Electronic Colloquium on Computational Complexity, Report No. 212 (2025)

Structure Theorems (and Fast Algorithms) for List Recovery of Subspace-Design Codes

Rohan Goyal[*] Venkatesan Guruswami[†]

Abstract

List recovery of error-correcting codes has emerged as a fundamental notion with broad applications across coding theory and theoretical computer science. Folded Reed-Solomon (FRS) and univariate multiplicity codes are explicit constructions which can be efficiently list-recovered up to capacity, namely a fraction of errors approaching $1 - R$ where R is the code rate.

Chen and Zhang and related works showed that folded Reed-Solomon codes and linear codes must have list sizes exponential in $1/\varepsilon$ for list-recovering from an error-fraction $1 - R - \varepsilon$. These results suggest that one cannot list-recover FRS codes in time that is also polynomial in $1/\varepsilon$. In contrast to such limitations, we show, extending algorithmic advances of Ashvinkumar, Habib, and Srivastava for list decoding, that even if the lists in the case of list-recovery are large, they are highly structured. In particular, we can output a compact description of a set of size only $\ell^{O((\log \ell)/\varepsilon)}$ which contains the relevant list, while running in time only polynomial in $1/\varepsilon$ (the previously known compact description due to Guruswami and Wang had size $\approx n^{\ell/\varepsilon}$). We also improve on the state-of-the-art algorithmic results for the task of list-recovery.

[*]Massachusetts Institute of Technology, Cambridge rohan_g@mit.edu.
[†]University of California, Berkeley venkatg@berkeley.edu.

ISSN 1433-8092

Contents

1 Introduction

List recovery of error-correcting codes is a generalization of list and unique decoding, where instead of receiving a single corrupted symbol for each coordinate, the decoder receives a set of possible candidates for each position. This is often used to model *decoding with soft information* i.e. where there is some uncertainty about each symbol but we would still like to recover all close-by codewords which are within the received sets in most coordinates.

Initially explicitly defined and named in [GI01] to construct efficiently decodable codes, the notion of list recovery has found a surprisingly diverse array of applications spanning constructions of condensers and extractors [SU05, GUV09, KTS22], group testing [INR10, CHKV11], compressed sensing [NPR11], heavy hitters [LNNT19, DW22], cryptography [HLR21], quantum advantage [YZ24, CT25, JSW+25] and more. The problem of list recovery is formalized as follows:

List-Recovery. An error-correcting code $\mathcal{C} \subseteq \Sigma^n$ is said to be (δ, ℓ, L) list-recoverable if given arbitrary subsets $L_1, \cdots, L_n \subseteq \Sigma$ each of size at most ℓ, there are at most L codewords $c \in \mathcal{C}$ for which there exists a $y \in L_1 \times \cdots \times L_n$ such that c and y differ on at most δn coordinates.

The case when $\ell = 1$ corresponds to **list-decoding** (up to a fraction δ of errors).

It is easily seen that any rate R error-correcting code $\mathcal{C}$ is not $(1-R+\varepsilon, \ell, q^{\varepsilon n})$ list-recoverable, even for $\ell = 1$. On the other hand, a random code of rate R over a sufficiently large alphabet $\Sigma = \Sigma(R, \varepsilon, \ell)$ is $(1 - R - \varepsilon, \ell, O(\ell/\varepsilon))$ list-recoverable with high probability [Res20].

Thus $1 - R$ is the *capacity* of list decoding/list recovery, as a function of the rate.

A fundamental question then is whether we can construct explicit codes which are list-recoverable up to distance $1 - R - \varepsilon$ with small lists.

Guruswami and Rudra [GR08] constructed explicit codes, namely folded Reed-Solomon (FRS) codes, which they showed to be efficiently list-recoverable up to distance $1 - R - \varepsilon$ with polynomially sized lists. This achieved the optimal trade-off between rate and decoding radius. However, the output list-size L in this result was very large, growing as $n^{\Omega(\ell/\varepsilon)}$.

Guruswami and Wang [GW13] showed that while the output list is large, it is contained in a low-dimensional subspace. This enabled improvements to the list-size by pre-coding the FRS code into an appropriate subspace-evasive set [GW13, DL12]. Furthermore, they also led to list-size improvements for FRS codes by *pruning* the concerned subspace to only include codewords close to the input.

It has recently become apparent that the fundamental property of FRS codes (and related codes such as univariate multiplicity codes) which implies both the containment of the list in a low-dimensional subspace and an effective pruning strategy to bound its size is a certain *subspace-design* property possessed by these codes. Informally, a rate R code $\mathcal{C} \subseteq (\mathbb{F}^s)^n$ is a good subspace-design code if for every low-dimensional subspace A of $\mathcal{C}$, the average dimension of $A \cap \mathcal{C}_i$ for random $i \in [n]$ drops to $\approx R \cdot \dim(A)$, where $\mathcal{C}_i = \{c \in \mathcal{C} \mid c_i = 0\}$.

The subspace-design property has since been used to bring down list sizes L for $(1-R-\varepsilon, \ell, L)$-list recovery of FRS codes to $O_{\ell,\varepsilon}(1)$, i.e., independent of the code length n, in [KRSW23, Tam24]. Similar list size bounds have also been shown for random linear and randomly punctured Reed-Solomon codes [LMS25, LS25].

The list-sizes L achieved by these list recovery results are *exponential* in $1/\varepsilon$ (unlike the case of list-decoding where the list sizes can be polynomial, and in fact linear, in $1/\varepsilon$ [AGG+25, Sri25, CZ25]). Perhaps surprisingly, this has been shown to be inherent. Specifically, a FRS code, or any linear code, cannot be $(1-R-\varepsilon, \ell, \ell^{o(R/\varepsilon)})$ list-recoverable [CZ25, LMS25, LS25]. For a recent survey of exciting developments concerning list recovery, we refer to Resch and Venkitesh [RV25].

The above list-size lower bound for list recovery implies that it is not feasible to output the entire list in time $\mathrm{poly}(\ell/\varepsilon, n)$. It might, however, still be possible that the list, despite being large, has a compact description and such a description can be found in time $\mathrm{poly}(n, \ell/\varepsilon)$. This is driving motivation for our work and we obtain results that make progress in this direction.

1.1 Our Results

Following [GW13], which showed the important structural result that not only is the list for the list-recovery problem for FRS codes up to distance $1-R-\varepsilon$ polynomially bounded, but it is contained in an affine space of dimension $\frac{\ell}{\varepsilon}$, list-recovery results for FRS codes have proceeded in two steps:

- Finding the low-dimensional subspace containing all codewords.
- Bounding the number of close-by codewords in any low-dimensional space, either directly or using the subspace-design property, via suitable *pruning* of the subspace.

The above two steps also govern associated algorithmic results for list recovery. Recently, [GHKS25] showed that the first step can be implemented in near-linear time in the block length and polynomial in $1/\varepsilon, \ell$. The pruning step, as implemented in [KRSW23, Tam24], takes time $\widetilde{O}(n \cdot \exp(\widetilde{O}((\log^2 \ell)/\varepsilon)))$, which is exponential in $1/\varepsilon$. For the case of list-decoding, a beautiful recent work [AHS25] showed that the pruning step can be implemented in $\widetilde{O}(n \cdot \mathrm{poly}(1/\varepsilon))$ time.

Our work provides improvements on two fronts. First, we extend the [AHS25] pruning algorithm to the task of list-recovery and improve on the best known algorithmic list-recovery bounds. We efficiently (in runtime $\widetilde{O}(n \cdot \mathsf{poly}(L))$ where L is the output list-size) find a list of size $\mathrm{poly}(\ell/\varepsilon) \cdot \ell^{O((\log \ell)/\varepsilon)}$ containing the close-by codewords, which is slightly better than the earlier $(\ell/\varepsilon)^{O((\log \ell)/\varepsilon)}$ bound. We note that our list-size bound is worse than the near-optimal combinatorial bound of $(\ell/(R+\varepsilon))^{(R+\varepsilon)/\varepsilon}$ shown recently in [BCDZ25a] (see Section 1.3) but improves on the best known algorithmic results.

Second, and more importantly, as a result complementing the list-size lower bound constructions in [CZ25], we show that the lists at distance $1-R-\varepsilon$ for subspace-design codes are contained in $\mathrm{poly}(\ell/\varepsilon)$ many $(O((\log \ell)/\varepsilon), \ell)$ *sum-sets*. In particular, this gives a way to succinctly describe a set of size $\mathrm{poly}(\ell/\varepsilon) \cdot \ell^{O((\log \ell)/\varepsilon)}$ that contains all close-by codewords.

Theorem 1.1. *For any good subspace-design code, given an r-dimensional linear space, and a list recovery input $L_1, \cdots, L_n$ with $|L_i| = \ell$, we can output $\mathrm{poly}(\log \ell, 1/\varepsilon, r)$ many $(O(\frac{R(\log(r\varepsilon/R)+1)}{\varepsilon}), \ell)$ sumsets of codewords which contain all codewords from the linear space that are within $1-R-\varepsilon$ distance of the list-recovery input in time $\widetilde{O}(n \cdot \mathrm{poly}(\ell/\varepsilon, r))$.*

A (u, v) sumset is any set of the form $A_1 + A_2 + \cdots + A_u = \{a_1 + a_2 + \cdots + a_u \mid a_i \in A_i\}$ where $|A_i| \le v$.

Combining with the [GHKS25] result, which lets us find the r-dimensional space efficiently for FRS codes, we get that we can output $\mathrm{poly}(\log \ell, 1/\varepsilon, r)$ many $(O(\frac{R(\log(r\varepsilon/R)+1)}{\varepsilon}), \ell)$ *sumsets* of codewords which contain all codewords that are within $1-R-\varepsilon$ distance of the list-recovery input in time $\widetilde{O}(n \cdot \mathrm{poly}(\ell/\varepsilon))$. Thus, we get an $\widetilde{O}(n \cdot \mathrm{poly}(\ell/\varepsilon))$ algorithm to succinctly describe an $\widetilde{O}(\ell^{(\log \ell)/\varepsilon})$ sized set containing all the close-by codewords to the product set $L_1 \times L_2 \cdots \times L_n$.

1.2 Overview of Techniques

In a companion paper [GG25], we showed that the subspace *pruning* algorithm of [AHS25] for the task of list-decoding can be made *oblivious* to the received word, and only depend on the space being pruned. This was then leveraged to carry out the pruning using shared randomness to find, in one go, many codewords that all have agreements on a small common set of coordinates.

This idea was then further developed to establish optimal proximity gaps for subspace-design codes.

Here, we again begin with the observation that running [AHS25] with shared randomness for the task of list-recovery succeeds in finding each codeword within the affine space which is close to the input with high probability. At the end of running [AHS25], we are left with a *small* set of coordinates where these successfully found codewords must all have agreements; the number of such codewords can then also be argued to be small. Combining the success probability and size of the lists found already gives our list-size bounds.

A high probability bound and union bound over all close-by codewords lets us conclude that running algorithm only $\mathrm{poly}(\ell/\varepsilon)$ times suffices to efficiently find all close-by codewords.

We also prove a generic linear-algebraic result showing that the set of codewords completely agreeing with ℓ sized lists on t coordinates is contained in a (t, ℓ) sum-set and the description of this sum-set can be found efficiently.

There are two major things we need to be careful about here:

- The high probability and union bound argument can only work if the probability of finding any codeword is sufficiently high. Thus, a pruning algorithm which does not find any given codeword with probability at least $\mathrm{poly}(\frac{\varepsilon}{\ell})$ would not suffice for bounding the number of sum-sets required even if it gets better list-size bounds. In particular, it is crucial to use the codeword oblivious pruning algorithm. Else, we would be stuck in regimes where the number of sumsets required is quasi-polynomial in ℓ.
- Directly implementing the [AHS25] pruning algorithm would not guarantee that the number of coordinates all codewords in a single iteration agree on is *small* (which in turn is necessary to ensure that the dimension t of the sum-set is small). For this purpose, we need to ensure that our pruning algorithm is more *aggressive*. In particular, we deviate from the previous pruning approaches and only fix coordinates which decrease the dimension of the current space by a $(1 - \Omega(\varepsilon))$ factor. This variant of the pruning is able to improve sum-set dimension from ℓ/ε to $(\log \ell)/\varepsilon$.

1.3 Related Work

In recent work, [BCDZ25a] made a rather remarkable connection between subspace-designs and discrete Brascamp-Lieb inequalities using which they established near-optimal combinatorial list recovery bounds for subspace-design codes such as folded Reed-Solomon and univariate multiplicity codes (see Appendix A). In a companion work [BCDZ25b], they showed, adapting the techniques of [LMS25], that their near-optimal bounds can be transferred from subspace-design codes to random linear and random Reed-Solomon codes as well.

It remains an interesting open question whether our bounds can be further improved to match theirs. We address their results again in Section 5 and Appendix A.

2 Preliminaries

We begin by introducing the basic coding-theoretic definitions we will be using.

Definition 1 (Fractional Hamming Distance)**.** For any two vectors x, y in Σ^n where Σ is some alphabet, we define $\Delta(x, y) = \frac{|\{i \in [n] : x_i \neq y_i\}|}{n}$ to be the fraction of coordinates where they differ.

For a set $S \subseteq \Sigma^n$, we define $\Delta(x, S) = \min_{y \in S} \frac{|\{i \in [n] : x_i \neq y_i\}|}{n}$ to be the minimum fractional Hamming distance of x to its closest vector in S.

Two fundamental quantities associated with a code are its rate and distance.

Definition 2 (Distance of a code)**.** For code $\mathcal{C} \subseteq \Sigma^n$, we define its (relative) distance as $\Delta(\mathcal{C}) = \min_{x, y \in \mathcal{C}, x \neq y} \Delta(x, y)$

Definition 3 (Rate of a code)**.** For a code $\mathcal{C} \subseteq \Sigma^n$, its rate $R(\mathcal{C})$ is defined as $R(\mathcal{C}) = \frac{\log_{|\Sigma|} |\mathcal{C}|}{n}$

We will focus on linear/additive codes in this paper, defined as follows.

Definition 4 (Additive codes)**.** Let $\mathbb{F}$ be a finite field and let $\Sigma = \mathbb{F}^s$ for some positive integer s. A code $\mathcal{C} \subseteq \Sigma^n$ is said to be $\mathbb{F}$-additive (or just additive when the field $\mathbb{F}$ is clear from context or not relevant to the discussion) if $\mathcal{C}$ is an $\mathbb{F}$-linear subspace of Σ^n. When $s = 1$, the code is simply called a linear code.

2.1 The list-recovery problem

This paper is focused on the problem of list-recovery, formally defined below.

Problem 2.1 (List Recovery)**.** *For a code $\mathcal{C} \subseteq \Sigma^n$, the* list-recovery *problem with error rate δ and input list size ℓ is defined as follows. The input consists of subsets $L_1, \cdots, L_n \subseteq \Sigma$ with $|L_i| = \ell$ for each i, and the task is to find the list of codewords*

$$\mathsf{LIST}(\mathcal{C}, \delta, L_1 \times L_2 \times \cdots \times L_n) := \{c \in \mathcal{C} \mid \Delta(c, L_1 \times L_2 \times \cdots \times L_n) \leq \delta\} .$$

The goal is to construct codes for which the list of codewords is small for all list-recovery inputs, subject to inherent trade-offs between rate, δ and ℓ. The following shows an inherent limitation of linear and FRS codes in this regard. We note that the trade-off $\delta \to 1 - R$ is the best possible even for list-decoding.

Theorem 2.2 ([CZ25, LMS25, LS25])**.** *For all $R \in (0, 1)$, $\varepsilon \in (0, 1 - R)$, positive integers ℓ, and finite fields $\mathbb{F}$ with $|\mathbb{F}| \geq \ell$, the following holds for every $\mathbb{F}$-linear code $\mathcal{C} \subset \Sigma^n$ of rate R and sufficiently large block length. There exist inputs $L_1, \ldots, L_n \subset \Sigma$ with $|L_i| = \ell$ to the list-recovery problem for the code $\mathcal{C}$ such that*

$$|\mathsf{LIST}(\mathcal{C}, 1 - R - \varepsilon, L_1 \times L_2 \times \cdots \times L_n)| \geq \ell^{\lfloor R/\varepsilon \rfloor} .$$

This result tells us that for constant rate codes, we cannot have algorithms with runtime polynomial in $1/\varepsilon$ which output the entire list of codewords since the valid list itself could be significantly larger.

2.2 Folded Reed-Solomon Codes

The first codes that were shown to be list-recoverable up to capacity were folded Reed-Solomon codes in the work of [GR08] and had the added property that they were list-recoverable algorithmically as well in polynomial time. In the following subsection, we formally define folded Reed-Solomon codes and present previous results about their list-decoding and recovery.

Definition 5 (Folded Reed-Solomon Codes [GR08])**.** Let n, k, s be positive integers and $\mathbb{F}_q$ be a field with $q > sn$ and γ be an element of the field. Additionally, let $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}_q$ be distinct elements such that $\alpha_i \gamma^t \neq \alpha_j$ for all $i \neq j$ and $t < s$.

The s-folded Reed-Solomon Code $\mathcal{C} \subseteq (\mathbb{F}_q^s)^n$ with message length k on $(\alpha_1, \ldots, \alpha_n)$ is given by:

$$\mathsf{FRS}_{q,n,k,s}(\bar{\alpha}) = \left\{ \left(\begin{bmatrix} f(\alpha_1) \\ f(\alpha_1\gamma) \\ \vdots \\ f(\alpha_1\gamma^{s-1}) \end{bmatrix}, \begin{bmatrix} f(\alpha_2) \\ f(\alpha_2\gamma) \\ \vdots \\ f(\alpha_2\gamma^{s-1}) \end{bmatrix}, \cdots, \begin{bmatrix} f(\alpha_n) \\ f(\alpha_n\gamma) \\ \vdots \\ f(\alpha_n\gamma^{s-1}) \end{bmatrix} \right) \middle| f \in \mathbb{F}_q[x], \deg f < k \right\}$$

As mentioned previously, FRS codes are extremely useful for the problem of list-recovery and we know this due to the following results which we now state informally:

Theorem 2.3 ([GW13, GR08, Kop14])**.** *For every choice of $\ell \in \mathbb{N}, \varepsilon > 0, R > 0$, there exists an $s_0 = \Omega(\ell/\varepsilon^2)$, such that any rate R s-folded-Reed Solomon code $\mathcal{C}$ for $s > s_0$ contains an affine space $\mathcal{H}$ of dimension $r < O(\ell/\varepsilon)$ such that $\mathsf{LIST}(\mathcal{C}, 1 - R - \varepsilon, L_1, \ldots, L_n) \subseteq \mathcal{H}$.*

Recent advances have shown that the task of finding the subspace as above can be achieved in near-linear time!

Theorem 2.4 ([GHKS25])**.** *The affine space $\mathcal{H}$ from Theorem 2.3 for list recovery of $\mathsf{FRS}_{q,n,Rn,s}$ can be found in time $\widetilde{O}(n \cdot \log q \cdot \mathrm{poly}(\ell/\varepsilon, s))$ where $\widetilde{O}$ ignores $\mathrm{polylog}\, n, \mathrm{polylog} \log q$ terms.*

More formal versions of the above results are stated in Theorem 4.1.

FRS codes have been extremely useful in understanding the problems of list-recovery and list-decoding but the list-sizes in these results remained rather large with Theorem 2.3 giving us a list size of $q^{O(\ell/\varepsilon)}$ when list-recovering from a distance of $1 - R - \varepsilon$ and ℓ sized lists in each coordinate.

In a beautiful work, [KRSW23] proved a dramatic improvement in list-size bounds for list-decoding and list-recovery of FRS codes proving list-size bounds of $O_{\varepsilon,\ell}(1)$ i.e., constant when ε and ℓ are treated as constants. They also gave an efficient pruning algorithm which when combined with [GW13, GHKS25] gives an efficient algorithm to list-recover the entire list of codewords when $\ell, 1/\varepsilon$ are treated as constants. The bounds and algorithms were improved and simplified in the follow up work of [Tam24].

Theorem 2.5 ([KRSW23, Tam24])**.** *For every choice of $\ell \in \mathbb{N}, \varepsilon > 0, R > 0$, there exists an $s_0 = \Omega(\ell/\varepsilon^2)$, such that any rate R s-folded-Reed Solomon code $\mathcal{C}$ for $s > s_0$ and lists $L_1, \cdots, L_n$ with $|L_i| = \ell$, we have $|\mathsf{LIST}(\mathcal{C}, 1 - R - \varepsilon, L_1 \times \cdots \times L_n)| \leq (\ell/\varepsilon)^{O(\frac{1+\log \ell}{\varepsilon})}$.*

Additionally, there exists an efficient randomized algorithm PRUNE *that takes in an affine space $\mathcal{H}$ of dimension $r < \ell/\varepsilon$ and outputs*

$$\mathsf{LIST}(\mathcal{C}, 1 - R - \varepsilon, L_1 \times \ldots \times L_n) \cap \mathcal{H}$$

in time $\widetilde{O}(n \log q \cdot (\ell/\varepsilon)^{O(\frac{1+\log \ell}{\varepsilon})} \cdot \mathrm{poly}(s))$.

One interesting property about [KRSW23, Tam24] was that for the task of list-decoding i.e. if $\ell = 1$ then their algorithms worked for all linear codes given a low dimensional affine space. Thus, it remained open whether list bounds could be improved if the list-size bounds could be improved if one used any property of folded Reed-Solomon codes.

In a line of exciting works [Sri25, AHS25] proved that there are algorithms that output lists of size only $O(1/\varepsilon^2)$ when $\ell = 1$. [CZ25] proved that combinatorially it can be shown that folded Reed-Solomon codes only have list sizes of $O(1/\varepsilon)$.

For us, we build on the work of [AHS25], which we state here:

Theorem 2.6 ([AHS25])**.** *For every choice of $\varepsilon > 0, R > 0$, there exists an $s_0 = \Omega(1/\varepsilon^2)$, such that for any $s \geq s_0$ and any rate R s-folded Reed Solomon code $\mathcal{C}$, and received word $y \in \mathbb{F}^s$, it outputs a list L with $|L| \leq 1/\varepsilon^2$ such that:*

$$L = \mathsf{LIST}(\mathcal{C}, 1 - R - \varepsilon, y)$$

with high probability and runs in time $\widetilde{O}(n \cdot \log q \cdot \mathrm{poly}(s, \frac{1}{\varepsilon}))$.

2.3 Subspace-Design Codes

Subspace designs were introduced in [GX13] as a way to pre-code certain Reed-Solomon and algebraic-geometric codes so that they could then be list-decoded with small lists. Informally, a subspace design consists of a collection of subspaces $\mathcal{H}_i$ of some ambient space such that every low-dimensional space has non-trivial intersection with few of them. While [GX13] used random constructions of subspace designs, explicit constructions of subspace designs were given in [GK16]. Curiously, the construction was based on list-decodable codes such as folded RS and univariate multiplicity codes. Thus, subspace-designs were defined with a coding-theoretic motivation, and then themselves constructed using algebraic codes. Perhaps this was an early indication that they are natively a coding-theoretic concept, and recent results have cemented their fundamental role in coding theory [Sri25, CZ25, AHS25, BCDZ25b, BCDZ25a, GG25].

We present the definition of subspace-design codes in the terminology of [GG25]. A similar concept was also defined in [CZ25] where they called them subspace-designable codes.

Definition 6 (Subspace-Design Property)**.** For any function $\tau : \mathbb{N} \to \mathbb{R}_{\leq 1}$, an $\mathbb{F}_q$-additive code $\mathcal{C} \subseteq (\mathbb{F}_q^s)^n$ is said to be a τ-*subspace design* code if for every $r \in \mathbb{N}$, and every $\mathbb{F}_q$-linear subspace $\mathcal{A}$ of $\mathcal{C}$ of dimension at most r, the following holds:

$$\frac{\sum_{i=1}^{n} \dim \mathcal{A}_i}{n} \leq \dim(\mathcal{A}) \cdot \tau(r)$$

where $\mathcal{A}_i = \{a \in \mathcal{A} \mid a_i = 0\}$.

Proposition 2.7. *We can assume that if an additive code is a τ-subspace design code, then τ is a non-decreasing function without loss of generality.*

Proof. Note that the guarantee for $\tau(r)$ holds for all subspaces of dimension at most r. Thus, even if we define $\tau'(r) \leftarrow \min_{r' \geq r} \tau(r')$ then the code is also a τ'-subspace design which is non-decreasing and the guarantees only improve. $\square$

Lemma 2.8. *For any τ-subspace-design $\mathbb{F}_q$-additive code of rate R, we must have $\tau(r) \geq R - \frac{1}{n}$ for all $r \in \mathbb{N}$.*

Proof. We just need to prove the result for $r = 1$ since we can just take $\mathcal{A}$ of dimension at 1. Pick any non-zero codeword c which is zero on at least $Rn - 1$ coordinates. Such a codeword exists since the code is additive and there are $|\Sigma|^{Rn}$ different codewords. Now, define the 1-dimensional subspace $\mathcal{A} = \{\alpha \cdot c \mid \alpha \in \mathbb{F}_q\}$. Clearly, $\dim \mathcal{A}_i = 1$ for at least $R - \frac{1}{n}$ fraction of coordinates and so $\tau(1) \geq R - 1/n$. $\square$

The subspace design property has proven to be extremely useful in recent times and as is turns out is at the heart of the stronger list size-bounds shown for list-decoding of folded Reed-Solomon codes and univariate multiplicity codes established in [Sri25, CZ25, AHS25].

Theorem 2.9 ([GK16])**.** *s-folded Reed-Solomon codes are τ-subspace-design codes for $\tau(r) = \frac{sR}{s-r+1}$ for all $1 \leq r \leq s$ and $\tau(r) = 1$ otherwise.*

Thus, it can be shown that almost all results proven previously can be shown for all subspace-design codes.

One key property about subspace design codes is that all close-by codewords to any input word lie in a low-dimensional space. Specifically, we have the following trade-off between decoding radius and dimension. The following can be thought of as extending Theorem 2.3 to all subspace-design codes.

Lemma 2.10. *For every τ-subspace-design additive code $\mathcal{C} \subseteq \Sigma^n$, every positive integer r, and all input lists $L_1, \cdots, L_n \in \Sigma^n$ with $|L_i| \leq \ell$, the set*

$$\mathsf{LIST}(\mathcal{C}, 1 - \tau(r) - \frac{\ell}{r}, L_1 \times L_t \times \cdots \times L_n)$$

is contained within a linear space of dimension less than r.

The proof of the following for $\ell = 1$ is implicit in [CZ25] and a more general version is proved in [GG25].

Proof. If $\dim \mathcal{C} < r$ there is nothing to prove. Otherwise, consider an arbitrary set of r linearly independent $c^{(1)}, \cdots, c^{(r)}$ codewords in $\mathcal{C}$.

Define $\mathcal{A} = \mathsf{span}(c^{(1)}, \cdots, c^{(r)})$. For each $j \in [n]$, let $\mathcal{A}_j = \{c \in \mathcal{A} \mid c_j \in L_j\}$. We now observe that

$$n\left(r \cdot \tau(r) + \ell\right) \geq \sum_{j=1}^{n} (\dim \mathcal{A}_j) \geq \sum_{j=1}^{n} |\{i \in [r] \mid c_j^{(i)} = y_j\}| = n \cdot \sum_{i=1}^{r} (1 - \Delta(y, c^{(i)}))\ ,$$

where the first inequality follows from the subspace design property of $\mathcal{C}$. By averaging, it follows that $\Delta(y, c^{(i)}) \geq 1 - \frac{1}{r} - \tau(r)$ for some $i \in [r]$. Thus, not all the codewords $c^{(i)}$, $i = 1, 2, \ldots, r$, can satisfy $\Delta(c^{(i)}, y) < 1 - \frac{1}{r} - \tau(r)$. $\square$

Remark 1. Observe that the above proofs naturally extend to the *average-radius list-recovery* as well.

In light of the above results, we can informally think of the list-recovery or list-decoding of τ-subspace design codes as proceeding in two modular steps:

Step 1: Given the code $\mathcal{C}$ and lists $L_1, \cdots, L_n$, find a low dimensional affine space $\mathcal{H}$ which contains all codewords close to the list-recovery input product space.

Step 2: Find all the codewords inside a low dimensional affine space which are actually close to the list-recovery input product space.

The focus of [KRSW23, Tam24, AHS25] has been on performing the second step of this algorithm efficiently. Our current paper also focuses on getting structural results and efficient algorithms for step 2 for all subspace-design codes.

One would expect that Step 1 should in general be hard algorithmically for list-decoding or list-recovering arbitrary subspace-design codes. In contrast, we have explicit codes like FRS and UMULT for which step 1 can be done efficiently. As mentioned previously, this has been the focus of works such as [GW13, GHKS24, GHKS25].

3 Structure Theorems and Algorithms

Before beginning our algorithm, we will review the previous works.

3.1 A review of recent subspace-pruning algorithms for list-decoding

As a warmup and review, we consider the following two algorithms which were proposed for list-decoding of Folded Reed-Solomon and Multiplicity codes but in fact work more generally for any $\tau-$subspace design codes. Both the algorithms take in an input affine space $\mathcal{H}$ of dimension r and *prune* this affine space until they are left with a single codeword. The algorithms have the broad structure outlined in Algorithm 1.

Algorithm 1: $\mathsf{PRUNE}_{\mathcal{D}}(\mathcal{H}, y)$: Pruning a linear space

```
1 Input: An $\mathbb{F}_q$-affine space $\mathcal{H}$, a received word $y \in \mathbb{F}_q^n$.
2 If $\dim \mathcal{H} = 0$, output the only element in $\mathcal{H}$.
3 Else, sample $i \sim \mathcal{D}_{\mathcal{H},y}$ where $\mathcal{D}_{\mathcal{H},y}$ is a distribution on $[n]$ and return $\mathsf{PRUNE}_{\mathcal{D}}(\mathcal{H}_i, y)$
    where $\mathcal{H}_i = \{h \in \mathcal{H} \mid h_i = y_i\}$.
```

The key method of the pruning algorithm is now in the choice of this distribution $\mathcal{D}$. This is where the algorithms of [KRSW23, Tam24] differ from the algorithm of [AHS25]. The former works use a uniform weighting on $[n]$, whereas [AHS25] uses a clever non-uniform distribution tailored to the guarantee offered by the subspace-design property which depends on the affine space being pruned. We begin with the analysis of the uniform pruning algorithm.

Theorem 3.1 ([KRSW23, Tam24]). *Let $\mathcal{C} \subseteq \Sigma^n$ be an additive code with relative distance $\Delta(\mathcal{C})$, let $\mathcal{H} \subseteq C$ be an affine space of dimension r, and let received word $y \in \Sigma^n$. Then for any, $c \in \mathcal{C} \cap \mathcal{H}$ such that $\Delta(c, y) < \Delta(\mathcal{C}) - \varepsilon$, the algorithm $\mathsf{PRUNE}_{\mathcal{U}_n}(\mathcal{H}, y)$ outputs c with probability at least ε^r where $\mathcal{U}_n$ is the uniform distribution on $[n]$.*

Proof. Consider any $c' \neq c$ in $\mathcal{H}$, then observe that

$$\Pr_{i \sim \mathcal{U}_n}[c_i = y_i \wedge c'_i \neq y_i] \geq \varepsilon$$

since c_i and c'_i differ in at least $\Delta(\mathcal{C})$ fraction of positions i, but c and y have relative Hamming distance at $\Delta(\mathcal{C}) - \varepsilon$. Therefore, since we pick such an i with probability $\geq \varepsilon$, c remains in the resulting affine space by pruning and the dimension also reduces.

Now let $p_{c,r,y}$ be the minimum probability of $\mathsf{PRUNE}_{\mathcal{U}_n}$ outputting c across all $\mathcal{H} \subseteq \mathcal{C}$ of dimension at most r which contain c. From the above, we have

$$p_{c,r,y} \geq \varepsilon \cdot p_{c,r-1,y} \geq \cdots \geq \varepsilon^r p_{c,0,r} = \varepsilon^r$$

as desired. □

[AHS25] exploit the subspace design property of FRS and UMULT codes to get better results. We state their algorithmic guarantee in the more general setting of subspace-design codes below. Our presentation of the results also varies slightly from [AHS25] but the crux remains the same.

Theorem 3.2. *Let $\mathcal{C}$ be a τ-subspace design additive code. Let $\mathcal{H} \subseteq C$ be an affine space of dimension r, and let received word $y \in \Sigma^n$. Then for any, $c \in \mathcal{C} \cap \mathcal{H}$ such that $\Delta(c, y) < 1 - \tau(r) - \varepsilon$, the algorithm $\mathsf{PRUNE}_{\mathcal{D}_\varepsilon}(\mathcal{H}, y)$ outputs c with probability at least $\frac{\varepsilon}{r+\varepsilon}$ where $\mathcal{D}_\varepsilon(\mathcal{H}, y)$ samples coordinate i with probability $p_i = \frac{w_i}{\sum_{j=1}^n w_j}$ where:*

$$w_i = \begin{cases} 0 & \mathcal{H}_i = \emptyset \\ 0 & \mathcal{H}_i = \mathcal{H} \\ \dim \mathcal{H}_i + \varepsilon & \textit{otherwise} \end{cases}$$

Here, as above, $\mathcal{H}_i = \{h \in \mathcal{H} \mid h_i = y_i\}$.

Proof. Before beginning, we observe that it's not possible that all w_i are 0 as that would contradict the subspace design property as the set of coordinates where $c_i = y_i$ is at least a $\tau(r) + \varepsilon$ fraction.

As with [AHS25], we prove the result by induction on r.

The result clearly holds when $r = 0$. Now, assume that the result holds for all $r' < r$. Then, if we let $\mathsf{p}_{\mathcal{H},c,y}$ be the probability that $\mathsf{PRUNE}_{\mathcal{D}_\varepsilon}(\mathcal{H}, y)$ outputs c, we have:

$$\mathsf{p}_{\mathcal{H},c,y} = \sum_{i: y_i = c_i} p_i \cdot \mathsf{p}_{\mathcal{H}_i,c,y} \overset{\text{induction}}{\geq} \sum_{i: y_i = c_i, \mathcal{H}_i \neq \mathcal{H}} \frac{w_i}{\sum_{j=1}^n w_j} \cdot \frac{\varepsilon}{w_i} \geq \varepsilon \cdot \sum_{i: y_i = c_i, \mathcal{H}_i \neq \mathcal{H}} \frac{1}{\sum_{j=1}^n w_j}$$

Now, if α fraction of the i's have the property that $\mathcal{H}_i = \mathcal{H}$, then we have

$$\mathsf{p}_{\mathcal{H},c,y} \geq \varepsilon \cdot \sum_{i:y_i=c_i,\mathcal{H}_i\neq\mathcal{H}} \frac{1}{\sum_{j=1}^n w_j} \geq \varepsilon \cdot \frac{1-\Delta(c,y)-\alpha}{\frac{\sum_{j=1}^n(\dim \mathcal{H}_i+\varepsilon)}{n} - (r+\varepsilon)\alpha} \geq \varepsilon \cdot \frac{\tau(r)+\varepsilon-\alpha}{\frac{\sum_{j=1}^n \dim \mathcal{H}_i}{n} + \varepsilon - \alpha(r+\varepsilon)}$$

Now, we use the subspace-design property (Definition 6) to conclude

$$\mathsf{p}_{\mathcal{H},c,y} \geq \varepsilon \cdot \frac{\tau(r)+\varepsilon-\alpha}{r\cdot\tau(r)+\varepsilon-\alpha(r+\varepsilon)} \geq \varepsilon \cdot \frac{\tau(r)+\varepsilon-\alpha}{(r+\varepsilon)\cdot\tau(r)+\varepsilon\cdot(r+\varepsilon)-\alpha(r+\varepsilon)} = \frac{\varepsilon}{r+\varepsilon} ,$$

which concludes the proof. □

3.2 Our pruning algorithm for list-recovery

3.2.1 Discussion on approaches

The algorithm of [AHS25] is a clever improvement over the methods of [KRSW23, Tam24] with better weighting. Now, one can try generalizing this approach to list-recovery. A natural first attempt is as follows:

- We do not just sample $i \in [n]$ but also an element $j \in [\ell]$ where ℓ is the size of the lists in each coordinate.
- We restrict ourselves then to the affine subspace of $\mathcal{H}$ agreeing with the i'th coordinate's j'th list-element.

This approach is in fact taken in [Tam24]. He argues that for generic $\mathbb{F}_q$-linear codes, this approach yields a list-size bound of at most $\left(\frac{\ell}{\varepsilon}\right)^r$ for decoding radius $1-\tau(\ell/\varepsilon)-\varepsilon$. For τ-subspace design codes, one can get a list size of at most $\left(\frac{\ell}{\varepsilon}\right)^{O\left(\frac{1+\log\ell}{\varepsilon}\right)}$ for the same decoding radius. Thus, it is natural to ask whether the cleverer sampling of [AHS25] could buy us improvements in list-size bounds. Even if it does, we seem to be stuck with an exponential time since we have a list-size lower bound of $\ell^{\Omega\left(\frac{R}{\varepsilon}\right)}$ by [CZ25, LS25]. We make the following key observations:

- The above pruning algorithms can be made to be *independent* of the received codeword; let $\mathcal{H}$ always be a linear space and $\mathcal{H}_i = \{h \in \mathcal{H} \mid h_i = 0\}$. At the end, if our algorithm pin a set of coordinates S, then we can output a codeword $c \in \mathcal{H}$ such that c agrees with y on all coordinates in S. By our pruning guarantee, there is at most one such codeword. We show that this algorithm also offers the same guarantees proved above for list-decoding. This observation was also exploited in [GG25] to get optimal proximity gaps for subspace-design codes.
- Since we are now only computing a set of coordinates to pin on, but always set the value on the coordinate to be 0, we may get many codewords within the lists in one go.
- We can make the pruning in [AHS25] much more aggressive, only pinning on coordinates i in which the dimensions of the subspaces $\mathcal{H}_i$ are a fair bit smaller (in relation to the dimension of $\mathcal{H}$). This can ensure that we reducing to a zero-dimensional space much faster.

Combining these ideas, we improve the list size bounds to roughly $\left(\frac{r+\varepsilon}{\varepsilon}\right) \cdot \ell^{O(\frac{1+\log \ell}{\varepsilon})}$. A more formal theorem is given in Theorem 3.6.

Our actual algorithm will in fact produce a succinct description of a list of size $\mathrm{poly}(\ell/\varepsilon) \cdot \ell^{O(\frac{1+\log \ell}{\varepsilon})}$ which contains $\mathsf{LIST}(\mathcal{C}, 1-R-\varepsilon, L_1 \times \cdots \times L_n)$. For a more formal result, check Theorem 3.9.

3.2.2 Our pruning algorithm

Definition 7. The weight function wt_η of a linear space $\mathcal{H}$ is given by $\mathsf{wt}_\eta(\mathcal{H}) = \dim(\mathcal{H}) + \eta$.

Proposition 3.3 (Subspace Design property for weights of spaces)**.** *Let $\mathcal{C} \subseteq \Sigma^n$ be a τ-subspace design $\mathbb{F}_q$-additive code and let $\eta > 0$. Let $A \subseteq \mathcal{C}$ be any $\mathbb{F}_q$-linear space of dimension $r > 0$. Now, if $\mathcal{A}_i = \{a \in \mathcal{A} \mid a_i = 0\}$ then:*

$$\frac{\sum_{i=1}^{n} \mathsf{wt}_\eta(A_i)}{n} \leq \mathsf{wt}_\eta(\mathsf{A}) \cdot (\tau(r) + \eta)$$

Proof. Indeed, we have

$$\frac{\sum_{i=1}^{n} \mathsf{wt}_\eta(A_i)}{n} = \frac{\sum_{i=1}^{n} \dim(A_i)}{n} + \eta \leq r \cdot \tau(r) + \eta \leq \mathsf{wt}_\eta(\mathcal{A}) \cdot (\tau(r) + \eta) \ .$$

The first step is by definition, the second is by the subspace-design property (Definition 6) and the final step holds since $\mathsf{wt}_\eta(\mathcal{A}) \geq r \geq 1$. We even had a similar result used inside the proof of Theorem 3.2 above. □

Algorithm 2 takes in an input as a linear space $\mathcal{H} \subseteq (\mathbb{F}_q^s)^n$ and outputs a set $T \subseteq [n]$ such that $|T| \leq \dim |\mathcal{H}|$.

Algorithm 2: $\mathsf{FPRUNE}(\mathcal{H}, \eta, \eta')$: Pruning a linear space

```
1 Input: An $\mathbb{F}_q$-linear space $\mathcal{H}$ and a real $\eta > 0$. Let $\mathcal{H}_i = \{c \in \mathcal{H} \mid c_i = 0\}$
2 If $\mathcal{H} = \{0\}$. Output $\Phi$. Sample $i$ from $[n]$ with probability $p_i$ where
      $p_i = \begin{cases} 0 & \mathsf{wt}_\eta(\mathcal{H}_i) > (1-\eta')\mathsf{wt}_\eta(\mathcal{H}) \\ \frac{\mathsf{wt}_\eta(\mathcal{H}_i)}{\sum_{i: \frac{\mathsf{wt}_\eta(\mathcal{H}_i)}{1-\eta'} \leq \mathsf{wt}_\eta(\mathcal{H})} \mathsf{wt}_\eta(\mathcal{H}_i)} & \text{otherwise} \end{cases}$
   Return $\{i\} \cup \mathsf{FPRUNE}(\mathcal{H}_i, \eta, \eta')$.
```

Towards analyzing the performance of Algorithm 2 we introduce the following potential function.

Definition 8. Let $\Sigma = \mathbb{F}_q^s$ for some finite field $\mathbb{F}_q$ and integer $s \geq 1$. For any $\mathbb{F}_q$-linear space $\mathcal{H} \subseteq \Sigma^n$, a vector $c \in \Sigma^n$, lists $L_1, \cdots, L_n \subset \Sigma$ and a set $T \subseteq [n]$ with the property that $\mathcal{H}$ is 0 on all coordinates in T, we define the following function:

$$f_{\eta,\eta'}(\mathcal{H}, c, T, L_1, \cdots, L_n) = \begin{cases} 0 & c_i \notin L_i \text{ for some } i \in T \\ \frac{(1-\eta')^{|T|}}{\text{wt}_\eta(\mathcal{H})} & \end{cases} .$$

We now provide a slightly stronger version of the Theorem 3.2 lemma ensuring dimension reduction in every step. We note a few changes:

- Our pruning algorithm outputs only a set of *agreement coordinates* and this step is independent of the lists in each coordinate after receiving the space $\mathcal{H}$. Thus, it is possible that our algorithm fixes sets of coordinates where there is no $c \in \mathcal{H}$ that agrees on these coordinates with any word.
- We have a faster dimension reduction and the associated guarantee that dimensions are much smaller each time.
- We prove the result using a *monovariant* instead of induction to emphasise the *top-down* nature of the pruning algorithm and a quantity that improves in each step. In contrast, [AHS25]'s proof is inductive. Our proof method also directly applies to the original [AHS25] lemma. A version of this theorem with $\eta' = 0$ appeared in [GG25].

Lemma 3.4 (Stronger main [AHS25] lemma)**.** *Let $\Sigma = \mathbb{F}_q^s$ for some finite field $\mathbb{F}_q$ and integer $s \geq 1$. Let r be a positive integer and let $\eta > 0$. Suppose we are given a τ-subspace-design code $\mathcal{C} \subseteq \Sigma^n$, input lists $L_1, \cdots, L_n \subseteq \Sigma$ (no restrictions on the sizes of these lists or what they look like), and a $\mathbb{F}_q$-linear space $\mathcal{H} \subset \Sigma^n$ of dimension r, and a subset $T \subseteq [n]$ so that $\mathcal{H}$ is identically 0 on all of T.*

Suppose we sample a coordinate $i \in [n]$ with probability p_i where

$$p_i = \begin{cases} 0 & \mathsf{wt}_\eta(\mathcal{H}_i) > (1-\eta')\mathsf{wt}_\eta(\mathcal{H}) \\ \dfrac{\mathsf{wt}_\eta(\mathcal{H}_i)}{\sum\limits_{i:\frac{\mathsf{wt}_\eta(\mathcal{H}_i)}{1-\eta'} \leq \mathsf{wt}_\eta(\mathcal{H})} \mathsf{wt}_\eta(\mathcal{H}_i)} & \textit{otherwise} \end{cases} .$$

Then, for any $c \in \mathcal{H}$ satisfying

$$\Delta(c, L_1 \times L_2 \times \cdots \times L_n) < 1 - \frac{\tau(r) + \eta}{1 - \eta'} , \tag{1}$$

the following monotonicity holds:

$$\mathbb{E}[f_{\eta,\eta'}(\mathcal{H}_i, c, T \cup \{i\}, L_1, \cdots, L_n)] \geq f_{\eta,\eta'}(\mathcal{H}, c, T, L_1, \cdots, L_n) .$$

Proof. We drop the $L_1, \cdots, L_n$ from the function definition for convenience since it is fixed across the proof anyway.

Observe that if $f_{\eta,\eta'}(\mathcal{H}, c, T) = 0$ then there is nothing to prove. So we assume that $f_{\eta,\eta'}(\mathcal{H}, c, T) \neq 0$. Let

$$w_i = \text{wt}_\eta(\mathcal{H}_i), \quad w = \text{wt}_\eta(\mathcal{H}), \text{ and } W = \sum_{j: w_j \leq (1-\eta')w} w_j .$$

Suppose that αn of the coordinates satisfy $\mathsf{wt}_\eta(\mathcal{H}_i) > (1-\eta')\mathsf{wt}_\eta(\mathcal{H})$. By the τ-subspace-design property of code, we have

$$W \le \big(\tau(r) + \eta - \alpha(1-\eta')\big)wn \tag{2}$$

We have

$$\begin{aligned}
\mathbb{E}[f_{\eta,\eta'}(\mathcal{H}_i, c, T \cup \{i\})] &= \Big(\sum_{i:c_i\in L_i} p_i \cdot f_{\eta,\eta'}(\mathcal{H}, c, T)\cdot \frac{w}{w_i}\Big)(1-\eta') \\
&= (1-\eta')f_{\eta,\eta'}(\mathcal{H},c,T)\cdot \sum_{i:c_i\in L_i,\ w_i\le(1-\eta')w} \frac{w_i}{W}\cdot\frac{w}{w_i} \\
&= (1-\eta')f_{\eta,\eta'}(\mathcal{H},c,T) \sum_{i:c_i\in L_i,\ w_i\le(1-\eta')w} \frac{w}{W} \\
&\ge (1-\eta')f_{\eta,\eta'}(\mathcal{H},c,T)\frac{w}{W}\big(|\{i : c_i \in L_i\}| - \alpha n\big)\ .
\end{aligned}$$

Now by hypothesis (1) we have $\frac{|\{i:c_i\in L_i\}|}{n} \ge \frac{\tau(r)+\eta}{1-\eta'}$, and we also have the upper bound (2) on W. Therefore we can bound

$$\begin{aligned}
\mathbb{E}[f_{\eta,\eta'}(\mathcal{H}_i, c, T\cup\{i\})] &\ge (1-\eta')\frac{\frac{\tau(r)+\eta}{1-\eta'} - \alpha}{\tau(r)+\eta-\alpha(1-\eta')}\cdot f_{\eta,\eta'}(\mathcal{H},c,T) \\
&= f_{\eta,\eta'}(\mathcal{H},c,T)\ ,
\end{aligned}$$

completing the proof. □

Lemma 3.5. *Let r be a positive integer and let $\eta > 0$. Suppose we are given τ-subspace design code $\mathcal{C} \subseteq \Sigma^n$, lists $L_1, \cdots, L_n \subseteq \Sigma$ (no restrictions on the sizes of these lists or what they look like), and a linear space $\mathcal{H}$ of dimension r. Then for any fixed $c \in \mathcal{H}$ satisfying $\Delta(c, L_1\times L_2\times\cdots\times L_n) < 1 - \frac{\tau(r)+\eta}{1-\eta'}$, the following holds: if algorithm $\mathsf{FPRUNE}(\mathcal{H},\eta,\eta')$ returns a (random) set T, then we have*

$$\mathbb{E}[X_{c,T}\cdot(1-\eta')^{|T|}] \ge \frac{\eta}{r+\eta}$$

where $X_{c,T}$ is the indicator random variable for whether $c_i \in L_i$ for all $i \in T$.

Proof. At the end if the output of the algorithm is a set T, then we know that

$$\mathbb{E}[f_{\eta,\eta'}(\{0\}, c, T)] \ge f_{\eta,\eta'}(\mathcal{H}, c, \emptyset) = \frac{1}{r+\eta} \implies \mathbb{E}[X_{c,T}\cdot(1-\eta')^{|T|}] \ge \frac{\eta}{r+\eta}\ .$$ □

Also, observe that for any values $\beta_1, \beta_2, \cdots, \beta_{|T|}$, if $\mathcal{H}$ outputs $i_1, \cdots, i_{|T|}$, there is at most 1 element $c \in \mathcal{H}$ such that $c_{i_1} = \beta_1, c_{i_2} = \beta_2$ and so on.

Theorem 3.6. *For any integer $r > 0$ and real $\eta > 0$ given a τ-subspace design code $\mathcal{C} \subseteq (\Sigma)^n$, lists $L_1, \cdots, L_n \subseteq \Sigma$ and $|L_i| = \ell$, and linear space $\mathcal{H}$ of dimension r, we have*

$$\left|\left\{c \in \mathcal{H} \mid \Delta(c, L_1\times L_2\times\cdots\times L_n) < 1 - \frac{\tau(r)+\eta}{1-\eta'}\right\}\right| \le \ell^{\min(r, \frac{1}{\eta'}(1+\log(r\eta')))}\left(\frac{r}{\eta}+1\right)$$

Proof. We run the algorithm $\mathsf{FPRUNE}(\mathcal{H}, \eta, \eta')$ to get a set T. Now, if we consider all $c \in \mathcal{H}$ such that for all $j \in [T]$, we have $c_j \in L_j$, then we get a list LIST of codewords of at most $\ell^{|T|}$ codewords.

Note that $|T| \le r$ since the dimension always reduces by at least 1. Also, we have

$$|T| \le \left\lceil \frac{1}{\eta'}(1 + \log(r\eta')) \right\rceil$$

since in the first $\frac{1}{\eta'} \log(r\eta')$ steps, the dimension reduces to at most $1/\eta'$ as

$$r \cdot (1-\eta')^{\frac{1}{\eta'}\log(r\eta')} \le r \cdot \frac{1}{r\eta'} \le \frac{1}{\eta'} ,$$

and then after that still always reduces by at least 1 whenever an element is added to T.

Now, by Lemma 3.5, for each c such that $\Delta(c, L_1 \times L_2 \times \cdots \times L_n) < 1 - \frac{\tau(r)+\eta}{1-\eta'}$, we have

$$\Pr[c \in \mathsf{LIST}] \ge \frac{\eta}{r+\eta} .$$

Thus summing over all relevant codewords, we can conclude that

$$\ell^{\min(r, \frac{1}{\eta'}(1+\log(r\eta')))} \ge \mathbb{E}[|\mathsf{LIST}|] \ge \frac{\eta}{r+\eta} \cdot \left| \left\{ c \in \mathcal{H} \mid \Delta(c, L_1 \times L_2 \times \cdots \times L_n) < 1 - \frac{\tau(r)+\eta}{1-\eta'} \right\} \right| ,$$

which gives us the desired list-size bound. □

Corollary 3.7. *For any integer $r > 0$ and real $\eta > 0$ given a τ-subspace design code $\mathcal{C} \subseteq \Sigma^n$, lists $L_1, \cdots, L_n \subseteq \Sigma$ and $|L_i| = \ell$, and linear space $\mathcal{H}$ of dimension r, let $\mathsf{LIST} = \{c \in \mathcal{H} \mid \Delta(c, L_1 \times L_2 \times \cdots \times L_n) < 1 - \frac{\tau(r)+\eta}{1-\eta'}\}$.*

If we run $\mathsf{FPRUNE}(\mathcal{H}, \eta)$ a total of $t = \frac{r+\eta}{\eta} \cdot (r \log \ell + \log(r/\eta + 1) + t')$ times, then we get sets $T_1, \cdots, T_t$ such that

$$\Pr[\forall c \in \mathsf{LIST}\ \exists j \in [t] \text{ such that } \forall i \in T_j : c_i \in L_i] \ge 1 - e^{-t'}$$

Proof. Observe that for any fixed $c \in \mathsf{LIST}$, $\Pr[\nexists j \in [t]$ such that $\forall i \in T_j : c_i \in L_i] \le (1 - \frac{\eta}{r+\eta})^t \le e^{-\frac{\eta}{r+\eta}\cdot t} \le e^{-(\log|\mathsf{LIST}| + t')} \le \frac{e^{-t'}}{\mathsf{LIST}}$, where we have used the upper bound on $|\mathsf{LIST}|$ proved in Theorem 3.6.

Now, we just conclude by union bounding over all the codewords in LIST. □

This result tells us that we can get an efficient algorithm that outputs a description of few sets such that any relevant list-recovery codeword would be in all the lists of at least one of the output sets.

We would like to move away from this description of a few sets to a more natural description. An example of a nicer description would be not in terms of agreement coordinates, but rather inside a more explicit space of codewords. We now turn to providing such a description.

3.3 Sum-set structure from complete agreement

In the following subsection, we consider the sets output by the list recovery algorithm from above and show that the lists as described before have an additional *sum-set* structure.

The results in this subsection are in fact more generic linear algebraic claims and holds for all $\mathbb{F}_q$-additive codes. Its benefit for τ-subspace design codes only holds when combined with *Corollary* 3.7.

Let's begin by defining what we mean by a *sum-set*-structure.

Definition 9 (Sum Sets)**.** $P \subseteq \mathcal{C}$ is an (u, v) *sum-set* in $\mathcal{C}$ if it is described by sets $A_1, A_2, \cdots A_{u'}$ with $|A_i| \leq v$, $u' \leq u$ and $A_i \in \mathcal{C}$ such that $P = A_1 + \cdots + A_{u'} = \{a_1 + a_2 + \ldots + a_{u'} \mid a_i \in A_i\}$.

Our key theorem in this subsection is that there is a natural way of going between agreeing on a few coordinates to a *sum-set* structure.

Theorem 3.8. *Let $\mathcal{H} \subseteq (\mathbb{F}_q^s)^n$ be an r-dimensional $\mathbb{F}_q$-linear space and let $\{t_1, \cdots, t_m\} = T \subseteq [n]$ and $m \leq r$ be such that $\{c \in \mathcal{H} \mid c_t = 0\ \forall t \in T\} = \{0\}$. Then the following holds for all subsets $L_{t_1}, L_{t_2}, \cdots, L_{t_m} \subseteq \mathbb{F}_q^s$ with $|L_i| = \ell$:*

There exist sets $A_{t_i} \subseteq \mathcal{H}$ with $|A_{t_i}| \leq \ell$ for each $i \in [m]$ such that

$$\{c \in \mathcal{H} : c_t \in L_t\ \forall t \in T\} \subseteq A_{t_1} + A_{t_2} + \cdots A_{t_m} = \{a_1 + a_2 + \ldots + a_m \mid a_i \in A_{t_i}\}\ .$$

In fact, there is an algorithm REDUCE *which outputs $A_{t_1}, \cdots, A_{t_m}$ and runs in time $\widetilde{O}(n \cdot \ell \cdot \log q \cdot \mathrm{poly}(s, r))$.*

Proof. Observe that since $\{c \in \mathcal{H} \mid c_i = 0\ \forall i \in T\} = \{0\}$, we have reduced $\mathcal{H}$ from dimension r to dimension 0.

We can now view $\left(\mathbb{F}_q^s\right)^n$ as $\mathbb{F}_q^{sn}$ and restrict our view to the sm coordinates in T. Now, by performing Gaussian elimination, we can find r positions within these sm coordinates such that the values on them are linearly independent. We call these positions the *independence* positions.

Thus, we can let $T_1, \cdots, T_m$ be sets with each $T_i \subseteq [s]$ and $\sum |T_i| = r$ such that $\mathcal{H}$ restricted to $(T_1, \cdots, T_m)$ is full dimensional, i.e., the projection of $\mathcal{H}$ onto the coordinates determined by $T_1, \cdots, T_m$ has dimension r equal to the dimension of $\mathcal{H}$ itself.

Now, if we restrict ourselves not to being in all the m lists entirely - but only on being in the respective lists restricted to the coordinates of the T_is, we can only get a larger list.

We use this to create the set A_{t_i}.

$$A_{t_i} = \{c \in \mathcal{H} \mid c_{t_j|T_j} = 0\ \forall j \neq i, c_{t_i|T_i} \in L_{t_i|T_i}\}$$

Since, for all $\beta \in L_i$, there is a unique $c \in \mathcal{H}$ such that $c \in \mathcal{H} \mid c_{t_j|T_j} = 0 \forall j \neq i, c_{t_i|T_i} = \beta_{|T_i}$, we have that $|A_{t_i}| \leq \ell$ for each i.

Now, also for each $\beta_1 \in L_{t_1}, \beta_2 \in L_{t_2}, \cdots, \beta_m \in L_{t_m}$, there is a unique element a_{β_i} in A_{t_i} such that $a_{\beta_i|T_i} = \beta_{i|T_i}$. Thus, $a_{\vec{\beta}} = a_{\beta_1} + \cdots + a_{\beta_m}$ is the unique element in $\mathcal{H}$ such that

$$a_{\vec{\beta}_{|T_i}} = \beta_{i|T_i}$$

for all $i \in [m]$. Thus, defining the sets A_{t_i} as above works as desired.

This protocol is clearly algorithmic and once we restrict $\mathcal{H}$ to just the coordinates of T, Gaussian elimination to find *independence* positions runs in time $\widetilde{O}(\log q)\,\mathrm{poly}(r,s)$. Now, computing the sets A_{t_i} requires solving an r-dimensional linear system and we have $\ell \cdot m$ such computations to do. Solving each of these linear systems takes $\widetilde{O}(\log q \cdot \mathrm{poly}(r,s))$ time and the claimed result follows. □

3.4 Combining pruning and sum-set structure results

Theorem 3.9. *For any integer $r, s > 0$ and real $\eta > 0$ given a τ-subspace design code $\mathcal{C} \subseteq (\mathbb{F}_q^s)^n$, and lists $L_1, \cdots, L_n \subseteq \mathbb{F}_q^s$ and $|L_i| = \ell$, and a linear space $\mathcal{H}$ of dimension r, there is an algorithm running in time $\widetilde{O}(n \log q \cdot \mathrm{poly}(\ell, s, r, \frac{1}{\eta \cdot \eta'}))$, which outputs the descriptions of t many $(\min(r, \lceil \frac{1}{\eta'}(1 + \log(r\eta'))\rceil), \ell)$ sum-sets $P_1, \cdots, P_t$ such that:*

$$\Pr[\{c \in \mathcal{H} \mid \Delta(c, L_1 \times L_2 \times \cdots \times L_n) < 1 - \frac{\tau(r) + \eta}{1 - \eta'} \subseteq \bigcup_{j=1}^{t} P_j] \geq 1 - e^{-t'}$$

and runs in time $\widetilde{O}(n \cdot \log q \cdot \mathrm{poly}(\ell, r, s, \frac{1}{\eta}) \cdot t')$ where $t = \frac{r+\eta}{\eta} \cdot (r \log \ell + \log(r/\eta + 1) + t')$

Proof. Run Algorithm 2, $\mathsf{FPRUNE}(\mathcal{H}, \eta, \eta')$ a total of t times and for each of output sets T, invoke the algorithm REDUCE from Theorem 3.8. The success probability follows from Corollary 3.7. □

4 Efficient list-recovery of FRS codes

As we discussed earlier, the current approaches for list recovering or list decoding subspace-design codes can be broken into two modular steps:

- **Step 1:** Find a low dimensional affine space in which all close codewords lie.
- **Step 2:** Prune this space to be left with only a few codewords.

In general, step 1 might not be feasible for an arbitrary code, but due to Theorem 3.9 and Theorem 3.2, we know that step 2 can always be done efficiently. Thus, a natural question concerns the codes for which one can implement step 1 efficiently.

Indeed, our motivation to study these bounds was heavily inspired by FRS and UMULT codes which indeed do have efficient algorithms to achieve step 1.

Theorem 4.1 ([GW13, GHKS25]). *There exists a deterministic algorithm such that for any rate R s-folded Reed-Solomon code and input lists $L_1, \cdots, L_n \subseteq \mathbb{F}_q^s$ with $|L_i| = \ell$, and parameter $r \leq s$ outputs an affine space $\mathcal{A}$ with the following properties:*

- $\dim \mathcal{A} \leq r$
- $\mathsf{LIST}(\mathcal{C}, 1 - \frac{s}{s-r}\left(\frac{\ell}{r} + R\right), L_1 \times L_2 \times \cdots \times L_n) \subseteq \mathcal{A}$.
- *The algorithm runs in time $\widetilde{O}(n \cdot \log q \cdot \mathrm{poly}(\ell, s))$.*

The runtime $\widetilde{O}(n\cdot\log q\cdot\mathrm{poly}(\ell,s))$ is due to [GHKS25] and the original [GW13] has a runtime of $\mathrm{poly}(n,\log q,\ell,s)$.

Theorem 4.2. *For every choice of $\ell\in\mathbb{N},\varepsilon>0,R>0$, any rate R s-folded-Reed Solomon code $\mathcal{C}$ for $s>s_0:=\frac{16(R+\varepsilon)\ell}{\varepsilon^2}$ has the following property.*

There exists an algorithm that takes in input lists $L_1,\cdots,L_n\subseteq\mathbb{F}_q^s$ of size ℓ each, runs in time $\widetilde{O}(n\cdot\log q\cdot\mathrm{poly}(s))$, and outputs the descriptions of $t=\widetilde{O}(\ell^2/\varepsilon^3)$ many

$$\left(\left\lceil\frac{2(R+\varepsilon)(\log(2e\ell/(R+\varepsilon)))}{\varepsilon}\right\rceil,\ell\right)\ \textit{sum-sets}\ P_1,\cdots,P_t$$

such that $\mathsf{LIST}(\mathcal{C},1-R-\varepsilon,L_1\times L_2\times\cdots\times L_n)\subseteq\bigcup_{i=1}^{t}P_i$ with probability $\geq\frac{1}{2}$.

Proof. We apply Theorem 4.1 with parameters $r=\frac{4\ell}{\varepsilon}$, $s_0=\frac{16(R+\varepsilon)\ell}{\varepsilon^2}$. It asserts that there is an affine space $\mathcal{A}$ of dimension at most r containing

$$\mathsf{LIST}(\mathcal{C},1-\tfrac{s}{s-r}\big(\tfrac{\ell}{r}+R\big),L_1\times L_2\times\cdots\times L_n)\ .$$

By our choices of s_0,r, we get that $\frac{\ell}{r}=\frac{\varepsilon}{4}$ and

$$\frac{s}{s-r}\leq 1+\frac{r\left(\frac{4(R+\varepsilon)}{\varepsilon}\right)}{r\left(\frac{4(R+\varepsilon)}{\varepsilon}-1\right)}\leq\frac{4R+4\varepsilon}{4R+3\varepsilon}\leq\frac{4R+2\varepsilon}{4R+\varepsilon}=\frac{R+\varepsilon/2}{R+\varepsilon/4}$$

Thus, $\frac{s}{s-r}(\frac{\ell}{r}+R)\leq R+\varepsilon/2$. In particular,

$$\mathsf{LIST}(\mathcal{C},1-R-\varepsilon,L_1\times L_2\times\cdots\times L_n)\subseteq\mathcal{A}\ , \tag{3}$$

where $\mathcal{A}$ is at most $\frac{4\ell}{\varepsilon}$ dimensional. This affine space $\mathcal{A}$ is also output by the [GHKS25] algorithm.

Now, we can apply Theorem 3.9 with parameters $t'=1$, $\eta=\frac{\varepsilon}{4}$, $\eta'=\frac{\varepsilon}{2(R+\varepsilon)}$. This algorithm outputs $P_1,\cdots,P_t$ which are all $\left(\left\lceil\frac{(\log(er\eta'))}{\eta'}\right\rceil,\ell\right)$ *sum-sets* such that

$$\Pr[\{c\in\mathcal{A}\mid\Delta(c,L_1\times L_2\times\cdots\times L_n)<1-\frac{\tau(r)+\eta}{1-\eta'}\}\subseteq\bigcup_{j=1}^{t}P_j]\geq 1-e^{-t'}\geq 1/2\ . \tag{4}$$

Note that Theorem 3.9 works with linear spaces and not affine space. Observe that this is easy to fix as we can translate the entire problem i.e. all the input lists and $\mathcal{A}$ by one of the elements of the affine space $\mathcal{A}$. This results in $\mathcal{A}$ becoming a subspace. This element can also be added back to each *sum-set* outputted by the Theorem 3.9 algorithm to get *sum-sets* that work for the original untranslated problem.

Observe that again by our setting of parameters, we have that $\tau(r)\leq R\cdot\frac{s}{s-r+1}<R\cdot\frac{s}{s-r}\leq R+\varepsilon/4$, $\eta\leq\varepsilon/4$, and thus

$$\frac{\tau(r)+\eta}{1-\eta'}<\frac{R+\varepsilon/2}{(2R+\varepsilon)/(2R+2\varepsilon)}=R+\varepsilon\ . \tag{5}$$

Combining (3), (4), and (5), we conclude

$$\Pr[\mathsf{LIST}(\mathcal{C},1-R-\varepsilon,L_1\times L_2\times\cdots\times L_n)\subseteq\bigcup_{j=1}^{t}P_j]\geq 1/2\ .$$

Additionally, we have that

$$\left\lceil \frac{(\log(er\eta'))}{\eta'} \right\rceil = \left\lceil \frac{2(R+\varepsilon)\log(2e\ell/(R+\varepsilon))}{\varepsilon} \right\rceil$$

as desired and $t = \lceil \frac{r+\eta}{\eta}\left(r\log\ell + (\log(r/\eta+1)) + 1\right)\rceil = \widetilde{O}(\ell^2/\varepsilon^3)$. ☐

We also get the following theorem:

Theorem 4.3. *For every choice of $\ell \in \mathbb{N}$, $\varepsilon > 0$, $R > 0$, $s_0 = \frac{16(R+\varepsilon)\ell}{\varepsilon^2}$, such that for any rate R s folded Reed-Solomon code $\mathcal{C}$ with $s > s_0$, we have that*

$$|\mathsf{LIST}(\mathcal{C}, 1-R-\varepsilon, L_1 \times L_2 \times \cdots \times L_n)| \leq O\left(\frac{\ell}{\varepsilon^2}\right) \cdot \ell^{2(\ln(2e\ell/(R+\varepsilon))\cdot\frac{R+\varepsilon}{\varepsilon}} .$$

Proof. By combining the choice of parameters described above with Theorem 4.1 and Theorem 3.6, we get that $|\mathsf{LIST}| \leq \ell^{\frac{1}{\eta'}\ln(er\eta')} \cdot (r/\eta + 1)$ where $r = 4\ell/\varepsilon$, $\eta = \varepsilon/4$, $\varepsilon/2(R+\varepsilon)$.

This gives us a bound of

$$|\mathsf{LIST}| \leq \left(16\ell/\varepsilon^2 + 4/\varepsilon\right) \cdot \ell^{\frac{2(R+\varepsilon)}{\varepsilon}\cdot\ln(2e\ell/(R+\varepsilon))} .$$ ☐

5 Open Questions and Conjectures

In recent work, [BCDZ25a] prove stronger combinatorial list-recovery bounds by insightfully connecting the subspace-design property to Brascamp-Lieb inequalities. [1]

Their list size bounds are significantly better than what we are able to achieve using our pruning techniques. Below we state their result in our terminology of subspace-design codes, and for completeness, in Appendix A, we present their proof.

Theorem 5.1 ([BCDZ25a] List-Recovery Size Bounds)**.** *If $\mathcal{C} \subseteq \Sigma^n$ is a τ-subspace design code, then for any linear space $\mathcal{H}$ of dimension r, and subsets $L_1, \cdots, L_n \subseteq \Sigma$ with each $|L_i| = \ell$, and any $\varepsilon > 0$, we have*

$$|\{c \in \mathcal{H} \mid \Delta(c, L_1 \times L_2 \times \cdots \times L_n) < 1 - \tau(r) - \varepsilon\}| \leq (\ell/(\tau(r)+\varepsilon))^{\frac{\tau(r)+\varepsilon}{\varepsilon}} .$$

This raises the question whether we can use our algorithmic techniques in combination with their ideas to prune the lists further? Along these lines, we make the following conjecture.

Conjecture 5.2. *For every choice of $\ell \in \mathbb{N}, \varepsilon > 0, R > 0$, there exists an $s_0 = \Omega(\ell/\varepsilon^2)$, such that any rate R s-folded Reed-Solomon code $\mathcal{C}$ for $s > s_0$, there exist* $\mathrm{poly}(\ell/\varepsilon)$ *many $\left(O\left(\frac{R+\varepsilon}{\varepsilon}\right), \ell\right)$ sum-sets $P_1, \cdots, P_t$ such that*

$$\{c \in \mathcal{C} \mid \Delta(c, L_1 \times L_2 \times \cdots \times L_n) < 1 - R - \varepsilon\} = \bigcup_{i=1}^{t} P_i.$$

[1] We note that our results were obtained before [BCDZ25a] was posted.

This conjecture would give a combinatorial list-recovery bound of $\mathrm{poly}(\ell/\varepsilon) \cdot \ell^{O\left(\frac{R}{\varepsilon}\right)}$ which we know to be optimal due to [CZ25] and [LS25]. Note that the above form of the conjecture is quite strong as it states that the list is exactly a union of a few low-dimensional sum-sets and not just contained within such a union.

It is also natural to make the same conjecture for random linear codes and random Reed-Solomon codes since we now know matching list recovery size bounds from [LMS25, BCDZ25b, LS25].

Our algorithm is randomized and derandomizing it is another interesting direction.

Question 5.3. *(Informal) Can the above pruning algorithms be made deterministic while also running in* $\mathrm{poly}(\ell, 1/\varepsilon)$ *time?*

[AHS25] show that the pruning for list-decoding can indeed be implemented with deterministic time complexity $\widetilde{O}(n) \cdot (1/\varepsilon)^{O(2^{1/\varepsilon})}$. It would be interesting to improve this complexity or to even match it for list-recovery.

6 Acknowledgements

We thank Joshua Brakensiek for useful discussions and feedback.

V.G. is supported by a Simons Investigator award, NSF grant CCF-2211972, and ONR grant N00014-24-1-2491.

R.G. is supported by (Yael Tauman Kalai's) grant from Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-25-C-0300. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

Part of this work was carried out while R.G. was visiting the Simons Institute for the Theory of Computing.

References

[AGG$^+$25] Omar Alrabiah, Zeyu Guo, Venkatesan Guruswami, Ray Li, and Zihan Zhang. Random Reed-Solomon codes achieve list-decoding capacity with linear-sized alphabets. *Advances in Combinatorics*, October 2025. 4

[AHS25] Vikrant Ashvinkumar, Mursalin Habib, and Shashank Srivastava. Algorithmic improvements to list decoding of folded Reed-Solomon codes. *arXiv preprint arXiv:2508.12548*, 2025. To appear in SODA 2026. 4, 5, 8, 9, 10, 11, 12, 14, 21

[BCDZ25a] Joshua Brakensiek, Yeyuan Chen, Manik Dhar, and Zihan Zhang. Combinatorial bounds for list recovery via discrete Brascamp–Lieb inequalities. *arXiv preprint arXiv:2510.13775*, 2025. 4, 5, 8, 20, 24

[BCDZ25b] Joshua Brakensiek, Yeyuan Chen, Manik Dhar, and Zihan Zhang. From random to explicit via subspace designs with applications to local properties and matroids. *arXiv preprint arXiv:2510.13777*, 2025. 5, 8, 21

[CCE09] Eric A. Carlen and Dario Cordero-Erausquin. Subadditivity of the entropy and its relation to Brascamp–Lieb type inequalities. *Geometric and Functional Analysis*, 19(2):373–405, 2009. 24

[CDK$^+$13] Michael Christ, James Demmel, Nicholas Knight, Thomas Scanlon, and Katherine Yelick. Communication lower bounds and optimal algorithms for programs that reference arrays – Part 1. *arXiv preprint arXiv:1308.0068*, 2013. 24

[CDK$^+$24] Michael Christ, James Demmel, Nicholas Knight, Thomas Scanlon, and Katherine A. Yelick. On multilinear inequalities of Holder-Brascamp-Lieb type for torsion-free discrete Abelian groups. *J. Log. Anal.*, 16, 2024. 24

[CHKV11] Mahdi Cheraghchi, Ali Hormati, Amin Karbasi, and Martin Vetterli. Group testing with probabilistic tests: Theory, design and application. *IEEE Transactions on Information Theory*, 57(10):7057–7067, October 2011. 3

[CT25] André Chailloux and Jean-Pierre Tillich. Quantum advantage from soft decoders. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC '25, page 738–749, New York, NY, USA, 2025. Association for Computing Machinery. 3

[CZ25] Yeyuan Chen and Zihan Zhang. Explicit folded Reed-Solomon and multiplicity codes achieve relaxed generalized singleton bound. In Michal Koucký and Nikhil Bansal, editors, *Proc. 57th ACM Symp. on Theory of Computing (STOC)*, pages 1–12, 2025. 4, 6, 8, 9, 12, 21

[DL12] Zeev Dvir and Shachar Lovett. Subspace evasive sets. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, page 351–358, New York, NY, USA, 2012. Association for Computing Machinery. 3

[DW22] Dean Doron and Mary Wootters. High-Probability List-Recovery, and Applications to Heavy Hitters. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:17, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 3

[GG25] Rohan Goyal and Venkatesan Guruswami. Optimal proximity gaps for subspace-design codes and (random) Reed-Solomon codes. Cryptology ePrint Archive, Paper 2025/2054, 2025. 5, 8, 9, 12, 14

[GHKS24] Rohan Goyal, Prahladh Harsha, Mrinal Kumar, and Ashutosh Shankar. Fast list-decoding of univariate multiplicity and folded Reed-Solomon codes. In Santosh Vempala, editor, *Proc. 65th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 328–343, 2024. 10

[GHKS25] Rohan Goyal, Prahladh Harsha, Mrinal Kumar, and Ashutosh Shankar. Fast list recovery of univariate multiplicity and folded Reed-Solomon codes. *arXiv preprint arXiv:2512.00248*, 2025. 4, 7, 10, 18, 19

[GI01] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 658–667, 2001. 3

[GK16] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016. 8, 9

[GR08] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Trans. Inform. Theory*, 54(1):135–150, 2008. (Preliminary version in *38th STOC*, 2006). 3, 7

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009. (Preliminary version in *22nd CCC*, 2007). 3

[GW13] Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of Reed-Solomon codes. *IEEE Trans. Inform. Theory*, 59(6):3257–3268, 2013. (Preliminary version in *26th IEEE Conference on Computational Complexity*, 2011 and *15th RANDOM*, 2011). 3, 4, 7, 10, 18, 19

[GX13] Venkatesan Guruswami and Chaoping Xing. List decoding Reed-Solomon, Algebraic-Geometric, and Gabidulin subcodes up to the Singleton bound. In *Proceedings of the 45th ACM Symposium on Theory of Computing*, pages 843–852, June 2013. 8

[HLR21] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. Fiat–Shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge). In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 750–760, New York, NY, USA, 2021. Association for Computing Machinery. 3

[INR10] Piotr Indyk, Hung Q. Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, page 1126–1142, USA, 2010. Society for Industrial and Applied Mathematics. 3

[JSW$^+$25] Stephen P. Jordan, Noah Shutty, Mary Wootters, Adam Zalcman, Alexander Schmidhuber, Robbie King, Sergei V. Isakov, Tanuj Khattar, and Ryan Babbush. Optimization by decoded quantum interferometry. *Nature*, 646(8086):831–836, October 2025. 3

[Kop14] Swastik Kopparty. Some remarks on multiplicity codes. In Alexander Barg and Oleg R. Musin, editors, *Discrete Geometry and Algebraic Combinatorics*, volume 625 of *Contemporary Mathematics*, pages 155–176. AMS, 2014. 7

[KRSW23] Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved list decoding of Folded Reed-Solomon and Multiplicity codes. *SIAM J. Comput.*, 52(3):794–840, 2023. (Preliminary version in *59th FOCS*, 2018). 3, 4, 7, 8, 10, 11, 12

[KTS22] Itay Kalev and Amnon Ta-Shma. Unbalanced Expanders from Multiplicity Codes. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*, volume 245 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 3

[LMS25] Matan Levi, Jonathan Mosheiff, and Nikhil Shagrithaya. Random Reed-Solomon codes and random linear codes are locally equivalent. *arXiv preprint arXiv:2406.02238*, 2025. To appear in FOCS 2025. 3, 4, 6, 21

[LNNT19] Kasper Green Larsen, Jelani Nelson, Huy L. Nguyen, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. *Commun. ACM*, 62(8):95–100, July 2019. 3

[LS25] Ray Li and Nikhil Shagrithaya. Near-Optimal List-Recovery of Linear Code Families. In Alina Ene and Eshan Chattopadhyay, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2025)*, volume 353 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 53:1–53:14, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 3, 4, 6, 12, 21

[NPR11] Hung Quoc Ngo, Ely Porat, and Atri Rudra. Efficiently decodable error-correcting list disjunct matrices and applications - (extended abstract). In *International Colloquium on Automata, Languages and Programming*, 2011. 3

[Res20] Nicolas Resch. List-decodable codes:(randomized) constructions and applications. 2020. Ph.D. Thesis. Carnegie Mellon University, 2020.: http://reportsarchive.adm.cs.cmu.edu/anon/2020/abstracts/20-113.html. 3

[RV25] Nicolas Resch and S. Venkitesh. List recoverable codes: The good, the bad, and the unknown (hopefully not ugly). *arXiv preprint arXiv:2510.07597*, 2025. 4

[Sri25] Shashank Srivastava. Improved list size for folded Reed-Solomon codes. In Yossi Azar and Debmalya Panigrahi, editors, *Proc. 36th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 2040–2050, 2025. 4, 8, 9

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, March 2005. 3

[Tam24] Itzhak Tamo. Tighter list-size bounds for list-decoding and recovery of folded Reed-Solomon and multiplicity codes. *IEEE Trans. Inform. Theory*, 70(12):8659–8668, 2024. 3, 4, 7, 8, 10, 11, 12

[YZ24] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. *Journal of the ACM*, 71(3):1–50, June 2024. 3

A List-recovery bounds via Brascamp-Lieb for subspace-design codes

The following results and proofs are entirely due to [BCDZ25a] (see their Theorems 4.2 and 5.2) and we state and reprove them here in our terminology of subspace-design codes.

The amazing insight in [BCDZ25a] was to connect the subspace-design property to discrete Brascamp Lieb inequalities [CCE09, CDK$^+$13, CDK$^+$24]. They also provide a self-contained proof of the version of the inequality needed for this application, as stated below.

Theorem A.1 (Brascamp-Lieb)**.** *Let V be a d-dimensional vector space over $\mathbb{F}_q$. For each $i \in [m]$, consider a scalar $s_i \geq 0$ and a linear map $\pi_i : V \to V_i$, where V_i is a d_i-dimensional vector space over $\mathbb{F}_q$. Assume further that for all subspaces $U \subseteq V$, the following holds:*

$$\dim U \leq \sum_{i=1}^{n} s_i \dim(\pi_i(U)) \ .$$

Then, for every probability distribution X over V, we have

$$H(X) \leq \sum_{i=1}^{n} s_i H(\pi_i(X)) \ ,$$

where as usual $H(D)$ denotes the Shannon entropy of a distribution D.

Lemma A.2. *If $\mathcal{C}$ is a τ-subspace-design code, and π_i is the projection map from $\mathcal{C}$ to the ith coordinate, then for very linear space $\mathcal{H}$ of dimension r, we have*

$$\dim \mathcal{H} \leq \frac{\sum_{i=1}^{n} \dim(\pi_i(\mathcal{H}))}{(1-\tau(r))n} \ . \tag{6}$$

Proof. Observe that (6) is equivalent to

$$\sum_{i=1}^{n} (\dim(\mathcal{H}) - \dim(\pi_i(\mathcal{H}))) \leq \tau(r) \cdot r \cdot n \iff \sum_{i=1}^{n} \dim(\mathcal{H}_i) \leq \tau(r) \cdot r \cdot n \ ,$$

where $\mathcal{H}_i = \{c \in \mathcal{H} \mid c_i = 0\}$, which is precisely the definition of a subspace-design code. □

Lemma A.3. *Let $\mathcal{C}$ be a τ-subspace-design code, and π_i the projection map from $\mathcal{C}$ to the ith coordinate. Then, the following holds for every linear space $\mathcal{H}$ of dimension r and every distribution X on $\mathcal{H}$:*

$$H(X) \leq \frac{\sum_{i=1}^{n} H(\pi_i(X))}{(1-\tau(r))n}$$

Proof. Let $V = \mathcal{H}$ and $V_i = \pi_i(V)$. Observe, that for all $U \subseteq V$ of dimension $r_U \leq r$, by Lemma A.2, we have

$$\dim U \leq \sum_{i=1}^{n} \frac{1}{(1-\tau(r_U))n} \dim(\pi_i(U)) \leq \sum_{i=1}^{n} \frac{1}{(1-\tau(r))n} \dim(\pi_i(U))$$

since τ is non-decreasing without loss of generality by Proposition 2.7. The claim now follows by applying Theorem A.1 to V. □

Theorem A.4. *If $\mathcal{C} \subseteq \Sigma^n$ is a τ-subspace design code, then for any linear space $\mathcal{H} \subset \mathcal{C}$ of dimension r, and input lists $L_1, \cdots, L_n$ with $L_i \subseteq \Sigma$ and $|L_i| = \ell$, we have*

$$|\{c \in \mathcal{H} \mid \Delta(c, L_1 \times L_2 \times \cdots \times L_n) < 1 - \tau(r) - \varepsilon\}| \leq \left(\tfrac{\ell}{\tau(r)+\varepsilon}\right)^{(\tau(r)+\varepsilon)/\varepsilon} .$$

Proof. Let $\mathsf{LIST} = \{c \in \mathcal{H} \mid \Delta(c, L_1 \times L_2 \times \cdots \times L_n) < 1 - \tau(r) - \varepsilon\}$. Let $L = |\mathsf{LIST}|$ and X be the uniform distribution on LIST. Now, by Lemma A.3, we have

$$(1-\tau(r))n \cdot \log L \leq \sum_{i=1}^{n} H(\pi_i(X)) . \tag{7}$$

For $i \in [n]$, let $\rho_i = \frac{|\{c \in \mathsf{LIST} | c_i \in L_i\}|}{L}$, and denote Y_i to be the indicator random variable for the event $\pi_i(X) \in L_i$. By the chain rule,

$$\begin{aligned} H(\pi_i(X)) &= H(Y_i) + H(\pi_i(X)|Y_i) \\ &\leq H(Y_i) + \rho_i \log \ell + (1-\rho_i)\log(L(1-\rho_i)) \\ &= -\rho_i \log \rho_i + \rho_i \log \ell + (1-\rho_i)\log L . \end{aligned} \tag{8}$$

Define ρ to be the average of the ρ_i's. Combining (7) and (8), we have

$$\begin{aligned} (\rho - \tau(r))\log L &\leq \rho \log \ell - \frac{\sum_{i=1}^{n} \rho_i \log \rho_i}{n} \\ &\leq \rho \log(\ell/\rho) \end{aligned} \tag{9}$$

where in the second step we used Jensen's inequality and concavity of the function $-x \log x$ for $x \in (0,1)$.

By the definition of LIST, we know $\rho \geq \tau(r) + \varepsilon$. Using this with (9), we have the desired list-size upper bound $L \leq \left(\frac{\ell}{\tau(r)+\varepsilon}\right)^{\frac{\tau(r)+\varepsilon}{\varepsilon}}$. □