^{1-2\alpha} = \lambda_2^{1-2\alpha}.\end{aligned}$$

Then the bias of $\mathcal{C}_\ell'$ is at most

$$\begin{aligned}(((\lambda_2')^Q)^2)^{(1-5\alpha)(1-\alpha)(t'-1)/Q} &\leq (\lambda_2^2)^{(1-5\alpha)(1-2\alpha)(1-\alpha)(t'-1)/Q} \\ &\leq (\lambda_2^2)^{(1-5\alpha)(1-2\alpha)(1-\alpha)(t-1)} \leq \varepsilon.\end{aligned}$$

For the rate computation of $\mathcal{C}_\ell'$, we will lower bound the term $((d_2')^2)^{-(t'-1)}$. Since $d_2 = (d_2')^Q$, $(d_2^2)^{-(t-1)} = \Omega(\varepsilon^{\frac{2+26\cdot\alpha}{1-2\alpha}})$ and $\frac{t'-1}{Q} \leq (1+2\alpha)(t-1)$ (the latter by Claim 3.8.8), the rate of $\mathcal{C}_\ell'$ is

$$\Omega(((d_2')^2)^{-(t'-1)}) = \Omega((d_2^2)^{-(t'-1)/Q}) = \Omega((d_2^2)^{-(1+2\alpha)(t-1)}) = \Omega((\varepsilon^{2+26\cdot\alpha})^{\frac{1+2\alpha}{1-2\alpha}}) = \Omega(\varepsilon^{2+40\cdot\alpha}).$$

■

3.8.3 Round III: Vanishing β as ε Vanishes

We are given the dimension of the code D and $\varepsilon \in (0, 1/2)$. As before, we set a parameter $\alpha \leq 1/128$ such that (for convenience) $1/\alpha$ is a power of 2. Set $s \coloneqq 1/\alpha$.

We will consider the regime where s is a function of ε. As a consequence, the parameters

$d_2, \lambda_2, d_1, \lambda_1, \varepsilon_0$ will also depend on ε. Since $x \leq 1/\log_2(1/x)$ for $x \leq 1/2$ (and $\alpha \leq 1/2$), if α satisfies $\alpha^6/4 \geq 1/\log_2(1/\beta)$, it also satisfies Eq. (3.9) (we lose a log factor by replacing $1/\log_2(1/\alpha)$ by α, but we will favor simplicity of parameters). In particular, we can set α so that s is

$$s = \Theta((\log_2(1/\varepsilon))^{1/6}),$$

and satisfy Eq. (3.9).

We follow the same choices as in Round II except for the base code $\mathcal{C}_0$.

The base code $\mathcal{C}_0$. Set $\varepsilon_0 = 1/d_2^2 = \lambda_2^4/b_2^4 \leq \lambda_2^4/3$. We choose an ε_0-balanced code $\mathcal{C}_0$ with support size $n = O(D/\varepsilon_0^c)$ where $c = 2.001$ (this choice of c is arbitrary, it is enough to have c as a fixed small constant) using the construction from Round II. It is crucial that we can unique decode $\mathcal{C}_0$ (using our algorithm), since this is required in order to apply the list decoding framework.

Note that ε_0 is no longer a constant. For this reason, we need to consider the rate computation of the final code $\mathcal{C}_\ell$ more carefully. The proof will follow an argument similar to Ta-Shma's.

Claim 3.8.10. *$\mathcal{C}_\ell$ has rate $\Omega(\varepsilon^{2+26\cdot\alpha})$ where $\alpha = \Theta(1/(\log_2(1/\varepsilon))^{1/6})$.*

Proof. The support size is the number of walks of length $t-1$ on the s-wide replacement product of G and H (each step of the walk has d_2^2 options), which is

$$\begin{aligned}
|V(G)||V(H)|d_2^{2(t-1)} = n' \cdot d_1^s \cdot d_2^{2(t-1)} &= n' \cdot d_2^{2(t-1)+4s} \leq n \cdot d_2^{2(t-1)+4s} \\
&= \Theta\left(\frac{D}{\varepsilon_0^c} \cdot d_2^{2(t-1)+4s}\right) \\
&= \Theta\left(D \cdot (d_2^2)^{(t-1)+2s+2.001}\right) \\
&= O\left(D \cdot (d_2^2)^{(1+2\alpha)(t-1)}\right).
\end{aligned}$$

From this point the proof continues exactly as the proof of Claim 3.8.5. ∎

3.8.4 Round IV: Arbitrary Gentle List Decoding

In round III, when we take

$$s = \Theta((\log_2(1/\varepsilon))^{1/6}),$$

we will have $\lambda_2 = 4s\log(s^{4s^2})/s^{2s^2} \leq s^{-s^2}$ provided s is large enough. This non-constant λ_2 will allow us perform "gentle" list decoding with radius arbitrarily close to $1/2$. More precisely, we have the following.

Theorem 3.8.11 (Gentle List Decoding (restatement of Theorem 3.1.2))**.** *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 ε-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 $N^{O_{\varepsilon,\beta}(1)}$.

Proof. We consider some parameter requirements in order to apply the list decoding framework Theorem 3.9.1 between $\mathcal{C}_{\ell-1}$ and $\mathcal{C}_\ell$. Suppose we want to list decode within radius $1/2 - \sqrt{\eta}$. For parity sampling, we need

$$s \geq \Theta(\log_2(1/\eta)).$$

Since the number of vertices in a walk can be at most s^2, for splittability we need

$$\eta^8/(s^2 \cdot 2^{2s^2}) \geq 2 \cdot s^{-s^2}.$$

In particular, we can take $\eta = 2^{-\Theta(s)}$ and satisfy both conditions above. ∎

3.9 Instantiating the List Decoding Framework

We established the tensoriality (actually two-step tensoriality) and parity sampling properties of every lifting between consecutive codes $\mathcal{C}_{i-1}$ and $\mathcal{C}_i$ in Ta-Shma's cascade. Using these properties, we will be able to invoke the list decoding framework from [AJQ+20] to obtain the following list decoding result.

Theorem 3.9.1 (Restatement of Theorem 3.6.1)**.** *Let $\eta_0 \in (0, 1/4)$ be a constant, $\eta \in (0, \eta_0)$, and*

$$k \geq k_0(\eta) \coloneqq \Theta(\log(1/\eta)).$$

Suppose $\mathcal{C} \subseteq \mathbb{F}_2^n$ is an η_0-balanced linear code and $\mathcal{C}' = \mathrm{dsum}_{W(k)}(\mathcal{C})$ is the direct sum lifting of $\mathcal{C}$ on a τ-splittable collection of walks $W(k)$, where $W(k)$ is either the set of walks $W[0, s]$ on an s-wide replacement product graph or a set of walks using the random walk operator $\mathsf{S}^{\triangle}_{r,r}$. There exists an absolute constant $K > 0$ such that if

$$\tau \leq \tau_0(\eta, k) \coloneqq \frac{\eta^8}{K \cdot k \cdot 2^{4k}},$$

then the code $\mathcal{C}'$ is η-balanced and can be efficiently list decoded in the following sense:

If $\tilde{y}$ is $(1/2 - \sqrt{\eta})$-close to $\mathcal{C}'$, then we can compute the list

$$\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}') \coloneqq \left\{(z, \mathrm{dsum}_{W(k)}(z)) \mid z \in \mathcal{C}, \Delta\Big(\mathrm{dsum}_{W(k)}(z), \tilde{y}\Big) \leq \frac{1}{2} - \sqrt{\eta}\right\}$$

in time

$$n^{O(1/\tau_0(\eta,k)^4)} \cdot f(n),$$

where $f(n)$ is the running time of a unique decoding algorithm for $\mathcal{C}$. Otherwise, we return $\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}') = \emptyset$ with the same running time of the preceding case.[8]

8. In the case $\tilde{y}$ is not $(1/2 - \sqrt{\eta})$-close to $\mathcal{C}'$, but the SOS program turns out to be feasible, some of the calls to the unique decoding algorithm of $\mathcal{C}$ (issued by the list decoding framework) might be outside all

3.9.1 List Decoding Framework

We recall the precise statement of the list decoding framework tailored to direct sum lifting.

Theorem 3.9.2 (List Decoding Theorem (Adapted from [AJQ+20]))**.** *Suppose* $\mathrm{dsum}_{W(k)}$ *is an* $(\eta^8/2^{30}, L)$*-two-step tensorial direct sum lifting from an* η_0*-balanced code* $\mathcal{C} \subseteq \mathbb{F}_2^n$ *to* $\mathcal{C}'$ *on a multiset* $W(k) \subseteq [n]^k$ *which is a* $(1/2 + \eta_0/2, \eta)$*-parity sampler.*

Let $\tilde{y} \in \mathbb{F}_2^{W(k)}$ *be* $(1/2 - \sqrt{\eta})$*-close to* $\mathcal{C}'$*. Then the List Decoding algorithm returns the coupled code list* $\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}')$*. Furthermore, the running time is* $n^{O(L+k)}\,(\mathrm{polylog}(1/\eta) + f(n))$ *where* $f(n)$ *is the running time of an unique decoding algorithm of* $\mathcal{C}$*.*

We apply the list decoding framework of Theorem 3.9.2 to the liftings arising in the Ta-Shma cascade to obtain Theorem 3.9.1. This requires choosing parameters so that both the parity sampling and tensoriality requirements are met at every level of the cascade, which we do by appealing to our results from Section 3.7.

Proof of Theorem 3.9.1. We want to define parameters for τ-splittability so that $W(k)$ satisfies strong enough parity sampling and tensoriality assumptions to apply Theorem 3.9.2.

For parity sampling, we require $W(k)$ to be an $(1/2 + \eta_0/2, \eta)$-parity sampler. Suppose $W(k)$ is τ-splittable with $\tau < 1/16$. By Corollary 3.7.4 or Corollary 3.7.7 and splittability, the collection of walks $W(k)$ is an (η_0', η')-parity sampler, where $\eta' \leq (\eta_0' + 2\tau)^{\lfloor (k-1)/2 \rfloor}$. To achieve the desired parity sampling, we take $\eta_0' = 1/2 + \eta_0/2$ and choose a value of k large enough so that $\eta' \leq \eta$. Using the assumption $\eta_0 < 1/4$, we compute

$$\eta' = (\eta_0' + 2\tau)^{\lfloor (k-1)/2 \rfloor} \leq (1/2 + \eta_0/2 + 2\tau)^{k/2-1} < (3/4)^{k/2-1},$$

unique decoding balls. Such cases may be handled by returning failure if the algorithm does not terminate by the time $f(n)$. Even if a codeword in $\mathcal{C}$ is found, the pruning step of list decoding [AJQ+20] will return an empty list for $\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}')$ since $\tilde{y}$ is not $(1/2 - \sqrt{\eta})$-close to $\mathcal{C}$.

which will be smaller than η as long as k is at least

$$k_0(\eta) = 2\left(1 + \frac{\log(1/\eta)}{\log(4/3)}\right) = \Theta(\log(1/\eta)).$$

Achieving this level of parity sampling also ensures that the lifted code $\mathcal{C}'$ is η-balanced.

The list decoding theorem also requires $(\eta^8/2^{30}, L)$-two-step tensoriality. Lemma 3.7.24 (with $s = k$) and Lemma 3.7.25 each provide (μ, L)-two-step tensoriality for τ-splittable walk collections on the s-wide replacement product and using $\mathsf{S}^{\triangle}_{r,r}$, respectively, with

$$L \geq \frac{128k^4 \cdot 2^{4k}}{\mu^4} \quad \text{and} \quad \tau \leq \frac{\mu}{4k \cdot 2^{4k}}.$$

To get $\mu = \eta^8/2^{30}$, we require

$$L \geq \frac{K' \cdot k^4 \cdot 2^{4k}}{\eta^{32}} \quad \text{and} \quad \tau \leq \tau_0(\eta, k) = \frac{\eta^8}{K \cdot k \cdot 2^{4k}},$$

where K and K' are (very large) constants. This ensures that τ is small enough for the parity sampling requirement as well. With these parameters, the running time for the list decoding algorithm in Theorem 3.9.2 becomes

$$n^{O(L+k)}(\text{polylog}(1/\eta) + f(n)) = n^{O(L)} \cdot f(n) = n^{O(1/\tau_0(\eta,k)^4)} \cdot f(n).$$

■

For decoding in fixed polynomial time, we also need a variation of list decoding where we don't run the unique decoding algorithm of the base code and only obtain an approximate list of solutions. The proof is very similar to the proof of Theorem 3.9.1 above.

Theorem 3.9.3 (Restatement of Theorem 3.6.12)**.** *Let $\eta_0 \in (0, 1/4)$ be a constant, $\eta \in$*

$(0, \eta_0)$, $\zeta = 1/8 - \eta_0/8$, *and*

$$k \geq k_0'(\eta) \coloneqq \Theta(\log(1/\eta)).$$

Suppose $\mathcal{C} \subseteq \mathbb{F}_2^n$ *is an* η_0*-balanced linear code and* $\mathcal{C}' = \mathrm{dsum}_{W(k)}(\mathcal{C})$ *is the direct sum lifting of* $\mathcal{C}$ *on a* τ*-splittable collection of walks* $W(k)$*, where* $W(k)$ *is either the set of walks* $W[0, s]$ *on an* s*-wide replacement product graph or a set of walks using the random walk operator* $\mathsf{S}^{\triangle}_{r,r}$*. There exists an absolute constant* $K > 0$ *such that if*

$$\tau \leq \tau_0(\eta, k) \coloneqq \frac{\eta^8}{K \cdot k \cdot 2^{4k}},$$

then the code $\mathcal{C}'$ *is* η*-balanced,* $W(k)$ *is a* $(1-2\zeta, \eta)$*-parity sampler, and we have the following:*

If $\tilde{y}$ *is* $(1/2 - \sqrt{\eta})$*-close to* $\mathcal{C}'$*, then we can compute a* ζ*-cover* $\mathcal{L}'$ *of the list*

$$\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}') \coloneqq \left\{(z, \mathrm{dsum}_{W(k)}(z)) \mid z \in \mathcal{C}, \Delta\Big(\mathrm{dsum}_{W(k)}(z), \tilde{y}\Big) \leq \frac{1}{2} - \sqrt{\eta}\right\}$$

in which $\Delta(y', \tilde{y}) \leq 1/2 - \sqrt{\eta}$ *for every* $(z', y') \in \mathcal{L}'$*,*[9] *in time*

$$n^{O(1/\tau_0(\eta,k)^4)}.$$

Otherwise, we return $\mathcal{L}' = \emptyset$ *with the same running time of the preceding case.*

Proof. The list decoding framework produces a cover $\mathcal{L}'$ of the list $\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}')$, and, as its final step, corrects the cover to obtain the actual list $\mathcal{L}(\tilde{y}, \mathcal{C}, \mathcal{C}')$ by running the unique decoding algorithm of $\mathcal{C}$ on each entry of $\mathcal{L}'$ (see [AJQ+20] for details). Using Theorem 3.9.2 with a $(1 - 2\zeta, \eta)$-parity sampler and omitting this final step of the algorithm, we can obtain the ζ-cover $\mathcal{L}'$ in time $n^{O(L+k)}\mathrm{polylog}(1/\eta)$.

The tensoriality part of the proof of Theorem 3.9.1 applies here unchanged, so we need

9. A randomized rounding will ensure this, but see Section 3.10.4 for how to obtain this property deterministically.

only make sure k is large enough to yield the stronger parity sampling necessary for this theorem. As in that proof, we have that $W(k)$ is an (η_0', η')-parity sampler with $\eta' \leq (\eta_0' + 2\tau)^{\lfloor (k-1)/2 \rfloor}$. Take $\eta_0' = 1 - 2\zeta = 3/4 + \eta_0/4$. Using $\eta_0 < 1/4$ and assuming $\tau < 1/16$, we have

$$\eta' \leq (\eta_0' + 2\tau)^{\lfloor (k-1)/2 \rfloor} \leq (3/4 + \eta_0/4 + 2\tau)^{k/2-1} < (15/16)^{k/2-1},$$

which will be smaller than η as long as k is at least

$$k_0'(\eta) = 2\left(1 + \frac{\log(1/\eta)}{\log(16/15)}\right) = \Theta(\log(1/\eta)).$$

■

3.10 Auxiliary Results

3.10.1 Obtaining Tensoriality

A key result used in the SOS rounding analysis is embodied in Lemma 3.10.1 below. Roughly speaking, it quantifies the decrease in the potential Φ^G, under conditioning on a random $\mathbf{Y}_i$ for $i \sim V$, when the ensemble $\{\mathbf{Y}_i\}$ has non-trivial correlation over the edges and G is a strong enough expander graph. A generalization of this result to low threshold rank graphs was present in [BRS11]. To derive sharper parameters in the simpler expander case and to make the presentation self-contained, we give (essentially) a full proof of this result.

Lemma 3.10.1 (Progress Lemma)**.** *Suppose G satisfies $\lambda_2(G) \leq \beta^2/q^4$. If*

$$\mathop{\mathbb{E}}_{i \sim j} \left[\left\| \{\mathbf{Y}_i \mathbf{Y}_j\} - \{\mathbf{Y}_i\}\{\mathbf{Y}_j\} \right\|_1 \right] \geq \beta,$$

then

$$\mathop{\mathbb{E}}_{j \sim V} \left[\Phi^G_{|\mathbf{Y}_j} \right] \leq \Phi^G - \frac{\beta^2}{4 \cdot q^4}.$$

Expander Case

We will need the following characterization of the spectral gap of regular graph G. We denote by A_G its adjacency operator and by L_G its Laplacian operator [Chu97].

Fact 3.10.2 (Spectral Gap [Chu97])**.**

$$\lambda_2(\mathsf{L}_G) \;=\; \min_{v_1,\ldots,v_n \in \mathbb{R}^n} \frac{\mathbb{E}_{i\sim j} \left\| v_i - v_j \right\|^2}{\mathbb{E}_{i,j\sim V} \left\| v_i - v_j \right\|^2}.$$

Using the above characterization, we derive the following local-to-global result.

Lemma 3.10.3 (Local-to-Global)**.** *Let $v_1, \ldots, v_n \in \mathbb{R}^n$ be vectors in the unit ball. Suppose $\lambda_2(\mathsf{L}_G) \geq 1 - \beta/2$ (equivalently $\lambda_2(\mathsf{A}_G) \leq \beta/2$). If $\mathbb{E}_{i\sim j} \left\langle v_i, v_j \right\rangle \geq \beta$, then*

$$\mathbb{E}_{i,j\sim V} \left\langle v_i, v_j \right\rangle \;\geq\; \frac{\beta}{2}.$$

Proof. Using Fact 3.10.2, we have

$$\lambda_2(\mathsf{L}_G) \;\leq\; \frac{\mathbb{E}_{i\sim V} \left\| v_i \right\|^2 - \mathbb{E}_{i\sim j} \left\langle v_i, v_j \right\rangle}{\mathbb{E}_{i\sim V} \left\| v_i \right\|^2 - \mathbb{E}_{i,j\sim V} \left\langle v_i, v_j \right\rangle}.$$

Set $\lambda_2 = \lambda_2(\mathsf{L}_G)$. We consider two cases: $\lambda_2 \leq 1$ and $\lambda_2 > 1$. First, suppose $\lambda_2 \leq 1$. Then

$$\begin{aligned}
\mathbb{E}_{i,j\sim V} \left\langle v_i, v_j \right\rangle \;&\geq\; \frac{1}{\lambda_2} \mathbb{E}_{i\sim j} \left\langle v_i, v_j \right\rangle - \left(\frac{1-\lambda_2}{\lambda_2} \right) \mathbb{E}_{i\sim V} \left\| v_i \right\|^2 \\
&\geq\; \frac{1}{\lambda_2} \left(\beta - (1-\lambda_2) \right) \\
&\geq\; \frac{1}{\lambda_2} \left(\beta - \left(\frac{\beta}{2} \right) \right) \geq \frac{\beta}{2}.
\end{aligned}$$

Now suppose $\lambda_2 > 1$. Then

$$\begin{aligned}\mathbb{E}_{i,j\sim V}\left\langle v_i, v_j\right\rangle &\geq \frac{1}{\lambda_2}\mathbb{E}_{i\sim j}\left\langle v_i, v_j\right\rangle - \left(\frac{1-\lambda_2}{\lambda_2}\right)\mathbb{E}_{i\sim V}\left\|v_i\right\|^2 \\ &\geq \frac{1}{\lambda_2}\mathbb{E}_{i\sim j}\left\langle v_i, v_j\right\rangle \geq \frac{1}{\lambda_2}\cdot\beta \geq \frac{\beta}{2},\end{aligned}$$

where the last inequality follows from $\lambda_2 \leq 2$ for any graph G. ∎

We will need some standard notions in information theory [CT06].

Definition 3.10.4 (Relative Entropy/Kullback-Leibler Divergence)**.** *The relative entropy of two distributions D_1 and D_2 with support contained in $\mathcal{Q}$ is*

$$\mathrm{KL}(D_1, D_2) \coloneqq \sum_{a\in\mathcal{Q}} D_1(a)\log\left(\frac{D_1(a)}{D_2(a)}\right).$$

Notation 3.10.5. *Let $\mathbf{X}$ be a random variable. We denote by $\{\mathbf{X}\}$ the distribution of $\mathbf{X}$.*

Definition 3.10.6 (Mutual Information)**.** *Let $\mathbf{X}, \mathbf{Y}$ be two random variables. The mutual information $\mathrm{I}(\mathbf{X}, \mathbf{Y})$ is*

$$\mathrm{I}(\mathbf{X}, \mathbf{Y}) \coloneqq \mathrm{KL}(\{\mathbf{X}, \mathbf{Y}\}, \{\mathbf{X}\}\{\mathbf{Y}\}).$$

Fact 3.10.7.

$$\mathrm{I}(\mathbf{X}, \mathbf{Y}) = \mathrm{H}(\mathbf{X}) - \mathrm{H}(\mathbf{X}|\mathbf{Y}).$$

Fact 3.10.8 (Fact B.5 of Raghavendra and Tan [RT12])**.** *Let $\mathbf{X}_a$ and $\mathbf{X}_b$ be indicator random variables. Then*

$$\mathrm{Cov}(\mathbf{X}_a, \mathbf{X}_b)^2 \leq 2\cdot\mathrm{I}(\mathbf{X}_a, \mathbf{X}_b).$$

We are ready to prove Lemma 3.10.1 which we restate below for convenience.

Lemma 3.10.9 (Progress Lemma (restatement of Lemma 3.10.1))**.** *Suppose G satisfy $\lambda_2(G) \leq$*

β^2/q^4. *If*

$$\mathop{\mathbb{E}}_{i\sim j}\left[\left\|\{\mathbf{Y}_i\mathbf{Y}_j\}-\{\mathbf{Y}_i\}\{\mathbf{Y}_j\}\right\|_1\right]\geq\beta,$$

then

$$\mathop{\mathbb{E}}_{j\sim V}\left[\Phi^G_{|\mathbf{Y}_j}\right]\leq\Phi^G-\frac{\beta^2}{4\cdot q^4}.$$

Proof. Firstly, we show how to relate the distances $\left\|\{\mathbf{Y}_i\mathbf{Y}_j\}-\{\mathbf{Y}_i\}\{\mathbf{Y}_j\}\right\|_1$ over the edges $i\sim j$ to certain covariances. Let $a,b\in[q]^2$. Observe that

$$\left|\mathrm{Cov}\left(\mathbf{Y}_{i,a},\mathbf{Y}_{j,b}\right)\right| \;=\; \left|\Pr[\mathbf{Y}_i=a\wedge\mathbf{Y}_j=b]-\Pr[\mathbf{Y}_i=a]\Pr[\mathbf{Y}_j=b]\right|.$$

We have

$$\begin{aligned}\mathop{\mathbb{E}}_{i\sim j}\left[\frac{1}{q^2}\sum_{a,b\in[q]^2}\mathrm{Cov}\left(\mathbf{Y}_{i,a},\mathbf{Y}_{j,b}\right)^2\right] &\geq \left(\mathop{\mathbb{E}}_{i\sim j}\left[\frac{1}{q^2}\sum_{a,b\in[q]^2}\left|\mathrm{Cov}\left(\mathbf{Y}_{i,a},\mathbf{Y}_{j,b}\right)\right|\right]\right)^2\\ &\geq \frac{1}{q^4}\left(\mathop{\mathbb{E}}_{i\sim j}\left[\left\|\{\mathbf{Y}_i\mathbf{Y}_j\}-\{\mathbf{Y}_i\}\{\mathbf{Y}_j\}\right\|_1\right]\right)^2\geq\frac{\beta^2}{q^4}.\end{aligned}$$

Note that the graph $\mathcal{F}\coloneqq G\otimes J/q$ is an expander with $\lambda_2(G\otimes J/q)=\lambda_2(G)$. Moreover, the matrix $\mathsf{C}\coloneqq\{\mathrm{Cov}\left(\mathbf{Y}_{i,a},\mathbf{Y}_{j,b}\right)\}_{i,j\in V;a,b\in[q]^2}$ is PSD since the vectorization $\{v_{i,a}-\mathbb{E}[\mathbf{Y}_{i,a}]\cdot v_\emptyset\}_{i\in V;a\in[q]}$ gives a Gram matrix decomposition of C. Thus, the covariance matrix $\{\mathrm{Cov}\left(\mathbf{Y}_{i,a},\mathbf{Y}_{j,b}\right)^2\}_{i,j\in V;a,b\in[q]^2}$ is also PSD since it is the Schur product (i.e., entrywise product) of two PSD matrices, namely, $\mathsf{C}\circ\mathsf{C}$. Therefore, we are in position of applying the

local-to-global Lemma 3.10.3 with the expander $\mathcal{F}$ and a vectorization for $\mathsf{C} \circ \mathsf{C}$. We have

$$\begin{aligned}
\frac{\beta^2}{q^4} &\leq \mathop{\mathbb{E}}_{i\sim j}\left[\frac{1}{q^2}\sum_{a,b\in[q]^2}\mathrm{Cov}\left(\mathbf{Y}_{i,a},\mathbf{Y}_{j,b}\right)^2\right] \\
&\leq 2\mathop{\mathbb{E}}_{i,j\sim V^{\otimes 2}}\left[\frac{1}{q^2}\sum_{a,b\in[q]^2}\mathrm{Cov}\left(\mathbf{Y}_{i,a},\mathbf{Y}_{j,b}\right)^2\right] && \text{(local-to-global Lemma 3.10.3)} \\
&\leq \frac{4}{q^2}\mathop{\mathbb{E}}_{i,j\sim V^{\otimes 2}}\left[\sum_{a,b\in[q]^2}\mathrm{I}\left(\mathbf{Y}_{i,a},\mathbf{Y}_{j,b}\right)\right] && \text{(Fact 3.10.8)} \\
&\leq \frac{4}{q^2}\mathop{\mathbb{E}}_{i,j\sim V^{\otimes 2}}\left[\sum_{a,b\in[q]^2}\mathrm{H}\left(\mathbf{Y}_{i,a}\right)-\mathrm{H}\left(\mathbf{Y}_{i,a}|\mathbf{Y}_{j,b}\right)\right] \\
&\leq \frac{4}{q}\left[\mathop{\mathbb{E}}_{i\sim V}\left[\sum_{a\in[q]}\mathrm{H}\left(\mathbf{Y}_{i,a}\right)\right]-\mathop{\mathbb{E}}_{i,j\sim V^{\otimes 2}}\left[\sum_{a\in[q]}\mathrm{H}\left(\mathbf{Y}_{i,a}|\mathbf{Y}_{j}\right)\right]\right] \\
&= 4\left[\mathop{\mathbb{E}}_{i\sim V}\left[\mathcal{H}\left(\mathbf{Y}_i\right)\right]-\mathop{\mathbb{E}}_{i,j\sim V^{\otimes 2}}\left[\mathcal{H}\left(\mathbf{Y}_i|\mathbf{Y}_j\right)\right]\right] \\
&= 4\left[\Phi^G-\mathop{\mathbb{E}}_{j\sim V}\left[\Phi^G_{|\mathbf{Y}_j}\right]\right].
\end{aligned}$$

Therefore, we have $\mathbb{E}_{j\sim V}[\Phi^G_{|\mathbf{Y}_j}] \leq \Phi^G - \beta^2/(4\cdot q^4)$, as claimed. ∎

3.10.2 Explicit Structures

We recall some explicit structures used in Ta-Shma's construction.

Explicit Ramanujan Graphs

The outer graph G in the s-wide replacement product was chosen to be a Ramanujan graph. Ta-Shma provides a convenient lemma to efficiently obtain explicit Ramanujan graphs given the intended number of vertices n (which might end up being nearly twice this much), the expansion λ and an error parameter $\theta > 0$. These Ramanujan graphs are based on the LPS construction [LPS88]. Due to number theoretic reasons, we might be forced to work with

slightly different parameters, but this is not an issue.

Lemma 3.10.10 (Lemma 12 [TS17]). *For every $\theta > 0$, there exists an algorithm that given n and $\lambda \in (0,1)$ runs in time* $\mathrm{poly}(n)$ *and outputs a Ramanujan graph G such that*

- *G has degree $d \leq 8/\lambda^2$,*
- *$\sigma_2(G) \leq \lambda$, and*
- *$|V(G)|$ is either in the range $[(1-\theta)n, n]$ or in the range $[(1-\theta)2n, 2n]$.*

Moreover, the algorithm outputs a locally invertible function $\varphi\colon [d] \to [d]$ computable in polynomial time in its input length.

Explicit Biased Distribution

The inner graph H in the s-wide replacement product is chosen to be a Cayley graph on $\mathbb{Z}_2^m$ for some positive integer m. Ta-Shma uses the construction of Alon et al. [AGHP92] (AGHP) to deduce a result similar to Lemma 3.10.11 below. To compute the refined parameter version of our main result Theorem 3.1.1, we will need the specifics of the AGHP construction.

Lemma 3.10.11 (Based on Lemma 6 [TS17]). *For every $\beta = \beta(m)$, there exists a fully explicit set $A \subseteq \mathbb{Z}_2^m$ such that*

- *$|A| \leq 4 \cdot m^2/\beta^2$, and*
- *for every $S \subseteq [m]$, we have $|\mathbb{E}_{z \in A} \chi_S(z)| \leq \beta$.*

Furthermore, if m/β is a power of 2, then $|A| = m^2/\beta^2$. In particular, the graph $\mathrm{Cay}(\mathbb{Z}_2^m, A)$ *is a $(n = 2^m, d = |A|, \lambda = \beta)$ expander graph.*

Remark 3.10.12. *Given $d, m \in \mathbb{N}^+$ such that d is the square of a power of 2 with $d \leq 2^m$, by setting $\beta = m/\sqrt{d}$ we can use Lemma 3.10.11 with β and m (note that m/β is a power of 2) to obtain a Cayley graph* $\mathrm{Cay}(\mathbb{Z}_2^m, A)$ *with parameters $(n = 2^m, d = |A|, \lambda = \beta)$.*

3.10.3 Zig-Zag Spectral Bound

We prove the zig-zag spectral bound Fact 3.4.4.

Claim 3.10.13. *Let G be an outer graph and H be an inner graph used in the s-wide replacement product. For any integer $0 \leq i \leq s-1$,*

$$\sigma_2((\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i(\mathsf{I} \otimes \mathsf{A}_H)) \leq \sigma_2(G) + 2 \cdot \sigma_2(H) + \sigma_2(H)^2.$$

Proof. Let v be a unit vector such that $v \perp \mathbf{1}$, and decompose it into $v = u + w$ such that $u \in \mathcal{W}^{\|} = \mathrm{span}\{a \otimes b \in \mathbb{R}^{V(G)} \otimes \mathbb{R}^{V(H)} \mid b = \mathbf{1}\}$ and $w \in \mathcal{W}^{\perp} = (\mathcal{W}^{\|})^{\perp}$.

$$\begin{aligned}
|\langle v, (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i(\mathsf{I} \otimes \mathsf{A}_H)v\rangle| \leq\ & |\langle u, (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i(\mathsf{I} \otimes \mathsf{A}_H)u\rangle| + |\langle u, (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i(\mathsf{I} \otimes \mathsf{A}_H)w\rangle| + \\
& |\langle w, (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i(\mathsf{I} \otimes \mathsf{A}_H)u\rangle| + |\langle w, (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i(\mathsf{I} \otimes \mathsf{A}_H)w\rangle| \\
\leq\ & |\langle u, \mathsf{G}_i u\rangle| + \|(\mathsf{I} \otimes \mathsf{A}_H)w\| + \\
& \|(\mathsf{I} \otimes \mathsf{A}_H)w\| + \|(\mathsf{I} \otimes \mathsf{A}_H)w\|^2 \\
\leq\ & |\langle u, \mathsf{G}_i u\rangle| + 2\sigma_2(H) + \sigma_2^2(H)
\end{aligned}$$

To bound $|\langle u, (\mathsf{I} \otimes \mathsf{A}_H)\mathsf{G}_i(\mathsf{I} \otimes \mathsf{A}_H)u\rangle|$, observe that $u = x \otimes \mathbf{1}$ for some $x \in \mathbb{R}^{V(G)}$. Then,

$$0 = \langle v, \mathbf{1}\rangle = \langle u, \mathbf{1}\rangle + \langle w, \mathbf{1}\rangle = \langle u, \mathbf{1}\rangle = \langle x, \mathbf{1}_G\rangle$$

so that $x \perp \mathbf{1}_G$. Because u is uniform over the H-component, $|\langle u, \mathsf{G}_i u\rangle| = |\langle x, \mathsf{G}x\rangle| \leq \sigma_2(G)$, which completes the proof. ∎

We also derive a (simple) tighter bound for the expansion of the zig-zag product in a particular parameter regime.

Claim 3.10.14. *Let G be a λ_1-two-sided expander and H be a λ_2-two-sided expander such*

that both are regular graphs. If $\lambda_1 \leq \lambda_2$, then

$$\sigma_2(G \circledS H) \leq 2 \cdot \lambda_2.$$

Proof. Let $v = a \cdot v^{\parallel} + b \cdot v^{\perp}$ with $a^2 + b^2 = 1$ be such that $v \perp 1$. In particular, if $v^{\parallel} = v^G \otimes 1^H$, then $v^G \perp 1^G$ since otherwise $\langle v, 1 \rangle = \left\langle v^G, 1^G \right\rangle \neq 0$. We have

$$\begin{aligned} \max_{a,b \in \mathbb{R} \colon a^2+b^2=1} a^2 \cdot \lambda_1 + 2ab \cdot \lambda_2 + b^2 \cdot \lambda_2^2 &\leq \max_{a,b \in \mathbb{R} \colon a^2+b^2=1} a^2 \cdot \lambda_2 + 2ab \cdot \lambda_2 + b^2 \cdot \lambda_2 \\ &= \max_{a,b \in \mathbb{R} \colon a^2+b^2=1} \lambda_2 + 2ab \cdot \lambda_2, \end{aligned}$$

where the inequality follows from the assumption $\lambda_1 \leq \lambda_2$ (and trivially $\lambda_2^2 \leq \lambda_2$) and the equality follows from $a^2 + b^2 = 1$. Since we also have $2ab = (a+b)^2 - (a^2 + b^2) \leq 1$, the result follows. ∎

3.10.4 Derandomization

To deterministically uniquely decode in fixed polynomial time (i.e., $\mathrm{poly}(n/\varepsilon)$), we need to prune the list of coupled words $\mathcal{L}$ covering the list $\mathcal{L}^*(\tilde{y}) = \{(z, y = \mathrm{dsum}(z)) \mid z \in \mathcal{C}, \Delta(\tilde{y}, y) \leq 1/2 - \sqrt{n}\}$ of codewords we want to retrieve. To do so, given $(z^*, y^* = \mathrm{dsum}(z^*)) \in \mathcal{L}^*(\tilde{y})$, we need to have $(z, y = \mathrm{dsum}(z)) \in \mathcal{L}$ such that

1. $|\langle y^*, \mathrm{dsum}(z) \rangle|$ is not too small, and
2. $\langle \tilde{y}, \mathrm{dsum}(z) \rangle$ is not too small (in order to apply Lemma 3.6.11).

The slice (S, σ) of the SOS solution from which y^* is recoverable satisfies in expectation

$$\mathbb{E}_{z \sim \{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}} \left[\langle y^*, \mathrm{dsum}(z) \rangle^2 \right] \geq 3\eta^2,$$

and

$$\mathbb{E}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}}\left[\langle\tilde{y},\mathrm{dsum}(z)\rangle\right]\geq 3\sqrt{\eta}/2.$$

Moreover, since $z\mapsto\langle y^*,\mathrm{dsum}(z)\rangle^2$ and $z\mapsto\langle\tilde{y},\mathrm{dsum}(z)\rangle$ are $O(1/n)$-Lipschitz with respect to the ℓ_1-norm,[10] Hoeffding's inequality gives

$$\mathop{\mathbb{P}}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}}\left[\langle y^*,\mathrm{dsum}(z)\rangle^2<\eta^2\right]\leq\exp\left(-\Theta(n)\right),$$

and

$$\mathop{\mathbb{P}}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}}\left[\langle\tilde{y},\mathrm{dsum}(z)\rangle<\sqrt{\eta}\right]\leq\exp\left(-\Theta(n)\right).$$

At least randomly, such a z can be easily found. In [AJQ+20], alternatively to satisfying Item 1 it was shown that by choosing $z'\in\{\pm1\}^n$ by majority vote, i.e.

$$z'_i=\mathop{\mathrm{argmax}}_{b\in\{\pm1\}}\mathbb{P}[\mathbf{Z}_i=b]$$

for $i\in[n]$, one has that $\left|\left\langle z^*,z'\right\rangle\right|$ is large which is enough to address the first item. More precisely, implicit in [AJQ+20], for any constant $\beta\in(0,1)$ as long as parity sampling is sufficiently strong we have

$$\mathbb{E}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}}\left[\left\langle z',z\right\rangle^2\right]\geq 1-\beta.$$

Similarly $z\mapsto\left\langle z',z\right\rangle^2$ is $O(1/n)$-Lipschitz with respect to the ℓ_1-norm, so Hoeffding's inequality yields

$$\mathop{\mathbb{P}}_{z\sim\{\mathbf{Z}^{\otimes}|_{(S,\sigma)}\}}\left[\left\langle z',z\right\rangle^2<1-\beta/2\right]\leq\exp\left(-\Theta(n)\right).$$

However, we want to efficiently and deterministically find a z satisfying $\left\langle z',z\right\rangle^2\geq 1-\beta/2$

10. In this fixed polynomial time regime, the parameters $s,d_1,d_2,\varepsilon_0,\eta$ are constant independent of the final bias ε.

as well as satisfying Item 2. Note that at this stage in the decoding process y^* is not known (without issuing a recursive unique decoding call), so running expectation maximization to satisfy item Item 1 would not be possible. Fortunately, the majority z' can be cheaply computed without a recursive call to a unique decoder. On the other hand z satisfying only Item 2 can be found by expectation maximization. We need to satisfy both conditions at the same time. For this reason, we design a simultaneous expectation maximization derandomization procedure tailored to our setting.

Abstract Derandomization: Simultaneous Expectation Maximization

Suppose that Ω is a probability space where two random variables $\mathbf{A}$ and $\mathbf{B}$ are defined satisfying the following first moment conditions

$$\mathbb{E}[\mathbf{A}] \geq a \quad \text{and} \quad \mathbb{E}[\mathbf{B}] \geq 1 - \beta.$$

We provide sufficient conditions so that $\omega \in \Omega$ satisfying

$$\mathbf{A}(\omega) \geq a' \quad \text{and} \quad \mathbf{B}(\omega) \geq 1 - \beta'$$

can be efficiently deterministically found with the aid of an oracle, where $a \approx a'$ and $\beta \approx \beta'$. More precisely, we have the following lemma.

Lemma 3.10.15. *Let $\Omega = (\{-1, 1\}^n, \nu_1 \times \cdots \times \nu_n)$ be a probability space with a product distribution. Suppose $\mathbf{A} \in [-1, 1]$ is a random variable on Ω satisfying, for $a > 0$ and for some function $e_A \colon \mathbb{N} \to \mathbb{R}^+$,*

$$\mathbb{P}[\mathbf{A} < a] \leq e_A(n).$$

Suppose $\mathbf{B} \in [-1, 1]$ is a random variable on Ω satisfying, for some function $e_B \colon \mathbb{N} \times \mathbb{R}^+ \to$

$\mathbb{R}^+$,

$$\mathbb{P}\left[\mathbf{B} < 1 - \beta\right] \leq e_B(n, \beta).$$

Suppose that there is an oracle to evaluate $\mathbb{E}\left[\mathbf{A}\mathbf{B}^{2k}\right]$ *under any product distribution* $\mu_1' \times \cdots \times \mu_n'$ *for* $k \in \mathbb{N}$. *Given* $\delta, \beta \in (0, 1)$, *if*

$$e_A(n) + e_B(n, \beta/(4(\lceil -\ln(a(1-\beta))/\delta \rceil + 1))) \leq a\frac{\beta}{2}, \tag{3.10}$$

then it is possible to find $\omega \in \{\pm 1\}^n$ *using* $2n$ *invocations to the oracle and satisfying*

$$\mathbf{A}(\omega) \geq a(1-\beta) \quad \textit{and} \quad |\mathbf{B}(\omega)| \geq 1 - \delta.$$

Proof. Set $k = \lceil -\ln(a(1-\beta))/\delta \rceil + 1$. Set $\beta' = \beta/(4k)$. Note that

$$\mathbb{E}\left[\mathbf{A}\mathbf{B}^{2k}\right] \geq a\left(1 - \frac{\beta}{4k}\right)^{2k} - e_A(n) - e_B(n, \beta') \geq a\left(1 - \beta\right),$$

where we use Eq. (3.10) in the last inequality. Do expectation maximization to deterministically find $\omega \in \{\pm 1\}^n$, with $2 \cdot n$ invocations to the oracle of $\mathbb{E}\left[\mathbf{A}\mathbf{B}^{2k}\right]$, such that

$$\mathbf{A}(\omega)\mathbf{B}(\omega)^{2k} \geq a\left(1 - \beta\right).$$

Since $\mathbf{B}(\omega)^{2k} \leq 1$, we have $\mathbf{A}(\omega) \geq a\left(1 - \beta\right)$. Towards a contradiction suppose $|\mathbf{B}(\omega)| \leq 1 - \delta$. Using that $\mathbf{A}(\omega) \leq 1$, we have

$$e^{-2k\cdot\delta} \geq (1-\delta)^{2k} \geq \mathbf{A}(\omega)\mathbf{B}(\omega)^{2k} \geq a(1-\beta). \tag{3.11}$$

By our choice of k, we get

$$e^{-2k\cdot\delta} < a(1-\beta),$$

contradicting Eq. (3.11). ■

Implementing the Oracle

Now, we provide an efficient deterministic oracle for our setting. We take

$$\mathbf{A} \coloneqq \langle \tilde{y}, \mathrm{dsum}(z) \rangle \quad \text{and} \quad \mathbf{B} \coloneqq \left\langle z', z \right\rangle^2,$$

where $z'_i = \mathrm{argmax}_{b \in \{\pm 1\}} \mathbb{P}[\mathbf{Z}_i = b]$. Note that

$$\mathbf{A}\mathbf{B}^{2k} = \sum_{T \subset [n] \colon |T| = O(1)} \alpha_T \prod_{i \in T} z_i.$$

To compute $\mathbb{E}\left[\mathbf{A}\mathbf{B}^{2k}\right]$ under any product distribution $\mu'_1 \times \cdots \times \mu'_n$, use linearity of expectation and sum at most $n^{O(1)}$ terms $\alpha_T \mathbb{E}\left[\prod_{i \in T} z_i\right]$ where each can be computed in $O(1)$ since restricted to T we have a product distribution taking values in $\{\pm 1\}^T$.

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. 7, 8, 16, 72, 103, 104

[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. 101, 167, 186

[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. 7, 104, 105, 109, 110, 111, 112, 115, 127, 129, 133, 144, 156, 160, 167, 168, 177, 178, 180, 189

[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. 4, 9, 11, 21, 23, 24, 25, 36, 37, 44, 66, 67, 82, 83, 84, 86, 89, 91, 111, 112, 160, 164

[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. 107

[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. 7, 103

[BHK+16] B. Barak, S. B. Hopkins, J. Kelner, P. Kothari, A. Moitra, and A. Potechin. A nearly tight sum-of-squares lower bound for the planted clique problem, 2016. 19

[BKS17] Boaz Barak, Pravesh K. Kothari, and David Steurer. Quantum entanglement, sum of squares, and the log rank conjecture. In *Proceedings of the 49th ACM Symposium on Theory of Computing*, pages 975–988. ACM, 2017. 19

[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`. 3, 104, 110, 129

[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. 17, 23, 43, 66, 67, 82, 88, 91, 112, 153, 160, 162, 164, 181

[Cha16] Siu On Chan. Approximation resistance from pairwise-independent subgroups. *J. ACM*, 63(3), August 2016. 7, 103

[Chu97] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997. 182

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, New York, NY, USA, 2006. 183

[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. 7, 103

[Del75] P. Delsarte. The association schemes of coding theory. In *Combinatorics*, pages 143–161. Springer Netherlands, 1975. 101

[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. 8, 10, 12, 27, 30, 73, 79, 104

[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. 4, 9, 26, 27, 28, 103

[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. 7, 103

[Gal62] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962. 106

[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. 7, 103

[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. 13, 105

[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. 2, 105, 106, 107

[Gil52] E.N. Gilbert. A comparison of signalling alphabets. *Bell System Technical Journal*, 31:504–522, 1952. 1, 101, 107

[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. 106, 107

[GM12] Bernd Gärtner and Jiri Matousek. *Approximation Algorithms and Semidefinite Programming*. Applications of Mathematics. Springer-Verlag Berlin Heidelberg, 2012. 40

[GNW95] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR lemma. Technical Report TR95-50, Electronic Colloquium on Computational Complexity, 1995. 16

[GR05] Venkatesan Guruswami and Atri Rudra. Explicit capacity-achieving list-decodable codes. Technical Report TR05-133, Electronic Colloquium on Computational Complexity, 2005.

[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. 2, 101, 106

[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. 106

[Gri01] Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theor. Comput. Sci.*, 259(1-2):613–622, 2001. 23, 45, 111

[GRS19] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. 2019. 48, 101

[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. 107

[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. 112

[Gur01] Venkatesan Guruswami. *List Decoding of Error-Correcting Codes*. PhD thesis, MIT, 2001. 7

[Gur05] Venkatesan Guruswami. Algebraic-geometric generalizations of the Parvaresh-Vardy codes. *Electronic Colloquium on Computational Complexity (ECCC)*, (132), 2005. 106

[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. 101, 103, 106

[Gur10] Venkatesan Guruswami. Bridging Shannon and Hamming: List error-correction with optimal rate. In *ICM*, 2010. 101, 107

[Ham50] Richard Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, 1950. 107

[Hås97] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 1–10, 1997. 104

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. 111

[HLW06] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43(04):439–562, August 2006. 7

[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. 106, 107

[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. 7, 103

[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. 7, 103

[JQST20] Fernando Granha Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. Unique decoding of explicit ε-balanced codes near the Gilbert-Varshamov bound. In *Proceedings of the 61st IEEE Symposium on Foundations of Computer Science*, pages 434–445. IEEE, 2020. 101

[KKK19] Sushrut Karmalkar, Adam R. Klivans, and Pravesh K. Kothari. List-decodable linear regression. *CoRR*, abs/1905.05679, 2019. URL: `http://arxiv.org/abs/1905.05679`, `arXiv:1905.05679`. 12, 13, 39

[KMOW17] Pravesh Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In *Proceedings of the 49th ACM Symposium on Theory of Computing*, 2017. 23, 45

[LPS88] Alexander Lubotzky, R. Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988. 11, 81, 185

[LSV05a] Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Explicit constructions of ramanujan complexes of type ad. *Eur. J. Comb.*, 26(6):965–993, August 2005. 11, 27

[LSV05b] Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Ramanujan complexes of typeãd. *Israel Journal of Mathematics*, 149(1):267–299, Dec 2005. 11, 27

[Lub18] Alexander Lubotzky. High dimensional expanders. In *ICM*, 2018. 7

[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. 106

[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. 101

[MRRZ+19] Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. LDPC codes achieve list decoding capacity, 2019. `arXiv:1909.06430`. 106

[MU17] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2017. 96, 97

[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. 101

[NS09] Michael Navon and Alex Samorodnitsky. Linear programming bounds for codes via a covering argument. *Discrete Comput. Geom.*, 41(2):199–207, March 2009. 101

[RT12] Prasad Raghavendra and Ning Tan. Approximating CSPs with global cardinality constraints using SDP hierarchies. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms*, SODA 2012, pages 373–387, 2012. 183

[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. 103, 115, 121

[RW17] Prasad Raghavendra and Benjamin Weitz. On the bit complexity of sum-of-squares proofs. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. 40

[RY19] Prasad Raghavendra and Morris Yau. List decodable learning via sum of squares. *CoRR*, abs/1905.04660, 2019. URL: `http://arxiv.org/abs/1905.04660`, `arXiv:1905.04660`. 12, 13, 39

[Sha48] Claude Shannon. A mathematical theory of communications. *Bell System Technical Journal*, 27:379–423, 623–656, 1948. 107

[SKS19] Ronen Shaltiel, Swastik Kopparty, and Jad Silbak. Quasilinear time list-decodable codes for space bounded channels. 2019. 107

[Sti08] Henning Stichtenoth. *Algebraic Function Fields and Codes*. Springer Publishing Company, Incorporated, 2nd edition, 2008. 106

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001. 7, 16

[Sud00] Madhu Sudan. List decoding: Algorithms and applications. In *Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics*, TCS '00, pages 25–41, Berlin, Heidelberg, 2000. Springer-Verlag. 7

[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. 2, 106, 107

[Tre04] Luca Trevisan. Some applications of coding theory in computational complexity. *Quaderni di Matematica*, 13:347–424, 2004. arXiv:cs.CC/0409044. 7

[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. 2, 5, 7, 16, 24, 26, 33, 81, 93, 101, 103, 107, 109, 110, 115, 116, 124, 126, 129, 166, 186

[Var57] R.R. Varshamov. Estimate of the number of signals in error correcting codes. *Doklady Akademii Nauk SSSR*, 117:739–741, 1957. 1, 101, 107

[vL99] Jacobus H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1999. 106