Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 91 (2018)

Improved List Decoding of Folded Reed-Solomon and Multiplicity Codes*

Swastik Kopparty† Noga Ron-Zewi‡ Shubhangi Saraf§ Mary Wootters¶

September 28, 2020

Abstract

We show new and improved list decoding properties of folded Reed-Solomon (RS) codes and multiplicity codes. Both of these families of codes are based on polynomials over finite fields, and both have been the sources of recent advances in coding theory: Folded RS codes were the first known explicit construction of *capacity-achieving* list decodable codes (Guruswami and Rudra, *IEEE Trans. Information Theory*, 2010), and multiplicity codes were the first construction of *high-rate* locally decodable codes (Kopparty, Saraf, and Yekhanin, *J. ACM*, 2014).

In this work, we show that folded RS codes and multiplicity codes are in fact better than was previously known in the context of list decoding and local list decoding. Our first main result shows that folded RS codes achieve list decoding capacity with *constant* list sizes, independent of the block length. Prior work with constant list sizes first obtained list sizes that are polynomial in the block length, and relied on pre-encoding with subspace evasive sets to reduce the list sizes to a constant (Guruswami and Wang, *IEEE Trans. Information Theory*, 2012; Dvir and Lovett, *STOC*, 2012). The list size we obtain is $(1/\varepsilon)^{O(1/\varepsilon)}$ where ε is the gap to capacity, which matches the list size obtained by pre-encoding with subspace evasive sets.

For our second main result, we observe that *univariate* multiplicity codes exhibit similar behavior, and use this, together with additional ideas, to show that *multivariate* multiplicity codes are *locally* list decodable *up to their minimum distance*. By known reductions, this gives in turn capacity-achieving locally list decodable codes with query complexity $\exp(\tilde{O}((\log N)^{5/6}))$. This improves on the tensor-based construction of (Hemenway, Ron-Zewi, and Wootters, *SICOMP*, 2019), which gave capacity-achieving locally list decodable codes of query complexity $N^{\tilde{O}(1/\log\log N)}$, and almost matches the best known query complexity of $\exp(\tilde{O}(\sqrt{\log N}))$ for high-rate locally (uniquely) decodable codes (Kopparty, Meir, Ron-Zewi, and Saraf, *J. ACM*, 2017).

*This is an updated and revised version of an extended abstract that has appeared at FOCS 2018 (see [KRSW18] for a full version). The current version contains some improvements and simplifications over the previous version. For simplicity and clarity, we have also decided to omit some of the results from [KRSW18].

†Department of Mathematics and Department of Computer Science, Rutgers University. Research supported in part by NSF grants CCF-1253886 and CCF-1540634. swastik.kopparty@gmail.com.

‡Department of Computer Science, Haifa University. noga@cs.haifa.ac.il. Research supported in part by BSF grant 2017732 and ISF grant 735/20.

§Department of Mathematics and Department of Computer Science, Rutgers University. Research supported in part by NSF grants CCF-1350572 and CCF-1540634. shubhangi.saraf@gmail.com.

¶Department of Computer Science and Department of Electrical Engineering, Stanford University. marykw@stanford.edu. Research supported in part by NSF grants CCF-1657049, CCF/BSF-1814629, and CAREER grant CCF-1844628, and by a Sloan Research Fellowship.

ISSN 1433-8092

1 Introduction

An error correcting code $C \subset \Sigma^n$ is a collection of *codewords* c of length n over an alphabet Σ. The goal in designing C is to enable the recovery of a codeword $c \in C$ given a corrupted version $\tilde{c}$ of c, while at the same time making C as large as possible. In the classical unique decoding problem, the goal is to efficiently recover c from any $\tilde{c} \in \Sigma^n$ so that c and $\tilde{c}$ differ in at most αn places; this requires the *relative distance* δ of the code (that is, the minimum fraction of places on which any two codewords differ) to be at least 2α.

Modern applications of error correcting codes, both in coding theory and theoretical computer science, have highlighted the importance of variants of the unique decoding problem, incuding *list decoding,* and *local decoding.* In list decoding, the amount of error α is large enough that unique recovery of the codeword c is impossible (that is, $\alpha > \delta/2$), and instead the goal is to return a short list $\mathcal{L} \subset C$ with the guarantee that $c \in \mathcal{L}$. In local decoding, we still have $\alpha < \delta/2$, but the goal is to recover a single symbol c_i of a codeword c, after querying not too many positions of the corrupted codeword $\tilde{c}$. In a variant known as *local list decoding,* we seek local information about a symbol even when $\alpha > \delta/2$. List-decoding, local decoding, and local list decoding are important primitives in error correcting codes, with applications in coding theory, complexity theory, pseudorandomness and cryptography.

Algebraic codes have been at the heart of the study of list decoding, local-decoding and local list decoding. One classical example of this is Reed-Solomon (RS) codes, whose codewords are comprised of evaluations of low-degree polynomials.[1] In the late 1990's, Guruswami and Sudan [Sud97, GS99] gave an algorithm for efficiently list decoding Reed-Solomon codes well beyond half the distance of the code, and this kicked off the field of algorithmic list decoding. A second example is Reed-Muller (RM) codes, the multivariate analogue of Reed-Solomon codes. The structure of Reed-Muller codes is very amenable to local algorithms: a codeword of a Reed-Muller code corresponds to a multivariate low-degree polynomial, and considering the restriction of that polynomial to a line yields a univariate low-degree polynomial, *a.k.a.* a Reed-Solomon codeword. This local structure is the basis for Reed-Muller codes being locally testable [RS96] and locally decodable [Lip90, BFLS91]. Using this locality in concert with the Guruswami-Sudan algorithm leads to local list decoding algorithms for these codes [AS03, STV01].

More recently, variants of Reed-Solomon and Reed-Muller codes have emerged to obtain improved list decoding and local-decoding properties. Two notable examples, which are the focus of this work, are *Folded Reed-Solomon* (FRS) *codes* and *multiplicity codes.* Both of these constructions have led to recent advances in coding theory. We introduce these codes informally here, and give formal definitions in Section 2.

Folded Reed-Solomon codes, introduced by Guruswami and Rudra in [GR08], are a simple variant of Reed-Solomon codes. If the codeword of a Reed-Solomon code is $(c_0, c_2, \ldots, c_{n-1}) \in \Sigma^n$, then

[1] That is, a codeword of an RS code has the form $(f(x_0), f(x_1), \ldots, f(x_{n-1})) \in \mathbb{F}^n$ for some low-degree polynomial $f \in \mathbb{F}[X]$.

the folded version (with *folding parameter* s) is

$$\left(\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{s-1} \end{bmatrix}, \begin{bmatrix} c_s \\ c_{s+1} \\ \vdots \\ c_{2s-1} \end{bmatrix}, \ldots, \begin{bmatrix} c_{n-s} \\ c_{n-s+1} \\ \vdots \\ c_{n-1} \end{bmatrix} \right) \in (\Sigma^s)^{n/s}.$$

The main property of these codes that makes them interesting is that they admit much better list decoding algorithms [GR08] than the original Guruswami-Sudan algorithm: more precisely, it allows for the error tolerance α to be much larger for a code of the same rate,[2] asymptotically obtaining the *optimal* trade-off.

Multiplicity codes, introduced in the univariate setting by Rosenbloom and Tsfasman in [RT97] and in the multivariate setting by Kopparty, Saraf and Yekhanin in [KSY14], are variants of polynomial codes that also include evaluations of derivatives. That is, while a symbol of a RS codeword is of the form $f(x) \in \mathbb{F}_q$ for some low-degree polynomial $f \in \mathbb{F}[X]$ and some $x \in \mathbb{F}_q$, a symbol in a univariate multiplicity code codeword is of the form $(f(x), f^{(1)}(x), f^{(2)}(x), \ldots, f^{(s-1)}(x)) \in \mathbb{F}_q^s$, where s is the *multiplicity parameter*. Similarly, while a symbol of an RM codeword is of the form $f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{F}_q^m$ for some low-degree multivariate polynomial $f \in \mathbb{F}[X_1, \ldots, X_m]$, a symbol in a multivariate multiplicty code includes all partial derivatives of order less than s. Multivariate multiplicity codes were shown in [KSY14] to have strong locality properties, and were the first constructions known of *high-rate* locally decodable codes. Meanwhile, univariate multiplicity codes were shown in [Kop15, GW13] to be list decodable in the same parameter regime as folded Reed-Solomon codes[3], also achieving asymptotically optimal trade-off between rate and error-tolerance.

In this work, we show that Folded Reed-Solomon codes, univariate multiplicity codes, and multivariate multiplicity codes are even more powerful than was previously known in the context of list decoding and local list decoding. Our motivations for this work are threefold:

1. First, FRS codes and multiplicity codes are basic and natural algebraic codes, central to many recent results in coding theory ([GR08, KSY14, Kop15, GW13, DL12, KMRS17], to name a few), and understanding their error-correcting properties is important in its own right.

2. Second, while there have been improved constructions of list decodable and locally list decodable codes building on FRS and multiplicity codes (discussed more below), those constructions involve significant additional pseudorandom ingredients. As a consequence, these constructions eliminate many of the original structural properties of the codes, such as linearity. Our results give simpler and more structured constructions of capacity-achieving list decodable and locally list decodable codes with the best known parameters.

3. Third, by composing our new results with known reductions, we obtain capacity-achieving locally list decodable codes with significantly improved query complexity than was previously known.

[2] The *rate* of a code $C \subseteq \Sigma^n$ is defined as $R = \frac{1}{n} \log_{|\Sigma|}(|C|)$ and quantifies how much information can be sent using the code. We always have $R \in (0, 1)$, and we would like R to be as close to 1 as possible.

[3] Univariate multiplicity codes were previously shown to be list decodable up to the Johnson bound by Nielsen [Nie01].

In what follows, we state our results and contributions more precisely, and provide a more detailed account of related work.

1.1 Our results

1.1.1 Improved list decoding of FRS codes

We start by fixing some notation. Let $C \subseteq \Sigma^n$ be a code. We define the *rate* of C as $R := \frac{1}{n}\log_{|\Sigma|}(|C|)$. The *relative distance* δ of C is the minimum fraction of coordinates on which any two codewords of C differ. The code C is (α, L)-*list decodable* if for any received word $w \in \Sigma^n$, there are at most L codewords $c \in C$ which satisfy that $c_i = w_i$ for all but an α fraction of the coordinates i.

The celebrated Guruswami-Sudan list decoding algorithm [Sud97, GS99], mentioned above, can efficiently list decode Reed-Solomon codes up to radius $\alpha = 1 - \sqrt{1-\delta}$, with polynomial list sizes L; this radius is known as the *Johnson bound*, and as Reed-Solomon codes satisfy $R = 1 - \delta$, this amounts to a decoding radius of $\alpha = 1 - \sqrt{R}$. It is a classical result that there are codes that go beyond a decoding radius of $1 - \sqrt{R}$, while keeping the list size polynomial in n, or even constant: for large alphabet sizes, the "correct" limit, called the *list decoding capacity*, is $\alpha = 1 - R$, and this is achieved by uniformly random codes. More precisely, it is well-known that a random code of rate R and alphabet size $\exp(1/\varepsilon)$ is $(1 - R - \varepsilon, O(1/\varepsilon))$-list decodable. For a decade it was open whether or not one could construct explicit codes which efficiently achieve list decoding capacity.

In a breakthrough result, Guruswami and Rudra [GR08] (building on the work of Parvaresh and Vardy [PV05]) showed that the folding operation described above can make RS codes approach capacity with polynomial list-sizes. More precisely, they showed that an FRS code of rate R and folding parameter $s \approx 1/\varepsilon^2$ is $\left(1 - R - \varepsilon, n^{O(1/\varepsilon)}\right)$-list decodable. Note that the list size is polynomial in n for any constant $\varepsilon > 0$, while ideally one could expect the list size to be independent of n.

Towards reducing the list size, it was later shown by Guruswami and Wang [GW13] that, surprisingly, the list of FRS is contained in a linear subspace of *constant* dimension $O(1/\varepsilon)$. Note that this still does not give an improvement on the list size, as the alphabet size of FRS is at least n. To reduce the list size to a constant, [GW13] proposed to pre-encode these codes with pseudorandom objects called *subspace evasive sets* that intersect constant dimensional subspaces in a constant number of points. They further showed that such objects exist non-explicitly, and posed the question of searching for an explicit construction. Subsequently, their program was carried out by Dvir and Lovett [DL12] who gave an explicit construction of subspace evasive sets that intersect any t-dimensional subspace in at most $(t/\varepsilon)^{O(1/\varepsilon)}$ points, which led to a list size of $(1/\varepsilon)^{O(1/\varepsilon)}$. Note that subspace evasive sets are inherently non-linear, and so the resulting code is non-linear as well.

In this work, we show that in fact FRS codes are *already* list decodable with constant list-sizes, with no additional modification needed! Interestingly, the list size we obtain is $(1/\varepsilon)^{O(1/\varepsilon)}$, which matches the list size obtained via the subspace evasive approach. In particular, this gives the first construction of *linear*[4] capacity-achieving list decodable codes with constant list size.

[4]Many codes in this paper have alphabet $\Sigma = \mathbb{F}_q^s$, where $\mathbb{F}_q$ is a finite field. For such "vector alphabet" codes, we

Theorem 1.1 (List decoding FRS with constant list size (Informal, see Theorem 3.10))**.** *Let C be an* FRS *code of constant rate $R \in (0,1)$ and folding parameter $s \geq \frac{16}{\varepsilon^2}$. Then C is $\left(1-R-\varepsilon, (1/\varepsilon)^{O(1/\varepsilon)}\right)$-list decodable.*

Remark 1.2 (Dealing with large alphabet sizes)**.** The FRS codes in Theorem 3.10 have large alphabet sizes. It turns out that this can be ameliorated using concatenation and expander-based distance amplification [AEL95, GI04]. The details can be found in our preliminary version [KRSW18, Section 6]. In the current version we omit these details and the corresponding theorem statement, as it is technical, and follows similarly to the use of these techniques for alphabet-reduction in prior work.

As mentioned above, it is known that it is possible for capacity-achieving list decodable codes to achieve a list size of $O(1/\varepsilon)$, and it would be very interesting to strengthen the above theorem to this bound.

1.1.2 Improved local list decoding of multiplicity codes

For some time, FRS codes were the only known route to capacity-achieving list decodable codes, until it was shown in [GW13, Kop15] that univariate multiplicity codes also do the job (again, with polynomial list sizes). We first observe that our approach to prove Theorem 1.1 above also applies to univariate multiplicity codes, albeit with a larger (but still constant) list size.[5]

Theorem 1.3 (List decoding univariate multiplicity codes with constant list size (Informal, see Theorem 3.8))**.** *Let C be a univariate multiplicity code of constant rate $R \in (0,1)$ and multiplicity parameter $s \geq \frac{16}{\varepsilon^2}$, defined over a sufficiently large prime field $\mathbb{F}_q$. Then C is $\left(1-R-\varepsilon, (s/\varepsilon)^{O(s/\varepsilon^2)}\right)$-list decodable.*

Our second main result uses the above theorem (together with some additional ideas) to obtain improved *local* list decoding algorithms for *multivariate* multiplicity codes. The definition of local list decoding (given formally as Definition 2.2 below) is a bit involved, but intuitively the idea is as follows. As with list decoding, we have a received word $w \in \Sigma^n$, and the goal is to obtain information about a single symbol c_i of close-by codeword c, given query access to w. More precisely, we will require that the decoder outputs a short list of randomized algorithms $A_1, \ldots, A_L$, each of which corresponds to a codeword c with $|\{i : c_i \neq w_i\}| \leq \alpha n$. The requirement is that if A_r corresponds to a codeword c, then on input i, $A_r(i)$ outputs c_i with high probability, and using no more than t queries to w. If such a decoder exists, we say that the code is (t, α, L)-locally-list decodable. Local list decoding algorithms are at the heart of algorithms in cryptography [GL89], learning theory [KM93], and hardness amplification and derandomization [STV01].

Perhaps the most natural algebraic approach for local list decoding is via Reed-Muller codes, which have a natural local structure. As discussed above, a Reed-Muller codeword corresponds to a

use the term "linear" to mean "$\mathbb{F}_q$-linear".

[5] As shown in our preliminary version [KRSW18, Section 4.1], when the degree d is smaller than the characteristic of the underlying field, one can obtain the same quantitative bound on the list size as in Theorem 1.1. However, when the degree d is larger than the characteristic—which is what is relevant for the application to local list decoding of multivariate multiplicity codes, described below—we only obtain a weaker bound. For simplicity we only state and prove the weaker bound on the list size here.

low-degree multivariate polynomial, and restricting such a polynomial to a line yields a low-degree univariate polynomial, which corresponds to a Reed-Solomon codeword. Using this connection, along with the Guruswami-Sudan list decoding algorithm for Reed-Solomon codes, Arora and Sudan [AS03] and Sudan, Trevisan and Vadhan [STV01] gave algorithms for locally list decoding Reed-Muller codes up to the Johnson bound.[6][7]

One might hope to be able to surpass the Johnson bound, and local list decode multivariate multiplicity codes up to their minimum distance δ; after all, the univariate versions are (globally) list decodable up to their minimum distance $\delta = 1 - R$. However, the natural approach (as in [AS03, STV01]) is to rely on the univariate case, and the fact that the list sizes were large for the univariate case was an obstacle to this approach. Thus, previous work on the local list decodability of multivariate multiplicity codes also only worked up to the Johnson bound [Kop15]. In this work, we use our Theorem 1.3 above, along with some additional ideas, to give a local list decoding algorithm for multivariate multiplicity codes up to their minimum distance.

Theorem 1.4 (Local list decoding multivariate multiplicity codes up to minimum distance (Informal, see Theorem 4.1))**.** *Let C be an m-variate multiplicity code of constant relative distance $\delta \in (0,1)$ and multiplicity parameter $s \geq \frac{64}{\varepsilon^2}$, defined over a sufficiently large prime field $\mathbb{F}_q$. Then C is $(t, \delta - \varepsilon, L)$-locally list decodable for $L = \exp(\mathsf{poly}(s/\varepsilon))$ and $t = \mathsf{poly}(q) \cdot \exp(m \cdot L)$.*

Note that an m-variate multiplicity code has length $N := q^m$, and so for any constant m and s the above theorem gives a query complexity of the form $O(N^{1/m})$. Moreover, by compromising on a sub-constant relative distance, m and s could also be taken to be super-constant, leading to a sub-polynomial query complexity of $N^{o(1)}$.

1.1.3 Capacity-achieving locally list decodable codes

List decoding algorithms for algebraic codes typically extend to the setting of *list recovery*. A code $C \subseteq \Sigma^n$ is (α, ℓ, L)-*list recoverable* if for any sequence of input lists $S_1, \ldots, S_n \subseteq \Sigma$, each of size at most ℓ, there are at most L codewords $c \in C$ which satisfy that $c_i \in S_i$ for all but an α fraction of the coordinates i. List-decoding is the special case of list recovery when $\ell = 1$. The definition can also be extended naturally to the setting of *local* list recovery (see Definition 2.2).

In the setting of list recovery, the Johnson bound translates into a decoding radius of $\alpha = 1 - \sqrt{\ell \cdot (1 - \delta)}$, which in particular requires the rate $R \leq 1 - \delta$ to be smaller than $1/\ell$ for a non-trivial decoding radius. On the other hand, list recovering up to capacity corresponds to a decoding radius of $\alpha = 1 - R$ (for any ℓ, provided that the alphabet size is sufficiently large compared to ℓ). Thus the Guruswami-Sudan list decoding algorithm for Reed-Solomon codes gives a list recovery algorithm for Reed-Solomon codes of rate $R < 1/\ell$, while the Guruswami-Rudra list decoding algorithm for FRS codes gives a list recovery algorithm for FRS codes of *high rate* (arbitrarily close to 1).

[6] Technically these algorithms were only able to list decode up to radius of $1 - \sqrt{2(1-\delta)}$. To go all the way to the Johnson bound of $1 - \sqrt{1-\delta}$, one needs some additional ideas [BK09]; see [GK16a, Kop15] for further variations on this.

[7] There is another regime, where the field size q is constant, and the degree d is much larger than q, in which the Reed-Muller codes can be locally list decoded well beyond the Johnson bound, up to the minimum distance [GKZ08, Gop13, BL18]. Note that in this regime the rate of Reed-Muller codes is extremely low, whereas we are interested in the regime of $d < q$ where both rate and relative distance can be made constant.

However, until recently, we did not know of *any* high-rate *locally* list recoverable codes. The significance of these codes is that, as shown in [HRW17], such codes can be transformed, using the expander-based distance amplification technique of Alon, Edmonds, and Luby (AEL) [AEL95, AL96], into capacity-achieving locally list decodable codes with roughly the same parameters. The only previous construction of high-rate locally list recoverable codes was the tensor-based construction of [HRW17], and applying the AEL distance amplification method mentioned above, this gave the first construction of capacity-achieving locally list decodable codes. This construction achieved arbitrarily small polynomial query complexity, and even slightly sub-polynomial query complexity $N^{\tilde{O}(1/\log\log N)}$.[8] In the recent work [KRR+19], it was shown that using the tensor-based approach, one cannot reduce the query complexity below $N^{\Omega(1/\log\log N)}$.

In this work we observe that the above Theorem 1.4 extends also to the setting of list recovery, and gives an alternative construction of high-rate locally list recoverable codes with significantly lower query complexity of $\exp(\tilde{O}((\log N)^{5/6}))$. Combined with the AEL distance amplification, this gives capacity-achieving locally list decodable codes with the same query complexity. This brings the query complexity for capacity-achieving local list decodability close to the best known query complexity for high-rate locally (uniquely) decodable codes [KMRS17], which is $\exp(\tilde{O}(\sqrt{\log n}))$ (for the same codes).

Theorem 1.5 (Capacity-achieving locally list decodable codes (Informal, see Theorem 5.4))**.** *For any constant $R, \varepsilon > 0$ there exists an infinite family $\{C_N\}_N$ of codes that satisfy the following.*

1. *C_N is a code of block length N and rate at least R.*
2. *C_N is $(t, 1-R-\varepsilon, L)$-locally list decodable for*

$$t = \exp_{R,\varepsilon}\left(\tilde{O}\left((\log N)^{5/6}\right)\right) \qquad \text{and} \qquad L = \exp_{R,\varepsilon}\left(\tilde{O}\left((\log N)^{2/3}\right)\right).$$

3. *The alphabet size of C_N is $O_{R,\varepsilon}(1)$.*

Remark 1.6 (Running time of the local list decoding algorithm)**.** In the preliminary version of this work [KRSW18], it was shown that the running time of the local list decoder can be made polynomial in the query complexity. For simplicity and clarity, we omit this statement—and the additional detail and notation required to prove it—from the current version.

Remark 1.7 (Reducing the query complexity of local list decoder)**.** In the preliminary version of this work [KRSW18] we have shown a slightly lower query complexity of $\exp(\tilde{O}((\log N)^{3/4}))$. This improvement followed from a tighter bound on the list size of univariate multiplicity codes given in Theorem 1.3 for the special case of list recovery from a small fraction of errors. As the analysis is quite involved, and for clarity we present in the current paper a simpler argument which obtains only the weaker bound of $\exp(\tilde{O}((\log N)^{5/6}))$.

Next we give an overview of the proofs of our main results.

[8] Here, the $\tilde{O}$ notation hides logarithmic factors in the argument.

1.2 Overview of techniques

1.2.1 List decoding FRS codes with constant list size

Let $C \subseteq \Sigma^n$ be a folded Reed-Solomon code with constant rate $R \in (0,1)$, relative distance $\delta = 1 - R$, and folding parameter s, so that $\Sigma = \mathbb{F}_q^s$. We begin by describing a simple argument that gives a slightly worst list size of $(1/\varepsilon)^{O(1/\varepsilon^2)}$. Then we will explain how it can be tightened to give a list size of $(1/\varepsilon)^{O(1/\varepsilon)}$, as stated in Theorem 1.1.

Recall that we are given a received word $w \in \Sigma^n$, and we want to find the list $\mathcal{L}$ of all codewords $c \in C$ such that $\mathrm{dist}(w, c) \leq 1 - R - \varepsilon = \delta - \varepsilon$ (i.e., $c_i \neq w_i$ for at most a $(\delta - \varepsilon)$-fraction of the coordinates). Recall also that by [GW13], we know that $\mathcal{L}$ is contained in an $\mathbb{F}_q$-linear subspace V of dimension at most $O(1/\varepsilon)$. How many elements c of the subspace $V \subseteq C$ can have $\mathrm{dist}(c, w) \leq \delta - \varepsilon$? We show that there cannot be too many such c.

The proof is algorithmic: we will give a simple randomized algorithm PRUNE, which when given w and the low dimensional subspace V, either outputs a codeword $c \in V$ or outputs 'Fail'. The guarantee is that for any $c \in \mathcal{L}$, c is output by the algorithm PRUNE with probability at least $p_0 = \Omega_\varepsilon(1)$. This implies that $|\mathcal{L}| \leq \frac{1}{p_0} = O_\varepsilon(1)$.

The algorithm PRUNE works as follows. For some parameter $t = O_\varepsilon(1)$, to be determined later on, we pick coordinates $i_1, i_2, \ldots, i_t \in [n]$ uniformly at random, and let $I := \{i_1, \ldots, i_t\}$. Then the algorithm PRUNE checks if there is a unique codeword $c \in V$ that agrees with w on I (that is, $c_i = w_i$ for all $i \in I$). If so, we output that unique element c; otherwise (i.e., either there are zero or greater than one such c's) we output Fail.

It remains to show that for any $c \in \mathcal{L}$, the algorithm outputs c with constant probability. Fix such a c. Let E_1 be the event that c agrees with w on I, and let E_2 be the event that two different codewords in V agree on I. Note that the algorithm will output c if and only if the event E_1 holds and the event E_2 does not hold, so the probability that c is output is at least $\Pr[E_1] - \Pr[E_2]$.

By assumption, $c_i = w_i$ for at least a $(1 - \delta + \varepsilon)$-fraction of the coordinates $i \in [n]$, and so $\Pr[E_1] \geq (1 - \delta + \varepsilon)^t$. Next, note that by linearity, the event E_2 can be alternatively expressed as $\dim\left(V \cap \left(\bigcap_{i \in I} H_i\right)\right) > 0$, where $H_i := \{v \in \Sigma^n | v_i = 0\}$. Lemma 2 from [SY11] (stated in a different language, and proven for a very different purpose) shows that for any linear space V with dimension r and relative distance at least δ, for a large enough t (depending on r and δ), it is very unlikely that $\dim\left(V \cap \left(\bigcap_{i \in I} H_i\right)\right) > 0$.

One way to prove (a version of) Lemma 2 from [SY11] is as follows. First observe that for a random $i \in [n]$, we have that $\dim(V \cap H_i) < \dim(V)$ with probability at least δ. To see this, fix a non-zero element $v \in V$. By assumption, v has at least a δ-fraction of non-zero coordinates, so with probability at least δ we have that $v_i \neq 0$. Conditioned on this, we have that $v \notin V \cap H_i$, and so the dimension reduces by at least 1 when intersecting with H_i. Iterating over this, we conclude that the probability that $\dim\left(V \cap \left(\bigcap_{i \in I} H_i\right)\right) > 0$ is at most

$$\binom{t}{t-r+1} \cdot (1-\delta)^{t-r+1} \leq (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r.$$

We conclude that c is output by the algorithm with probability at least

$$p_0 \geq \Pr[E_1] - \Pr[E_2] \geq (1-\delta+\varepsilon)^t - (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r, \tag{1}$$

where $\delta \in (0,1)$ is a constant and $r = O(1/\varepsilon)$. Finally, it can be verified that the right hand term is comparable to the left hand term when setting $t \approx \frac{1}{\varepsilon^2}\log\left(\frac{1}{\varepsilon}\right)$. In this case, we get that the algorithm PRUNE outputs any codeword $c \in \mathcal{L}$ with probability at least $p_0 = \varepsilon^{O(1/\varepsilon^2)}$, and so the list size is at most $|\mathcal{L}| \leq \frac{1}{p_0} = (1/\varepsilon)^{O(1/\varepsilon^2)}$.

Note that in the above argument, the only special property of FRS (besides linearity and distance) we use is that its list is contained in a low-dimensional subspace. Specifically, the above argument shows more generally (see Lemma 3.1 for a formal statement) that if C is *any* $\mathbb{F}_q$-linear code of relative distance δ that is list decodable up to a radius of $\delta - \varepsilon$ with a list of dimension r, then C is $(\delta-\varepsilon, L)$-list decodable, where L depends only on δ, ε, and r.

In particular, this argument applies not only to FRS codes but also to univariate multiplicity codes. This results in Theorem 1.3.

In Theorem 1.1, we get an improved bound of $(1/\varepsilon)^{O(1/\varepsilon)}$ on the list size of FRS codes. To obtain this, we use a tighter bound on the probability that $\dim\left(V \cap \left(\bigcap_{i\in I} H_i\right)\right) > 0$ for the special case where V is a subspace of FRS. Such a bound was shown in [GK16b] (once again, in a different language, and for a very different purpose), and it roughly says that the expected dimension of $V \cap H_i$, for a random $i \in [n]$, is at most $(1-\delta)\dim(V)$. Thus, with t applications, we get that the probability that $\dim\left(V \cap \left(\bigcap_{i\in I} H_i\right)\right) > 0$ is at most $(1-\delta)^t \dim(V)$. Consequently, the bound in (1) becomes

$$p_0 \geq \Pr[E_1] - \Pr[E_2] \geq (1-\delta+\varepsilon)^t - (1-\delta)^t \cdot r,$$

for constant $\delta \in (0,1)$ and $r = O(1/\varepsilon)$, and setting this time $t \approx \frac{1}{\varepsilon}\log\left(\frac{1}{\varepsilon}\right)$ gives $p_0 = \varepsilon^{O(1/\varepsilon)}$ and $|\mathcal{L}| \leq \frac{1}{p_0} = (1/\varepsilon)^{O(1/\varepsilon)}$.

Remark 1.8 (A tighter bound on the list size for low-degree univariate multiplicity codes.)**.** The tighter bound of $(1/\varepsilon)^{O(1/\varepsilon)}$ on the list size can also be shown to hold for univariate multiplicity codes whose degree d is smaller than the characteristic of the field $\mathbb{F}_q$. The idea is to use a different but analogous lemma from [GK16b]. The precise statement and proof can be found in our preliminary version [KRSW18, Section 4.1]. However, for our application to local list decoding of multivariate multiplicity codes, we need to deal with univariate multiplicity codes where the degree d is larger than q. In that case the lemma from [GK16b] does not apply. Thus, in this work, for simplicity we only state and prove the weaker bound of $(s/\varepsilon)^{O(s/\varepsilon^2)}$ stated in Theorem 1.3. This is what follows from the approach described above using the lemma from [SY11] and the bound of $O(s/\varepsilon)$ on the dimension of the list of univariate multiplicity codes.

1.2.2 Local list decoding multivariate multiplicity codes up to minimum distance

We now describe the high-level view of our local list decoding algorithm for multivariate multiplicity codes. Our algorithm follows the general paradigm for local list decoding of Reed-Muller codes up to the Johnson bound by Arora and Sudan [AS03] and Sudan, Trevisan and Vadhan [STV01]. To locally list decode up to the minimum distance, we use our improved bound on the list size for

univariate multiplicity codes (Theorem 1.3), together with some additional ideas elaborated on below.

Local list decoding of Reed-Muller codes is the following problem: we are given a function $r : \mathbb{F}_q^m \to \mathbb{F}_q$ which is promised to be close to the evaluation table of some degree d polynomial $Q(X_1, \ldots, X_m)$. At the high level, the local list decoding algorithm of [STV01] for Reed-Muller codes has two phases: generating advice, and decoding with advice. To generate the advice, we pick a uniformly random $\mathbf{a} \in \mathbb{F}_q^m$ and "guess" a value $z \in \mathbb{F}_q$ (this guessing can be done by going over all $z \in \mathbb{F}_q$; each possible guess corresponds to an element in the list). Our hope for this guess is that z equals $Q(\mathbf{a})$.

Once we have this advice, we use it to decode. We define an oracle machine $M^r[\mathbf{a}, z]$, which takes as advice $[\mathbf{a}, z]$, has query access to r, and given an input $\mathbf{x} \in \mathbb{F}_q^m$, tries to compute $Q(\mathbf{x})$. The algorithm first considers the line λ passing through $\mathbf{x}$ and the advice point $\mathbf{a}$, and globally list decodes the restriction of r to this line to obtain a list $\mathcal{L}_\lambda$ of univariate polynomials. These univariate polynomials are candidates for $Q|_\lambda$. Which of these univariate polynomials is $Q|_\lambda$? We use our guess z (which is supposed to be $Q(\mathbf{a})$): if there is a unique univariate polynomial in the list with value z at $\mathbf{a}$, then we deem that to be our candidate for $Q|_\lambda$, and output its value at the point $\mathbf{x}$ as our guess for $Q(\mathbf{x})$.

The above algorithm will be correct on the point $\mathbf{x}$ if (1) there are not too many errors on the line through $\mathbf{x}$ and $\mathbf{a}$, and (2) no other polynomial in $\mathcal{L}_\lambda$ takes the same value at $\mathbf{a}$ as $Q|_\lambda$ does. The first event is high probability by standard sampling bounds. As to the second event, using the random choice of $\mathbf{a}$, and that any pair of polynomials in $\mathcal{L}_\lambda$ differ by at least a $\delta := 1 - \frac{d}{q}$ fraction of the points, we get that any other polynomial $P \in \mathcal{L}_\lambda$ will agree with $Q|_\lambda$ on $\mathbf{a}$ with probability at most $1 - \delta$. Assuming that $L := |\mathcal{L}_\lambda| \ll \frac{1}{1-\delta}$, one can then apply a union bound over all elements of $\mathcal{L}_\lambda$ to show that with high probability, no other polynomial $P \in \mathcal{L}_\lambda$ agrees with $Q|_\lambda$ on $\mathbf{a}$.

The Johnson bound tells us that $L \ll \frac{1}{1-\delta}$ as long as the decoding radius α is smaller than $1 - \sqrt{2(1-\delta)}$. Suppose we wanted to locally list decode the Reed-Muller code from a larger decoding radius, in which case L may be larger. Our first idea is to use *derivatives* to disambiguate the list. Specifically, as before the advice is generated by choosing a uniformly random point $\mathbf{a} \in \mathbb{F}_q^m$, but now the guess z is supposed to equal to all derivatives up to order s of Q at $\mathbf{a}$ for some integer $s > 0$. To take advantage of derivatives, we recall the "Schwartz-Zippel lemma with multiplicities" from [DKSS13], which says that two univariate degree d polynomials agree on all derivatives up to order s on at most a $\frac{d}{sq}$-fraction of points. This implies in turn that the advice fails to disambiguate any pair of polynomials in $\mathcal{L}_\lambda$ with probability at most $\frac{d}{sq}$, and taking $s \gg L$ allows us to apply a union bound over $\mathcal{L}_\lambda$.

The above idea can be used to locally list decode the Reed-Muller code up to a radius of $1 - \sqrt{1-\delta}$, where the Johnson bound guarantees a constant size list (in particular, this can remove the restriction from [AS03, STV01] that the degree d needs to be at most $1/2$ the size of the field $\mathbb{F}_q$).[9] Beyond this radius, the only guarantee on the list size given by the above algorithm is the

[9] In [BK09] (see also [GK16a, Kop15]), an alternative approach was given to locally list decode the Reed-Muller code up to the Johnson bound. In a nutshell, the idea is to generate the advice by choosing a random *line* ℓ, globally list decoding on the line, and treating each output element in the list as a separate advice. Then to locally decode a point $\mathbf{x}$ one restricts to the unique *plane* spanned by ℓ and $\mathbf{x}$, globally list decodes on the plane, and chooses the unique element in the output list $\mathcal{L}$ that agrees with the advice on ℓ, if such exists. Jumping ahead, this approach will not work for us since in order to locally list decode up to the minimum distance we shall require a good bound

number of different guesses which is $q^{\binom{m+s-1}{m}}$ (since the number of m-variate derivatives up to order s is $\binom{m+s-1}{m}$, and each such derivative can evaluate to any point in $\mathbb{F}_q$). Note that this number is greater than the code length $N := q^m$ for any $s > 0$ (and generally, is very large for our choice of $s \gg L$).

To get a *sublinear* list size, we generate the guess z more cleverly. Specifically, we first choose t random lines through $\mathbf{a}$, then (globally) list decode on the restriction of r to these lines, and finally aggregate the results to obtain a short list Z of guesses for all derivatives up to order s of Q at $\mathbf{a}$. This aggregation turns out to be a *list recovery* problem for Reed-Muller codes evaluated on product sets, and in particular the Johnson bound for list recovery implies that by choosing $t = L^{O(m)}$, the size of Z can be made as small as $O(L)$. Note however that number of queries grew to $L^{O(m)} \cdot q$ (since we are querying $t = L^{O(m)}$ lines, and each containing q points).

For Reed-Muller codes $L \gg q$ beyond the Johnson bound, and so this time number of queries would be super-linear in the code length $N = q^m$. However, at this point we can resort to our improved bound on the list size of univariate multiplicity codes (Theorem 1.3), which gives a constant list size well beyond the Johnson bound, up to the minimum distance of the code. We show that the above approach, with some modifications, can be adapted to locally list decode multiplicity codes up to their minimum distance with sublinear query complexity and list size.

Organization. We begin in Section 2 with notation and preliminary definitions. Once these are in place, in Section 3 we show that folded Reed-Solomon codes and univariate multiplicity codes achieve list decoding capacity with constant list size. In Section 4, we present our local list decoding algorithm for multivariate multiplicity codes up to their minimum distance. Finally, in Section 5 we explain how one can use an extension of the local list decoding algorithm to the setting of list recovery, combined with the expander-based machinery of [AEL95], to give capacity-achieving locally list decodable codes with low query complexity.

2 Notation and Preliminaries

We begin by formally defining the coding-theoretic notions we will need, and by setting notation. We denote by $\mathbb{F}_q$ the finite field of q elements. For any finite alphabet Σ and for any string $x \in \Sigma^n$ the relative weight $\mathsf{wt}(x)$ of x is the fraction of non-zero coordinates of x, that is, $\mathsf{wt}(x) := |\{i \in [n] : x_i \neq 0\}|/n$. For any pair of strings $x, y \in \Sigma^n$, the relative distance between x and y is the fraction of coordinates on which x and y differ, and is denoted by $\mathrm{dist}(x, y) := |\{i \in [n] : x_i \neq y_i\}|/n$. For a positive integer ℓ we denote by $\binom{\Sigma}{\ell}$ the set containing all subsets of Σ of size ℓ, and for any pair of strings $x \in \Sigma^n$ and $S \in \binom{\Sigma}{\ell}^n$ we denote by $\mathrm{dist}(x, S)$ the fraction of coordinates $i \in [n]$ for which $x_i \notin S_i$, that is, $\mathrm{dist}(x, S) := |\{i \in [n] : x_i \notin S_i\}|/n$. Throughout the paper, we use $\exp(n)$ to denote $2^{\Theta(n)}$. Whenever we use log, it is to the base 2. The notation $O_a(n)$ and $\mathsf{poly}_a(n)$ means that we treat a as a constant; that is, $\mathsf{poly}_a(n) = n^{O_a(1)}$.

on the size of $\mathcal{L}$. Our Theorem 1.3 gives such a bound for univariate multiplicity codes, but *a priori* we cannot show such a bound for bivariate multiplicity codes.

2.1 Error-correcting codes

Let Σ be an alphabet and let n be a positive integer (the block length). A code is simply a subset $C \subseteq \Sigma^n$. The elements of a code C are called codewords. If $\mathbb{F}$ is a finite field and Σ is a vector space over $\mathbb{F}$, we say that a code $C \subseteq \Sigma^n$ is $\mathbb{F}$-linear if it is an $\mathbb{F}$-linear subspace of the $\mathbb{F}$-vector space Σ^n. The rate of a code C is the ratio $R := \frac{\log |C|}{\log(|\Sigma|^n)}$, which for $\mathbb{F}$-linear codes equals $\frac{\dim_{\mathbb{F}}(C)}{n \cdot \dim_{\mathbb{F}}(\Sigma)}$. The relative distance $\mathrm{dist}(C)$ of C is the minimum $\delta > 0$ such that for every pair of distinct codewords $c_1, c_2 \in C$ it holds that $\mathrm{dist}(c_1, c_2) \geq \delta$, which for $\mathbb{F}$-linear codes equals the minimum $\delta > 0$ such that $\mathsf{wt}(c) \geq \delta$ for every $c \in C$.

Given a code $C \subseteq \Sigma^n$, we will occasionally abuse notation and think of $c \in C$ as a map $c : \mathcal{D} \to \Sigma$, where $\mathcal{D}$ is some domain of size n. With this notation, the map $c : \mathcal{D} \to \Sigma$ corresponds to the vector $(c(x))_{x \in \mathcal{D}} \in \Sigma^n$. For a code $C \subseteq \Sigma^n$ of relative distance δ, a given parameter $\alpha < \delta/2$, and a string $w \in \Sigma^n$, the problem of decoding from α fraction of errors is the task of finding the unique $c \in C$ (if any) which satisfies $\mathrm{dist}(c, w) \leq \alpha$.

List decoding is a paradigm that allows one to correct more than a $\delta/2$ fraction of errors by returning a small list of close-by codewords. More formally, for $\alpha \in [0, 1]$ and an integer L we say that a code $C \subseteq \Sigma^n$ is (α, L)-list decodable if for any $w \in \Sigma^n$ there are at most L different codewords $c \in C$ which satisfy that $\mathrm{dist}(c, w) \leq \alpha$. List recovery is a more general notion where one is given as input a small list of candidate symbols for each of the coordinates and is required to output a list of codewords that are consistent with many of the input lists. Formally we say that a code $C \subseteq \Sigma^n$ is (α, ℓ, L)-list recoverable if for any $S \in \binom{\Sigma}{\ell}^n$ there are at most L different codewords $c \in C$ which satisfy that $\mathrm{dist}(c, S) \leq \alpha$. Note that list decoding corresponds to the special case of $\ell = 1$.

2.2 Locally correctable and locally list recoverable codes

Intuitively, a code is said to be locally correctable [BFLS91, STV01, KT00] if, given a codeword $c \in C$ that has been corrupted by some errors, it is possible to decode any coordinate of c by reading only a small part of the corrupted version of c. Formally, it is defined as follows.

Definition 2.1 (Locally correctable code (LCC))**.** *We say that a code $C \subseteq \Sigma^n$ is (t, α)-locally correctable if there exists a randomized algorithm A that satisfies the following requirements:*

- ***Input:*** *A takes as input a coordinate $i \in [n]$ and also gets oracle access to a string $w \in \Sigma^n$ that is α-close to a codeword $c \in C$.*
- ***Query complexity:*** *A makes at most t queries to the oracle w.*
- ***Output:*** *A outputs c_i with probability at least $\frac{2}{3}$.*

The following definition generalizes the notion of locally correctable codes to the setting of list decoding / recovery. In this setting the algorithm A is required to find all the nearby codewords in an implicit sense.

Definition 2.2 (Locally list recoverable code)**.** *We say that a code $C \subseteq \Sigma^n$ is (t, α, ℓ, L)-locally list recoverable if there exists a randomized algorithm A that satisfies the following requirements:*

- ***Input:*** *A gets oracle access to a string $S \in \binom{\Sigma}{\ell}^n$.*
- ***Query complexity:*** *A makes at most t queries to the oracle S.*
- ***Output:*** *A outputs L randomized algorithms $A_1, \ldots, A_L$, where each A_j takes as input a coordinate $i \in [n]$, makes at most t queries to the oracle S, and outputs a symbol in Σ.*
- ***Correctness:*** *For every codeword $c \in C$ for which $\mathrm{dist}(c, S) \leq \alpha$, with probability at least $\frac{2}{3}$ over the randomness of A the following event happens: there exists some $j \in [L]$ such that for all $i \in [n]$,*

$$\Pr[A_j(i) = c_i] \geq \frac{2}{3},$$

where the probability is over the internal randomness of A_j.

We say that a code is (t, α, L)-locally list decodable if it is $(t, \alpha, 1, L)$-locally list recoverable.

2.3 Some families of polynomial codes

In this section, we formally define the families of codes we will study: folded Reed-Solomon codes [GR08], univariate multiplicity codes [RT97, KSY14, GW13], and multivariate multiplicity codes [KSY14].

Folded Reed-Solomon codes. Let q be a prime power, and let s, d, n be nonnegative integers such that $n \leq (q-1)/s$. Let $\gamma \in \mathbb{F}_q$ be a primitive element of $\mathbb{F}_q$, and let $a_1, a_2, \ldots, a_n$ be distinct elements in $\{\gamma^{si} \mid 0 \leq i \leq (q-1)/s - 1\}$. Let $\mathcal{D} = \{a_1, \ldots, a_n\}$.

For a polynomial $P(X) \in \mathbb{F}_q[X]$ and $a \in \mathbb{F}_q$, let $P^{[s]}(a) \in \mathbb{F}_q^s$ denote the vector:

$$P^{[s]}(a) = \begin{bmatrix} P(a) \\ P(\gamma a) \\ \vdots \\ P(\gamma^{s-1} a) \end{bmatrix}.$$

The folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n, d)$ is a code over alphabet $\mathbb{F}_q^s$. To every polynomial $P(X) \in \mathbb{F}_q[X]$ of degree at most d, there corresponds a codeword c:

$$c : \mathcal{D} \to \mathbb{F}_q^s,$$

where for each $a \in \mathcal{D}$:

$$c(a) = P^{[s]}(a).$$

Explicitly,

$$\begin{aligned} P(x) &\mapsto \left(P^{[s]}(a_1), P^{[s]}(a_2), \ldots, P^{[s]}(a_n)\right) \\ &= \left(\begin{bmatrix} P(a_1) \\ P(\gamma a_1) \\ \vdots \\ P(\gamma^{s-1} a_1) \end{bmatrix}, \begin{bmatrix} P(a_2) \\ P(\gamma a_2) \\ \vdots \\ P(\gamma^{s-1} a_2) \end{bmatrix}, \ldots, \begin{bmatrix} P(a_n) \\ P(\gamma a_n) \\ \vdots \\ P(\gamma^{s-1} a_n) \end{bmatrix} \right). \end{aligned}$$

Note that Reed-Solomon codes correspond to the special case of $s = 1$. The following claim summarizes the basic properties of folded Reed-Solomon codes.

Claim 2.3 ([GR08])**.** *The folded Reed-Solomon code* $\mathrm{FRS}_{q,s}(n, d)$ *is an* $\mathbb{F}_q$*-linear code over alphabet* $\mathbb{F}_q^s$ *of block length* n*, rate* $(d+1)/(sn)$*, and relative distance at least* $1 - d/(sn)$*.*

Univariate multiplicity codes. Let $P(x)$ be a univariate polynomial over $\mathbb{F}_q$. For $i \in \mathbb{N}$, we define the i'th (Hasse) derivative $P^{(i)}(X)$ as the coefficient of Z^i in the expansion

$$P(X + Z) = \sum_i P^{(i)}(X)Z^i.$$

Let q be a prime power, and let s, d, n be nonnegative integers such that $n \leq q$. Let $a_1, a_2, \ldots, a_n$ be distinct elements in $\mathbb{F}_q$. Let $\mathcal{D} = \{a_1, \ldots, a_n\}$.

For a polynomial $P(X) \in \mathbb{F}_q[X]$, let $P^{(<s)}(a) \in \mathbb{F}_q^s$ denote the vector:

$$P^{(<s)}(a) = \begin{bmatrix} P(a) \\ P^{(1)}(a) \\ \vdots \\ P^{(s-1)}(a) \end{bmatrix}.$$

The univariate multiplicity code $\mathsf{MULT}_{q,s}^{(1)}(n, d)$ is a code over alphabet $\mathbb{F}_q^s$. To every polynomial $P(X) \in \mathbb{F}_q[X]$ of degree at most d, there corresponds a codeword c:

$$c : \mathcal{D} \to \mathbb{F}_q^s,$$

where for each $a \in \mathcal{D}$:

$$c(a) = P^{(<s)}(a).$$

Explicitly,

$$\begin{aligned} P(x) &\mapsto \left(P^{(<s)}(a_1), P^{(<s)}(a_2), \ldots, P^{(<s)}(a_n)\right) \\ &= \left(\begin{bmatrix} P(a_1) \\ P^{(1)}(a_1) \\ \vdots \\ P^{(s-1)}(a_1) \end{bmatrix}, \begin{bmatrix} P(a_2) \\ P^{(1)}(a_2) \\ \vdots \\ P^{(s-1)}(a_2) \end{bmatrix}, \ldots, \begin{bmatrix} P(a_n) \\ P^{(1)}(a_n) \\ \vdots \\ P^{(s-1)}(a_n) \end{bmatrix}\right). \end{aligned}$$

Once again, Reed-Solomon codes correspond to the special case of $s = 1$.

Claim 2.4 ([KSY14], Lemma 9)**.** *The univariate multiplicity code* $\mathsf{MULT}_{q,s}^{(1)}(n, d)$ *is an* $\mathbb{F}_q$*-linear code over alphabet* $\mathbb{F}_q^s$ *of block length* n*, rate* $(d+1)/(sn)$*, and relative distance at least* $1 - d/(sn)$*.*

Of particular importance is the setting where $q = n$ and $\mathcal{D}$ equals the whole field $\mathbb{F}_q$. We refer to this code as the *whole-field univariate multiplcity code*, and denote it by $\mathsf{MULT}_{q,s}^{(1)}(d)$. This will be relevant to multivariate multiplicity codes, which we define next.

Multivariate multiplicity codes. Multivariate multiplicity codes are a generalization of whole-field univariate multiplicity codes to the multivariate setting. For multivariate polynomials $P \in \mathbb{F}_q[X_1, \ldots, X_m]$, we use the notation $\mathbf{X} = (X_1, \ldots, X_m)$ and $\mathbf{X}^{\mathbf{i}} = \prod_j X_j^{i_j}$ where $\mathbf{i} = (i_1, \ldots, i_m) \in \mathbb{N}^m$. For $\mathbf{i} \in \mathbb{N}^m$, we define the **i'th (Hasse) derivative** $P^{(\mathbf{i})}(\mathbf{X})$ by

$$P(\mathbf{X} + \mathbf{Z}) = \sum_{\mathbf{i}} P^{(\mathbf{i})}(\mathbf{X})\mathbf{Z}^{\mathbf{i}}.$$

Let q be a prime power, and let s, d, m be nonnegative integers. Let $U_{m,s}$ denote the set $\{\mathbf{i} \in \mathbb{N}^m \mid \mathsf{wt}(\mathbf{i}) < s\}$. Note that $|U_{m,s}| = \binom{m+s-1}{m}$. Let $\Sigma_{m,s} = \mathbb{F}_q^{U_{m,s}}$.

For a polynomial $P(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$, and a point $\mathbf{a} \in \mathbb{F}_q^m$, define $P^{(<s)}(\mathbf{a}) \in \Sigma_{m,s}$ by:

$$P^{(<s)}(\mathbf{a}) = (P^{(\mathbf{i})}(\mathbf{a}))_{\mathbf{i} \in U_{m,s}}.$$

The **multiplicity code** $\mathsf{MULT}_{q,s}^{(m)}(d)$ is a code over alphabet $\Sigma_{m,s}$. To every polynomial $P(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ of (total) degree at most d, there corresponds a codeword c:

$$c : \mathbb{F}_q^m \to \Sigma_{m,s},$$

where for each $\mathbf{a} \in \mathbb{F}_q^m$,

$$c(\mathbf{a}) = P^{(<s)}(\mathbf{a}).$$

Note that **Reed-Muller codes** correspond to the special case of $s = 1$.

Claim 2.5 ([KSY14], Lemma 9)**.** *The multivariate multiplicity code* $\mathsf{MULT}_{q,s}^{(m)}(d)$ *is an* $\mathbb{F}_q$*-linear code over alphabet* $\Sigma_{m,s}$ *of block length* q^m*, rate at least* $(1 - m^2/s)(d/(sq))^m$*, and relative distance at least* $1 - d/(sq)$*.*

3 Constant-size output list for folded Reed-Solomon and univariate multiplicity codes

In this section we show that folded Reed-Solomon and univariate multiplicity codes are list decodable up to capacity with constant list size, independent of the block length. For this, we first show in Section 3.1 that the list cannot contain many codewords from a low dimensional subspace. Then in Section 3.2 we instantiate this with the results of [GW13] showing that the list of both folded Reed-Solomon and univariate multiplicity codes is contained in a low-dimensional subspace. Finally in Section 3.3 we show a tighter bound on the list size of FRS codes. Towards our results on local list decoding, we prove a more general result that applies also to list recovery.

3.1 Output list has small intersection with low dimensional subspaces

Our main lemma shows that when list recovering a linear code of large distance, the output list cannot contain many codewords from a low dimensional subspace.

Lemma 3.1. *Suppose that $C \subseteq (\mathbb{F}_q^s)^n$ is an $\mathbb{F}_q$-linear code of relative distance δ that is $(\delta-\varepsilon, \ell, L)$-list recoverable. Suppose furthermore that the output list is contained in an $\mathbb{F}_q$-linear subspace $V \subseteq C$ of dimension r. Then C is $(\delta-\varepsilon, \ell, L')$-list recoverable with*

$$L' = \left(\frac{\ell}{1-\delta}\right)^{O\left(\frac{r}{\varepsilon(1-\delta)} \cdot \log\left(\frac{r}{\varepsilon(1-\delta)}\right)\right)}$$

Moreover, there is a randomized algorithm that, given a basis for V, list recovers C with the above parameters in time $\mathsf{poly}\left(\log q, s, n, L'\right)$.

To show that the output list $\mathcal{L}$ of C cannot contain too many elements from V (and to find $\mathcal{L}$ in the process), we first give a preliminary randomized algorithm PRUNE that outputs a small size list $\hat{\mathcal{L}}$ such that any codeword of $\mathcal{L}$ appears in $\hat{\mathcal{L}}$ with probability at least p_0. This implies that $|\mathcal{L}| \leq |\hat{\mathcal{L}}|/p_0$, proving the first part of Lemma 3.1. Now that we know that $|\mathcal{L}|$ is small, our final algorithm simply runs PRUNE $O(\frac{1}{p_0}\log|\mathcal{L}|)$ times and returns the union of the output lists. By a union bound, all elements of $\mathcal{L}$ will appear in the union of the output lists with high probability. This will complete the proof of the second part of Lemma 3.1.

We start by describing the algorithm PRUNE and analyzing it. The algorithm is given as input a tuple of input lists $S \in \binom{\mathbb{F}_q^s}{\ell}^n$, (a basis for) an $\mathbb{F}_q$-linear subspace $V \subseteq C$ of dimension r containing the output list, and a parameter $t \in \mathbb{N}$ to be determined later on.

Algorithm PRUNE(S, V, t)

1. Initialize $\hat{\mathcal{L}} = \emptyset$.
2. Pick $i_1, i_2, \ldots, i_t \in [n]$ independently and uniformly at random.
3. For each choice of $y_1 \in S_{i_1}, y_2 \in S_{i_2}, \ldots, y_t \in S_{i_t}$:
   - If there is exactly one codeword $c \in V$ such that $c_{i_j} = y_j$ for all $j \in [t]$, then:
   $$\hat{\mathcal{L}} \leftarrow \hat{\mathcal{L}} \cup \{c\}.$$
4. Output $\hat{\mathcal{L}}$.

Lemma 3.2. *The algorithm* PRUNE *runs in time* $\mathsf{poly}(\log q, s, n, \ell^t)$*, and outputs a list $\hat{\mathcal{L}}$ containing at most ℓ^t codewords of C, such that any codeword $c \in V$ with* $\mathrm{dist}(c, S) \leq \delta - \varepsilon$ *appears in $\hat{\mathcal{L}}$ with probability at least*

$$(1-\delta+\varepsilon)^t - (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r.$$

Proof. We clearly have that $|\hat{\mathcal{L}}| \leq \ell^t$, and that the algorithm has the claimed running time. Fix a codeword $\hat{c} \in V$ such that $\mathrm{dist}(\hat{c}, S) \leq \delta - \varepsilon$, we shall show below that $\hat{c}$ belongs to $\hat{\mathcal{L}}$ with probability at least

$$(1-\delta+\varepsilon)^t - (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r.$$

Let E_1 denote the event that $\hat{c}_{i_j} \in S_{i_j}$ for all $j \in [t]$. Let E_2 denote the event that two different codewords $v, v' \in V$ agree on $i_1, \ldots, i_t$ (that is, $v_{i_j} = v'_{i_j}$ for all $j \in [t]$). By the assumption that $\mathrm{dist}(\hat{c}, S) \leq \delta - \varepsilon$, we readily have that

$$\Pr[E_1] \geq (1 - \delta + \varepsilon)^t.$$

Claim 3.3 below also shows that

$$\Pr[E_2] \leq (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r. \tag{2}$$

So we have that both E_1 occurs and E_2 does not occur with probability at least

$$(1 - \delta + \varepsilon)^t - (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r.$$

If E_2 does not occur, then for every choice of $y_1 \in S_{i_1}, y_2 \in S_{i_2}, \ldots, y_t \in S_{i_t}$, there can be at most one codeword $c \in V$ such that $c_{i_j} = y_j$ for all $j \in [t]$. If E_1 also occurs, then in the iteration of Step 3 where $y_j = \hat{c}_{i_j}$ for each $j \in [t]$, the algorithm will take $c = \hat{c}$, and thus $\hat{c}$ will be included in $\hat{\mathcal{L}}$. This completes the proof of the lemma. □

It remains to prove the following claim.

Claim 3.3.

$$\Pr[E_2] \leq (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r.$$

Proof. For $i \in [n]$, let

$$H_i := \{v \in (\mathbb{F}_q^s)^n \mid v_i = 0\},$$

and for $j = 0, 1, \ldots, t$, let

$$V_j := V \cap H_{i_1} \cap H_{i_2} \cap \ldots \cap H_{i_j},$$

and $r_j := \dim_{\mathbb{F}_q}(V_j)$. Observe that $r = r_0 \geq r_1 \geq \ldots \geq r_t$, and that event E_2 holds if and only if $r_t > 0$.

Next we claim that for any $j = 0, 1, \ldots, t-1$, it holds that $r_{j+1} \leq \min\{0, r_j - 1\}$ with probability at least δ over the choice of i_{j+1}. To see this, note first that if $r_j = 0$, then $r_{j+1} \leq r_j \leq 0$ and so we are done. Otherwise, if $r_j \neq 0$ then there exists a non-zero vector $v \in V_j$. Recalling that $V_j \subseteq V \subseteq C$, we have that $\mathsf{wt}(v) \geq \delta$, and so $v_{i_{j+1}} \neq 0$ with probability at least δ over the choice of i_{j+1}. But recalling that $V_{j+1} \subseteq H_{i_{j+1}}$, this implies in turn that $v \notin V_{j+1}$. We conclude that there exists a non-zero $v \in V_j$ so that $v \notin V_{j+1}$, and so the dimension of V_{j+1} is strictly smaller than that of V_j.

Finally, note that if $r_t > 0$, then it must hold that $r_{j+1} > \min\{0, r_j - 1\}$ for at least $t - r + 1$ of the j's in $0, 1, \ldots, t-1$. By the above, the probability of this event is at most

$$\binom{t}{t-r+1} \cdot (1-\delta)^{t-r+1} \leq (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r.$$

□

Our main Lemma 3.1 now follows as an immediate corollary of the above Lemma 3.2.

Proof of Lemma 3.1. Setting

$$t := 3 \cdot \frac{r}{\varepsilon(1-\delta)} \cdot \log\left(\frac{r}{\varepsilon(1-\delta)}\right)$$

in Lemma 3.2, we have that

$$|\hat{\mathcal{L}}| \leq \ell^t \leq \ell^{O\left(\frac{r}{\varepsilon(1-\delta)} \cdot \log\left(\frac{r}{\varepsilon(1-\delta)}\right)\right)},$$

and moreover, any codeword in the output list $\mathcal{L}$ of C appears in $\hat{\mathcal{L}}$ with probability at least

$$\begin{aligned}
p_0 : &= (1-\delta+\varepsilon)^t - (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r \\
&\geq (1-\delta)^t \cdot \left[(1+\varepsilon)^t - \left(\frac{t}{1-\delta}\right)^r\right] \\
&\geq (1-\delta)^t \cdot \left[\left(\frac{r}{\varepsilon(1-\delta)}\right)^{3\cdot r} - \left(3 \cdot \frac{r}{\varepsilon(1-\delta)^2} \cdot \log\left(\frac{r}{\varepsilon(1-\delta)}\right)\right)^r\right] \\
&\geq (1-\delta)^t \\
&\geq (1-\delta)^{O\left(\frac{r}{\varepsilon(1-\delta)} \cdot \log\left(\frac{r}{\varepsilon(1-\delta)}\right)\right)},
\end{aligned}$$

where the one before last inequality holds when the ratio $\frac{r}{\varepsilon(1-\delta)}$ is sufficiently large.

This implies in turn that

$$|\mathcal{L}| \leq \frac{|\hat{\mathcal{L}}|}{p_0} \leq \left(\frac{\ell}{1-\delta}\right)^{O\left(\frac{r}{\varepsilon(1-\delta)} \cdot \log\left(\frac{r}{\varepsilon(1-\delta)}\right)\right)},$$

proving the first part of Lemma 3.1.

To find $\mathcal{L}$, our final algorithm simply runs PRUNE $O(\frac{1}{p_0} \log |\mathcal{L}|)$ times and returns the union of the output lists. By a union bound, all elements of $\mathcal{L}$ will appear in the union of the output lists with high probability (say, at least 0.99). This completes the proof of the second part of Lemma 3.1. □

3.2 Constant-size output list for folded Reed-Solomon and univariate multiplicity codes

Our first corollary is obtained by instantiating Lemma 3.1 with the following result from [GW13] (see also [GW12]), showing that the output list of folded Reed-Solomon codes is contained in a low dimensional subspace (which can also be found efficiently).

Theorem 3.4 (Constant-dimensional output list for FRS, [GW12], Theorem 7)**.** *Let q be a prime power, and let s, d, n be nonnegative integers such that $n \leq (q-1)/s$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n,d)$. Let $\varepsilon > 0$ and $\ell \in \mathbb{N}$ be such that $\frac{16\ell}{\varepsilon^2} \leq s$. Then $\mathrm{FRS}_{q,s}(n,d) \subseteq (\mathbb{F}_q^s)^n$ is $(\delta-\varepsilon, \ell, L)$-list recoverable, where the output list is contained in an $\mathbb{F}_q$-linear subspace $V \subseteq \mathrm{FRS}_{q,s}(n,d)$ of dimension at most $r = \frac{4\ell}{\varepsilon}$.*

Moreover, there is a (deterministic) algorithm that outputs a basis for V in time $\mathsf{poly}(\log q, s, d, n)$.

Remark 3.5. Theorem 7 of [GW12] only deals with the case where $a_i = \gamma^{s(i-1)}$ for all $i = 1, \ldots, n$, and $\ell = 1$. However, it can be verified that the proof goes through for any choice of distinct $a_1, a_2, \ldots, a_n$ in $\{\gamma^{si} \mid 0 \le i \le (q-1)/s - 1\}$, and $\ell \in \mathbb{N}$ (see discussion at end of [GW12, Section 2.4]). In this setting, the bound on the decoding radius in Theorem 7 of [GW12] becomes

$$\alpha := 1 - \frac{\ell}{r+1} - \frac{r}{r+1} \cdot \frac{s}{s-r+1} \cdot \frac{d}{sq},$$

and the above Theorem 3.4 then follows by noting that $\alpha \ge \delta - \varepsilon$ when setting $s \ge \frac{16\ell}{\varepsilon^2}$ and $r = \frac{4\ell}{\varepsilon}$.

We now obtain the following theorem as an immediate corollary of Lemma 3.1 and Theorem 3.4.

Theorem 3.6 (Constant-size output list for FRS)**.** *Let q be a prime power, and let s, d, n be nonnegative integers such that $n \le (q-1)/s$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n, d)$. Let $\varepsilon > 0$ and $\ell \in \mathbb{N}$ be such that $\frac{16\ell}{\varepsilon^2} \le s$. Then $\mathrm{FRS}_{q,s}(n, d)$ is $(\delta - \varepsilon, \ell, L)$-list recoverable with*

$$L = \left(\frac{\ell}{\varepsilon(1-\delta)}\right)^{O\left(\frac{\ell}{\varepsilon^2(1-\delta)} \cdot \log\left(\frac{\ell}{1-\delta}\right)\right)}.$$

In particular, if $\delta \in (0, 1)$ and $\ell \in \mathbb{N}$ are constant, then the output list size is $\left(\frac{1}{\varepsilon}\right)^{O(1/\varepsilon^2)}$.

Moreover, there is a randomized algorithm that list recovers $\mathrm{FRS}_{q,s}(n, d)$ with the above parameters in time $\mathsf{poly}(\log q, s, d, n, L)$.

Our second corollary is obtained by replacing Theorem 7 of [GW12] with Theorem 17 of that paper, showing that the output list of univariate multiplicity codes is contained in a low-dimensional subspace. However, towards our application for local list decoding, we shall need a slight modification of that result, specifically: (1) we need to talk about list recovery, not just list decoding, (2) we allow the degree to be larger than the field size, while [GW12] assumes that the degree is smaller than the characteristic of the field, and (3) we work with Hasse derivatives, while [GW12] works with standard derivatives. All differences are minor; for completeness we include a full proof in Appendix A.

Theorem 3.7 (Constant-dimensional output list for UniMULT)**.** *Let q be a prime power, and let s, d, n be nonnegative integers such that $n \le q$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(n, d)$. Let $\varepsilon > 0$ and $\ell \in \mathbb{N}$ be such that $\frac{16\ell}{\varepsilon^2} \le s$ and $\frac{4\ell}{\varepsilon} \le \mathsf{char}(\mathbb{F}_q)$. Then $\mathsf{MULT}^{(1)}_{q,s}(n, d) \subseteq (\mathbb{F}_q^s)^n$ is $(\delta - \varepsilon, \ell, L)$-list recoverable, where the output list is contained in an $\mathbb{F}_q$-linear subspace $V \subseteq \mathsf{MULT}^{(1)}_{q,s}(n, d)$ of dimension at most $\frac{4\ell}{\varepsilon} \cdot \left(1 + \frac{d}{\mathsf{char}(\mathbb{F}_q)}\right)$.*

Moreover, there is a (deterministic) algorithm that outputs a basis for V in time $\mathsf{poly}(\log q, s, d, n)$.

Instantiating Lemma 3.1 with the above theorem gives the following.

Theorem 3.8 (Constant-size output list for UniMult)**.** *Let q be a prime power, and let s, d, n be nonnegative integers such that $n \le q$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance*

of the univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(n,d)$*. Let* $\varepsilon > 0$ *and* $\ell \in \mathbb{N}$ *be such that* $\frac{16\ell}{\varepsilon^2} \le s$ *and* $\frac{4\ell}{\varepsilon} \le \mathsf{char}(\mathbb{F}_q)$*. Then* $\mathsf{MULT}^{(1)}_{q,s}(n,d)$ *is* $(\delta-\varepsilon,\ell,L)$*-list recoverable with*

$$L = \left(\frac{d}{\mathsf{char}(\mathbb{F}_q)} \cdot \frac{\ell}{\varepsilon(1-\delta)}\right)^{O\left(\frac{d}{\mathsf{char}(\mathbb{F}_q)} \cdot \frac{\ell}{\varepsilon^2(1-\delta)} \cdot \log\left(\frac{\ell}{1-\delta}\right)\right)}.$$

Moreover, there is a randomized algorithm that list recovers $\mathsf{MULT}^{(1)}_{q,s}(n,d)$ *with the above parameters in time* $\mathsf{poly}(\log q, s, d, n, L)$*.*

3.3 Tighter bound on output list size of folded Reed-Solomon codes

For the case of folded Reed-Solomon codes we can obtain the following stronger version of Lemma 3.1, showing that the output list of folded Reed-Solomon codes contains even fewer elements from a low-dimensional subspace.

Lemma 3.9. *Let* q *be a prime power, and let* s, d, n *be nonnegative integers such that* $n \le (q-1)/s$*. Let* $\delta := 1 - \frac{d}{sn}$ *be a lower bound on the relative distance of the folded Reed-Solomon code* $\mathrm{FRS}_{q,s}(n,d)$*. Suppose that* $\mathrm{FRS}_{q,s}(n,d)$ *is* $(\delta-\varepsilon,\ell,L)$*-list recoverable, and that the output list is contained in an* $\mathbb{F}_q$*-linear subspace* $V \subseteq \mathrm{FRS}_{q,s}(n,d)$ *of dimension* $r \le \frac{\varepsilon s}{4}$*. Then* $\mathrm{FRS}_{q,s}(n,d)$ *is* $(\delta-\varepsilon,\ell,L')$*-list recoverable with* $L' = \left(\frac{\ell}{1-\delta}\right)^{O((\log r)/\varepsilon)}$*.*

Moreover, there is a randomized algorithm that, given a basis for V*, list recovers* $\mathrm{FRS}_{q,s}(n,d)$ *with the above parameters in time* $\mathsf{poly}\left(\log q, s, n, L'\right)$*.*

Combining the above lemma with Theorem 3.4, we obtain the following corollary.

Theorem 3.10 (Tighter bound on output list size for FRS)**.** *Let* q *be a prime power, and let* s, d, n *be nonnegative integers such that* $n \le (q-1)/s$*. Let* $\delta := 1 - \frac{d}{sn}$ *be a lower bound on the relative distance of the folded Reed-Solomon code* $\mathrm{FRS}_{q,s}(n,d)$*. Let* $\varepsilon > 0$ *and* $\ell \in \mathbb{N}$ *be such that* $\frac{16\ell}{\varepsilon^2} \le s$*. Then* $\mathrm{FRS}_{q,s}(n,d)$ *is* $(\delta-\varepsilon,\ell,L)$*-list recoverable with*

$$L = \left(\frac{\ell}{\varepsilon}\right)^{O\left(\frac{1}{\varepsilon} \cdot \log\left(\frac{\ell}{1-\delta}\right)\right)}.$$

In particular, if $\delta \in (0,1)$ *and* $\ell \in \mathbb{N}$ *are constant, then the output list size is* $\left(\frac{1}{\varepsilon}\right)^{O(1/\varepsilon)}$*.*

Moreover, there is a randomized algorithm that list recovers $\mathrm{FRS}_{q,s}(n,d)$ *with the above parameters in time* $\mathsf{poly}(\log q, s, d, n, L)$*.*

Remark 3.11. Note that also with respect to input list size ℓ, the above theorem reduces the output list size to quasi-logarithmic in ℓ, whereas Theorem 3.6 only gave a bound that is exponential in ℓ.

The proof of Lemma 3.9 is identical to that of Lemma 3.1, except that we obtain a better bound on the probability of the event E_2 using the following theorem from [GK16b].

Theorem 3.12 ([GK16b], Theorem 14)**.** *Let q be a prime power, and let s, d, n be nonnegative integers such that $n \leq (q-1)/s$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n,d)$. Let $V \subseteq \mathrm{FRS}_{q,s}(n,d)$ be an $\mathbb{F}_q$-linear subspace of dimension $r \leq s$. For $i \in [n]$, let*

$$H_i = \left\{ v \in (\mathbb{F}_q^s)^n \mid v_i = 0 \right\}.$$

Then

$$\mathbb{E}_{i \in [n]} \left[\dim(V \cap H_i) \right] \leq \frac{1-\delta}{1 - r/s} \cdot r.$$

Proof of Lemma 3.9. We first use the above Theorem 3.12 to deduce a better bound on the probability of the event E_2, compared to the bound given in Claim 3.3. As in the proof of Claim 3.3, for $i \in [n]$, let

$$H_i := \{ v \in (\mathbb{F}_q^s)^n \mid v_i = 0 \},$$

and for $j = 0, 1, \ldots, t$, let

$$V_j := V \cap H_{i_1} \cap H_{i_2} \cap \ldots \cap H_{i_j},$$

and $r_j := \dim_{\mathbb{F}_q}(V_j)$. Recall once more that $r = r_0 \geq r_1 \geq \ldots \geq r_t$, and that event E_2 holds if and only if $r_t > 0$.

By Theorem 3.12,

$$\mathbb{E}\left[r_{j+1} \mid r_j = r' \right] = \mathbb{E}_{i \in [n]} \left[\dim(V_j \cap H_i) \mid \dim(V_j) = r' \right] \leq \frac{1-\delta}{1 - r'/s} \cdot r' \leq \frac{1-\delta}{1 - r/s} \cdot r'.$$

Thus

$$\mathbb{E}\left[r_{j+1} \right] \leq \mathbb{E}\left[r_j \right] \cdot \frac{1-\delta}{1 - r/s},$$

and

$$\mathbb{E}\left[r_t \right] \leq \mathbb{E}\left[r_0 \right] \cdot \left(\frac{1-\delta}{1 - r/s} \right)^t = (1-\delta)^t \cdot \frac{r}{(1 - r/s)^t}.$$

Finally, by Markov's inequality this implies in turn that

$$\Pr\left[E_2 \right] = \Pr\left[r_t > 0 \right] \leq (1-\delta)^t \cdot \frac{r}{(1 - r/s)^t} \leq (1-\delta)^t \cdot \frac{r}{(1 - \varepsilon/4)^t},$$

where the last inequality follows by assumption that $r \leq \frac{\varepsilon s}{4}$.

Plugging the above bound in the proof of Lemma 3.2, we obtain that any codeword $c \in V$ with $\mathrm{dist}(c, S) \leq \delta - \varepsilon$ appears in $\hat{\mathcal{L}}$ with probability at least

$$(1 - \delta + \varepsilon)^t - (1-\delta)^t \cdot \frac{r}{(1 - \varepsilon/4)^t}. \tag{3}$$

Setting $t := \frac{4 \log r}{\varepsilon}$ in Lemma 3.2, we have that $|\hat{\mathcal{L}}| \leq \ell^t \leq \ell^{O((\log r)/\varepsilon)}$. Moreover, using (3) any

codeword in the output list $\mathcal{L}$ of C appears in $\hat{\mathcal{L}}$ with probability at least

$$\begin{aligned} p_0 : &= (1-\delta+\varepsilon)^t - (1-\delta)^t \cdot \frac{r}{(1-\varepsilon/4)^t} \\ &\geq (1-\delta+\varepsilon)^t - \frac{1}{2} \cdot \left(\frac{1+\varepsilon/4}{1-\varepsilon/4} \cdot (1-\delta)\right)^t \\ &\geq \frac{1}{2}(1-\delta+\varepsilon)^t \\ &\geq (1-\delta)^{O((\log r)/\varepsilon)}, \end{aligned}$$

where the first inequality holds for sufficiently small $\varepsilon > 0$ by our choice of $t = \frac{4 \log r}{\varepsilon}$.

Finally, this implies in turn that $|\mathcal{L}| \leq \frac{|\hat{\mathcal{L}}|}{p_0} \leq \left(\frac{\ell}{1-\delta}\right)^{O((\log r)/\varepsilon)}$, as well as the claimed running time. □

4 Local list decoding multivariate multiplicity codes

In this section we use our results from Section 3 on *global* list decoding of *univariate* multiplicity codes with constant list size to show that *multivariate* multiplicity codes can be *locally* list decoded up to their minimum distance. As before, we show a more general version that applies also to list recovery.

Theorem 4.1 (Local list recovering up to minimum distance for MultiMult)**.** *There exists an absolute constant $c_0 > 0$ so that the following holds. Let q be a prime, let s, d, m be nonnegative integers, and let $\delta := 1 - \frac{d}{sq}$. Let $\varepsilon > 0$ and $\ell, L \in \mathbb{N}$ be such that $s \geq \frac{64\ell}{\varepsilon^2}$, $L \geq \left(\frac{1}{1-\delta}\right)^{\left(\frac{s\ell}{\varepsilon}\right)^{c_0}}$, and $q \geq \max\{10m, \exp(L)\}$. Then the multivariate multiplicity code $\mathsf{MULT}_{q,s}^{(m)}(d)$ is $(t, \delta - \varepsilon, \ell, L)$-locally list recoverable for $t = q^{O(1)} \cdot \delta^{-O(m)} \cdot \exp(m \cdot L)$.*

The above theorem is a consequence of the following lemma which relates the parameters of the global list recovery algorithm for univariate multiplicity codes to that of the local list recovery algorithm for the corresponding multivariate multiplicity codes.

Lemma 4.2. *Let q be a prime power, let s, d, m be nonnegative integers, and let $\delta := 1 - \frac{d}{sq}$. Let $\alpha \in (0, \delta)$ and $\ell \in \mathbb{N}$, and suppose that the univariate multiplicity code $\mathsf{MULT}_{q,s}^{(1)}(d)$ is (α, ℓ, L)-(globally) list recoverable and $(\alpha, 100L, L')$-(globally) list recoverable. Let $\varepsilon > 0$ be a parameter, and suppose that $q \geq \max\{10m, 100^2 \frac{s \cdot L \cdot L'}{\alpha^3 \cdot \varepsilon^2}\}$. Then the multivariate multiplicity code $\mathsf{MULT}_{q,s}^{(m)}(d)$ is $(t, \alpha - \varepsilon, \ell, O(L))$-locally list recoverable for $t = q^{O(1)} \cdot \left(\frac{s \cdot L \cdot L'}{\alpha}\right)^{O(m)}$.*

Instantiating the above lemma with the global list recovery algorithm for univariate multiplicity codes from Theorem 3.8 gives our main Theorem 4.1. The rest of this section is devoted to the proof of Lemma 4.2. We first give a short overview of the approach, and then flesh out the details in the subsequent three subsections.

The local list recovering algorithm has three main subroutines that we will describe and analyze in the next three subsections. Briefly, the three components are the following:

1. A subroutine RecoverCandidates, given in Section 4.1. RecoverCandidates takes as input a point $\mathbf{a} \in \mathbb{F}_q^m$, has query access to the tuple of input lists S, and returns a short list Z of guesses for $Q^{(<\tilde{s})}(\mathbf{a})$, where we will take $\tilde{s}$ to be some parameter larger than s.

2. A (deterministic) oracle machine $M^S[\mathbf{a}, z]$ to evaluations of $Q^{(<s)}$, given in Section 4.2. The oracle machine $M^S[\mathbf{a}, z]$ is defined using an advice string $(\mathbf{a}, z)$, has query access to S, and with high probability over the choice of a random point $\mathbf{a}$, we will have that $M^S[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})]$ recovers most of the points of $Q^{(<s)}$.

3. The final local list recovery algorithm LocalListRecoverMULT, given in Section 4.3. Recall that the goal is to output a list of randomized algorithms A_j so that for each codeword $c \in \mathsf{MULT}_{q,s}^{(m)}(d)$ that agrees with many of the input lists in S, with probability at least $2/3$, there exists some A_j so that for any coordinate i, $\Pr[A_j(i) = c_i] \geq 2/3$. We arrive at these algorithms A_j as follows.

   First, the algorithm runs RecoverCandidates on a random point $\mathbf{a} \in \mathbb{F}_q^m$ to generate a short list Z of possibilities for $Q^{(<\tilde{s})}(\mathbf{a})$. Then, for each $z \in Z$, it forms the oracle machine $M^S[\mathbf{a}, z]$. At this point it would be tempting to output the list of these oracle machines, but we are not quite done: even if $z = Q^{(<\tilde{s})}(\mathbf{a})$ corresponds to the correct advice and the choice of $\mathbf{a}$ is good, for some small fraction of points $\mathbf{x}$, we may still have $M^S[\mathbf{a}, z](\mathbf{x}) \neq Q^{(<s)}(\mathbf{x})$. Fortunately, for most $\mathbf{x}$ this will not be the case, and so we can implement the local correction algorithm of [KSY14] for multiplicity codes on top of $M^S[\mathbf{a}, z]$. This will give us our final list of randomized algorithms A_j that the local list recovery algorithm returns.

4.1 The algorithm RecoverCandidates

As an important subroutine of the local list recovering algorithm, we will implement an algorithm which we call RecoverCandidates. The main feature of this algorithm is that given oracle access to small lists, that for most coordinates agree with the evaluations of order s derivatives of Q, it can output for most coordinates, a small list that agrees with evaluations of order $\tilde{s}$ derivatives of Q (think of $\tilde{s}$ to be much larger than s).

Specifically, the algorithm RecoverCandidates will have oracle access to a function $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$. Think of this function as assigning to each element of $\mathbb{F}_q^m$ a list of size ℓ of alphabet symbols of the multiplicity code. Now suppose that Q is an m-variate polynomial of degree at most d (think of Q to represent a true codeword of the multiplicity code) that "agrees" with at least $1 - \alpha + \varepsilon$ fraction of these lists. On being input $\mathbf{x}$, a random element of $\mathbb{F}_q^m$, and for some parameter $\tilde{s}$, the algorithm RecoverCandidates will make few queries to S and output a small list $Z \subseteq \Sigma_{m,\tilde{s}}$, such that with high probability (over the choice of $\mathbf{x}$ and the randomness of the algorithm), the list Z contains $Q^{(<\tilde{s})}(\mathbf{x})$.

Lemma 4.3. *Let q be a prime power, and let s, d, m be nonnegative integers. Let $\alpha \in (0, 1)$ and $\ell \in \mathbb{N}$, and suppose that $\mathsf{MULT}_{q,s}^{(1)}(d)$ is (α, ℓ, L)-(globally) list recoverable. Let $\tilde{s} > 0$ be a parameter, and suppose that $q > 100 \cdot L \cdot \tilde{s}$. Let $\varepsilon > 0$ be a parameter.*

Let $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$, and suppose that $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ is a polynomial of degree

at most d such that:

$$\Pr_{\mathbf{x}\in\mathbb{F}_q^m}\left[Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x})\right] > 1-\alpha+\varepsilon.$$

There is an algorithm RecoverCandidates *which on input* $\mathrm{x} \in \mathbb{F}_q^m$*, and given oracle access to* S*, makes at most* $q\cdot(L\tilde{s})^{O(m)}$ *queries to* S*, and outputs a list* $Z \subseteq \Sigma_{m,\tilde{s}}$ *of size at most* $100L$ *such that*

$$\Pr\left[Q^{(<\tilde{s})}(\mathbf{x}) \in Z\right] \geq 1-\frac{1}{\varepsilon^2 q},$$

where the probability is over the choice of a uniform random $\mathbf{x}\in\mathbb{F}_q^m$ *and the random choices of the algorithm* RecoverCandidates*.*

The high level idea of the algorithm is as follows. On input $\mathbf{x}$, we take several random lines passing through $\mathbf{x}$, and run the univariate multiplicity list recovery algorithm on the restrictions of the input lists to those lines. This gives us, for each of these lines, a list of univariate polynomials. For a given line, this list of univariate polynomials contains candidates for Q restricted to that line. In particular, this gives us candidate values for $Q(\mathbf{x})$ and the all higher order directional derivatives of Q at $\mathbf{x}$ in the directions of those lines. We combine this information about the different directional derivatives to reconstruct $Q^{(<\tilde{s})}(\mathbf{x})$. This combination turns out to be a certain kind of polynomial list recovery problem: namely list recovery for tuples of polynomials.

Lemma 4.4 (Vector-valued Reed-Muller list recovery on a grid)**.** *Let* d, m, ℓ, K *be given parameters. Let* $\mathbb{F}$ *be a finite field. Suppose that* $U \subseteq \mathbb{F}$ *and* $|U| \geq 2d\ell K$*. Let* $\alpha < 1-\frac{1}{\sqrt{K}}$ *be a parameter.*

Then for every $f : U^m \to \binom{\mathbb{F}^t}{\ell}$*, if*

$$\mathcal{L} = \Big\{(Q_1, Q_2, \ldots, Q_t) \in (\mathbb{F}[Y_1, Y_2, \ldots, Y_m])^t \mid \forall i \in [t], \deg(Q_i) \leq d \text{ and } \Pr_{\mathbf{u}\in U^m}\left[(Q_1(\mathbf{u}), Q_2(\mathbf{u}), \ldots, Q_t(\mathbf{u})) \notin f(\mathbf{u})\right] < \alpha\Big\},$$

then $|\mathcal{L}| \leq 2K\ell$*.*

Proof. We shall reduce the case of tuples of polynomials to the case of a single polynomial over a large field. Specifically, let $\mathbb{K}$ be the degree t field extension of $\mathbb{F}$, and let $\phi : \mathbb{F}^t \to \mathbb{K}$ be an arbitrary $\mathbb{F}$-linear bijection. Then to every function $f : U^m \to \mathbb{F}^t$, we can associate a function $\tilde{f} : U^m \to \mathbb{K}$, where $\tilde{f} = \phi \circ f$. This identifies the underlying Hamming metric spaces. The key observation is that under this identification, f is the evaluation table of a tuple of t polynomials in $\mathbb{F}[X_1, \ldots, X_m]$ of degree $\leq d$ if and only if $\tilde{f} : U^m \to \mathbb{K}$ is the evaluation table of a degree $\leq d$ polynomial in $\mathbb{K}[X_1, \ldots, X_m]$.

The bound on $|\mathcal{L}|$ then follows from the Johnson bound for list recovery, which is a general statement implying good list recoverability for codes with large distance. Specifically, Lemma V.2 in [GKO+18] (see also Corollary 3.7 in [Gur04]) states that a code of relative distance δ is (α, ℓ, L) list recoverable for $\alpha > 1-\sqrt{\ell(1-\delta)}$ and $L = \frac{\ell}{(1-\alpha)^2-\ell(1-\delta)}$. In our setting, the code of polynomials of degree at most d on U^m has relative distance δ at least $1-\frac{d}{|U|} \geq 1-\frac{1}{2K\ell}$. Thus for $\alpha < 1-\frac{1}{\sqrt{K}}$, the Johnson bound for list recovery implies that we can take $L = 2K\ell$, as desired. □

To see why the above lemma is relevant, we note the following chain rule for Hasse derivatives.

Claim 4.5. *Let $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$, let $\mathbf{x}, \mathbf{b} \in \mathbb{F}_q^m$, and let $\lambda(T) = \mathbf{x} + T\mathbf{b}$ be the line passing through $\mathbf{x}$ in direction $\mathbf{b}$. Then for any integer $j \geq 0$,*

$$(Q \circ \lambda)^{(j)}(0) = \sum_{\mathbf{j}:\mathsf{wt}(\mathbf{j})=j} Q^{(\mathbf{j})}(\mathbf{x})\mathbf{b}^{\mathbf{j}}.$$

Proof. We have from the definition of the Hasse derivative that

$$\begin{aligned}
\sum_i (Q \circ \lambda)^{(i)}(0) \cdot T^i &= (Q \circ \lambda)(T) \\
&= Q(\mathbf{x} + T\mathbf{b}) \\
&= \sum_{\mathbf{i}} Q^{(\mathbf{i})}(\mathbf{x}) \cdot \mathbf{b}^{\mathbf{i}} \cdot T^{\mathsf{wt}(\mathbf{i})} \\
&= \sum_i \left(\sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} Q^{(\mathbf{i})}(\mathbf{x})\mathbf{b}^{\mathbf{i}} \right) T^i,
\end{aligned}$$

and so by matching coefficients we have

$$(Q \circ \lambda)^{(j)}(0) = \sum_{\mathbf{j}:\mathsf{wt}(\mathbf{j})=j} Q^{(\mathbf{j})}(\mathbf{x})\mathbf{b}^{\mathbf{j}}.$$

□

Now let $\tilde{s}$ be an integer, and for each $0 \leq j < \tilde{s}$, define the polynomial

$$H_{\mathbf{x},j}(Y_1, \ldots, Y_m) = \sum_{\mathbf{j}:\mathsf{wt}(\mathbf{j})=j} Q^{(\mathbf{j})}(\mathbf{x})\mathbf{Y}^{\mathbf{j}},$$

and let $H_{\mathbf{x}}(Y_1, \ldots, Y_m) \in (\mathbb{F}_q[Y_1, \ldots, Y_m])^{\tilde{s}}$ be the tuple of polynomials:

$$H_{\mathbf{x}} = (H_{\mathbf{x},0}, H_{\mathbf{x},1}, \ldots, H_{\mathbf{x},\tilde{s}-1}).$$

Then we have

$$(Q \circ \lambda)^{(<\tilde{s})}(0) = H_{\mathbf{x}}(\mathbf{b}). \tag{4}$$

Thus, given information about $(Q \circ \lambda)^{(<\tilde{s})}(0)$ for various lines λ and for some $\tilde{s}$, we have information about the tuple of polynomials $H_{\mathbf{x}}(\mathbf{Y})$, evaluated at many different points $\mathbf{b}$. It is on these polynomials that we will use Lemma 4.4. The polynomials $H_{\mathbf{x}}(\mathbf{Y})$ will let us in turn to reconstruct $Q^{(<\tilde{s})}(\mathbf{x})$.

Now we present our main subroutine RecoverCandidates, and analyze it below. For an element $z \in \Sigma_{m,s}$, and a direction $\mathbf{b} \in \mathbb{F}_q^m$, we define the restriction of z to direction $\mathbf{b}$ (denoted $z|_{\mathbf{b}}$) to equal $h \in \Sigma_{1,s}$, given by:

$$h^{(j)} = \sum_{\mathsf{wt}(\mathbf{j})=j} z^{(\mathbf{j})}\mathbf{b}^{\mathbf{j}}$$

for each j such that $0 \leq j < s$.

Main Subroutine RecoverCandidates**.**

- Oracle access to $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$.
- **INPUT:** $\mathbf{x} \in \mathbb{F}_q^m$, parameter $\tilde{s} \in \mathbb{N}$.
- The goal is to recover a small list of candidates for $Q^{(<\tilde{s})}(\mathbf{x})$.

1. Let $U \subseteq \mathbb{F}_q$ be a set of size $100\tilde{s}L$.
2. Let $\mathbf{b} \in \mathbb{F}_q^m$ be picked uniformly at random.
3. Let $B = \{\mathbf{b_u} = \mathbf{b} + \mathbf{u} \mid \mathbf{u} \in U^m\}$.
4. For each $\mathbf{u} \in U^m$:
   (a) Let $\lambda_{\mathbf{u}}(T)$ be the line $\lambda_{\mathbf{u}}(T) = \mathbf{x} + T\mathbf{b_u}$.
   (b) Consider the restriction $S_{\mathbf{u}} : \mathbb{F}_q \to \binom{\Sigma_{1,s}}{\ell}$ of S to $\lambda_{\mathbf{u}}$. Formally:
   $$S_{\mathbf{u}} = S \circ \lambda_{\mathbf{u}}(T) = \{z|_{\mathbf{b_u}} \mid z \in S(\lambda_{\mathbf{u}}(T))\}.$$
   (c) Run the univariate list recovery algorithm on $S_{\mathbf{u}}$ with error-tolerance α for degree d polynomials to obtain a list $\mathcal{L}_{\lambda_{\mathbf{u}}} \subseteq \mathbb{F}_q[T]$.
5. Define a function $f : U^m \to \binom{\mathbb{F}_q^{\tilde{s}}}{L}$ as follows. For each $\mathbf{u} \in U^m$, define
   $$f(\mathbf{u}) = \left\{P^{(<\tilde{s})}(0) \mid P(T) \in \mathcal{L}_{\lambda_{\mathbf{u}}}\right\}.$$
6. Let $\mathcal{L}'$ be the set of all $\tilde{s}$-tuples of polynomials
   $$(Q'_j(Y_1, \ldots, Y_m))_{j=0}^{\tilde{s}-1}$$
   where Q'_j is homogeneous of degree j, and such that
   $$(Q'_j(\mathbf{u}))_{j=0}^{\tilde{s}-1} \in f(\mathbf{u})$$
   for at least $1/4$ fraction of the $\mathbf{u} \in U^m$.
   Obtain this list $\mathcal{L}'$ by applying Lemma 4.4 for $\tilde{s}$-tuples of polynomials of degree $\leq \tilde{s}$, where the evaluation points are U^m. Then prune the resulting list $\mathcal{L}$ to ensure that for each member
   $$(Q'_j(Y_1, \ldots, Y_m))_{j=0}^{\tilde{s}-1}$$
   of the list $\mathcal{L}$, and for each j such that $0 \leq j \leq \tilde{s}-1$, Q'_j is homogeneous of degree j. This pruned list is $\mathcal{L}'$.

7. For each
$$(Q'_j(\mathbf{Y}))_{j=0}^{\tilde{s}-1} \in \mathcal{L}',$$
let
$$(P_j(\mathbf{Y}))_{j=0}^{\tilde{s}-1} = (Q'_j(\mathbf{Y}-\mathbf{b}))_{j=0}^{\tilde{s}-1},$$
and add this to a new list of tuples of polynomials that we call $\mathcal{L}''$.

8. Let
$$Z = \left\{ z \in \Sigma_{m,\tilde{s}} \mid (\sum_{\mathsf{wt}(\mathbf{j})=j} z^{(\mathbf{j})}\mathbf{Y}^{\mathbf{j}})_{j=0}^{\tilde{s}-1} \in \mathcal{L}'' \right\}.$$

9. Return Z.

We now prove Lemma 4.3.

Proof of Lemma 4.3. Suppose $Q(X_1,\ldots,X_m) \in \mathbb{F}_q[X_1,\ldots,X_m]$ is a polynomial of degree at most d such that:
$$\Pr_{\mathbf{x}\in\mathbb{F}_q^m}[Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x})] > 1-\alpha+\varepsilon.$$

For each line λ in $\mathbb{F}_q^m$, let $\mathcal{L}_\lambda$ be the result of univariate list recovering $S\circ\lambda$ with error-parameter α. Let $\mathbf{b}\in\mathbb{F}_q^m$ be the random choices of RecoverCandidates. For $\mathbf{x},\mathbf{b}\in\mathbb{F}_q^m$, and for $\mathbf{u}\in U^m$, recall that $\lambda_{\mathbf{u}}(T)$ denotes the line $\lambda_{\mathbf{u}}(T) = \mathbf{x}+T\mathbf{b}_{\mathbf{u}}$, where $\mathbf{b}_{\mathbf{u}} = \mathbf{b}+\mathbf{u}$. Then for each $\mathbf{u}\in U^m$, let $B_{\mathbf{u}}$ be the event that $Q\circ\lambda_{\mathbf{u}}(T)$ is not in $\mathcal{L}_{\lambda_{\mathbf{u}}}$.

Claim 4.6. *For each fixed* $\mathbf{u}\in U^m$,
$$\Pr_{\mathbf{x},\mathbf{b}\in\mathbb{F}_q^m}[B_{\mathbf{u}}] \le \frac{1}{4\varepsilon^2 q}.$$

Proof. Note that when $\mathbf{x},\mathbf{b}$ are uniformly random elements of $\mathbb{F}_q^m$, then $\lambda_{\mathbf{u}}(T)$ is a uniformly random line. The event that $Q\circ\lambda_{\mathbf{u}}(T)$ is not in $\mathcal{L}_{\lambda_{\mathbf{u}}}$ is a subset of the event that $Q^{(<s)}(\mathbf{x})\notin S(\mathbf{x})$ for more than α-fraction of points $\mathbf{x}$ on the line $\lambda_{\mathbf{u}}(T)$. The claim then follows from a standard application of Chebyshev's inequality, using the fact that the points on a uniformly random line are pairwise independent.

More precisely,
$$\begin{aligned}
\Pr_{\mathbf{x},\mathbf{b}\in\mathbb{F}_q^m}[B_{\mathbf{u}}] &\le \Pr_{\mathbf{x},\mathbf{b}\in\mathbb{F}_q^m}\left[\sum_{t\in\mathbb{F}_q}\mathbf{1}\left\{Q^{(<s)}(\lambda_{\mathbf{u}}(t))\notin S(\lambda_{\mathbf{u}}(t))\right\} > \alpha q\right] \\
&= \Pr_{Y}\left[\sum_{t\in\mathbb{F}_q} Y_t > \alpha q\right] \\
&\le \Pr_{Y}\left[\sum_{t\in\mathbb{F}_q}(Y_t-\mathbb{E}Y_t) > \varepsilon\cdot q\right],
\end{aligned}$$

where the Y_t are pairwise independent $\{0,1\}$-valued random variables with $\mathbb{E}Y_t \leq \alpha - \varepsilon$. Then by Chebyshev's inequality, this last quantity is at most

$$\frac{\sum_{t\in\mathbb{F}_q} \mathbb{E}(Y_t - \mathbb{E}Y_t)^2}{\varepsilon^2 q^2} \leq \frac{1}{4\varepsilon^2 q}.$$

□

Claim 4.7.

$$\Pr_{\mathbf{x},\mathbf{b}\in\mathbb{F}_q^m}\left[\sum_{\mathbf{u}\in U^m} \mathbf{1}_{B_\mathbf{u}} > \frac{|U|^m}{4}\right] < \frac{1}{\varepsilon^2 q}.$$

Proof. The proof is immediate from the previous claim and Markov's inequality. □

Thus we conclude that with probability at least $1 - \frac{1}{\varepsilon^2 q}$, when $\mathbf{x}$ is a uniformly random element of $\mathbb{F}_q^m$, for at least $1/4$ of the $\mathbf{u} \in U^m$, we have that

$$Q \circ \lambda_\mathbf{u}(T) \in \mathcal{L}_{\lambda_\mathbf{u}}.$$

We assume that this happens, and let $G \subseteq U^m$ be this set of $\mathbf{u}$.

Recall that $\mathcal{L}_{\lambda_\mathbf{u}}$ is a list of size L. Consider the function

$$f : U^m \to \binom{\mathbb{F}_q^{\tilde{s}}}{L},$$

where for each $\mathbf{u} \in U^m$,

$$f(\mathbf{u}) = \{P^{(<\tilde{s})}(0) \mid P(T) \in \mathcal{L}_{\lambda_\mathbf{u}}\}.$$

Fix any $\mathbf{u} \in G$. Then since

$$Q \circ \lambda_\mathbf{u}(T) \in \mathcal{L}_{\lambda_\mathbf{u}},$$

it holds that

$$(Q \circ \lambda_\mathbf{u})^{(<\tilde{s})}(0) \in f(\mathbf{u}).$$

Now observe that by (4), we have

$$H_\mathbf{x}(\mathbf{b}_\mathbf{u}) = (Q \circ \lambda_\mathbf{u})^{(<\tilde{s})}(0),$$

and thus

$$H_\mathbf{x}(\mathbf{b} + \mathbf{u}) = H_\mathbf{x}(\mathbf{b}_\mathbf{u}) \in f(\mathbf{u}).$$

Since this happens for each $\mathbf{u} \in G$, we have that this happens for at least $1/4$ fraction of $\mathbf{u} \in U^m$.

Now by our assumption that $|U| \geq 100L\tilde{s}$, Lemma 4.4 implies that $H_\mathbf{x}(\mathbf{b} + \mathbf{Y})$ is included in $\mathcal{L}'$ (here we also use the fact that $H_{\mathbf{x},j}$ is an homogeneous m-variate polynomial of degree j). In this event, $\mathcal{L}''$ will contain

$$H_\mathbf{x}(\mathbf{Y})$$

and then it follows that in Step 8 of RecoverCandidates, the list Z will contain $Q^{(<\tilde{s})}(\mathbf{x})$.

It remains to show the claimed output list size and query complexity. To this end, note first that the output list size is at most the size of $\mathcal{L}$, which is at most $100L$ by Lemma 4.4. The algorithm RecoverCandidates runs the global list recovery algorithm for the univariate multiplicity codes on $|U|^m = (L\tilde{s})^{O(m)}$ lines, and so the query complexity is at most $q \cdot (L\tilde{s})^{O(m)}$. □

4.2 The Oracle Machine M

The oracle machine M will output a short list of oracle machines, each of which is defined by a piece of advice. In this case, the advice will be a point $\mathbf{a} \in \mathbb{F}_q^m$, and $z \in \Sigma_{m,\tilde{s}}$, which is meant to be a guess for $Q^{(<\tilde{s})}(\mathbf{a})$. Given this advice, the oracle machine works as follows: on input $\mathbf{x}$, with corresponding input list $S(\mathbf{x})$, it will run the univariate list recovery algorithm on the line $\lambda(T) = \mathbf{x} + T(\mathbf{a} - \mathbf{x})$ through $\mathbf{x}$ and $\mathbf{a}$ to obtain a list $\mathcal{L}$ of univariate polynomials $P(T)$. We will show that with high probability (assuming the advice is good), there will be a unique polynomial $P(T)$ in $\mathcal{L}$ so that both $P^{(<\tilde{s})}(1)$ is consistent with z, and $P^{(<s)}(0)$ is consistent with some element of $S(\mathbf{x})$. Then the oracle machine will output the symbol in $S(\mathbf{x})$ that $P^{(<s)}(0)$ agrees with.

The key later will be that the advice z will not vary over all possibilities in $\Sigma_{m,\tilde{s}}$; this would result in too long a list. Rather, we will use RecoverCandidates in order to generate this advice.

Formally, we will prove the following lemma about our oracle machine, which we define below.

Lemma 4.8. *Let q be a prime power, and let s, d, m be nonnegative integers such that $d < sq$. Let $\alpha \in (0,1)$ and $\ell \in \mathbb{N}$, and suppose that $\mathsf{MULT}_{q,s}^{(1)}(d)$ is (α, ℓ, L)-(globally) list recoverable. Let $\tilde{s} > 0$ and $\varepsilon > 0$ be parameters.*

Let $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$, and suppose that $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ is a polynomial of degree at most d such that:

$$\Pr_{\mathbf{x} \in \mathbb{F}_q^m}\left[Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x})\right] > 1 - \alpha + \varepsilon.$$

There is an algorithm $M^S[\mathbf{a}, z](\mathbf{x})$ which on input $\mathbf{x} \in \mathbb{F}_q^m$, given as advice a point $\mathbf{a} \in \mathbb{F}_q^m$, and $z \in \Sigma_{m,\tilde{s}}$, and given oracle access to S, makes at most q queries to S, and outputs an element of $\Sigma_{m,s} \cup \{\perp\}$ such that if $\mathbf{x}, \mathbf{a} \in \mathbb{F}_q^m$ are chosen uniformly at random, then:

$$\Pr_{\mathbf{a},\mathbf{x} \in \mathbb{F}_q^m}\left[M^S[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})](\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \geq 1 - \gamma$$

for

$$\gamma = \alpha - \varepsilon + \frac{\ell s}{q} + \frac{1}{\varepsilon^2 q} + \frac{sL}{\tilde{s}}.$$

We will first decribe the algorithm and then show that it satisfies the required properties.

Oracle machine M.

- Oracle access to $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$.
- **INPUT:** $\mathbf{x} \in \mathbb{F}_q^m$.
- **ADVICE:** Point $\mathbf{a} \in \mathbb{F}_q^m$, and $z \in \Sigma_{m,\tilde{s}}$.

1. Set $\mathbf{b}_* = \mathbf{a} - \mathbf{x}$.
2. Let $\lambda_{\mathbf{b}_*}$ be the line $\lambda_{\mathbf{b}_*}(T) = \mathbf{x} + T\mathbf{b}_*$.

3. Consider the restriction $S_{\mathbf{b}_*} : \mathbb{F}_q \to \binom{\Sigma_{1,s}}{\ell}$ of S to the line $\lambda_{\mathbf{b}_*}$, and list recover this with error-tolerance α for degree d polynomials, and obtain the list $\mathcal{L}_{\lambda_{\mathbf{b}_*}} \subseteq \mathbb{F}_q[T]$.

4. If there exists exactly one $P(T) \in \mathcal{L}_{\lambda_{\mathbf{b}_*}}$ such that $P^{(<\tilde{s})}(1) = z|_{\mathbf{b}_*}$, then set $P_{\mathbf{b}_*}(T)$ to equal that $P(T)$, otherwise output $\perp$ and exit.

5. If there exists exactly one $y \in S(\mathbf{x})$ for which $y|_{\mathbf{b}_*} = P_{\mathbf{b}_*}^{(<s)}(0)$, then output that y.

6. Otherwise output $\perp$.

We will now analyze the above algorithm and show that is satisfies the required properties.

Proof of Lemma 4.8. By the description of the oracle machine, it is clear that it makes at most q queries.

It remains to show that when $\mathbf{a}$ and $\mathbf{x}$ are chosen uniformly at random from $\mathbb{F}_q^m$, then

$$\Pr_{\mathbf{a},\mathbf{x}\in\mathbb{F}_q^m}\left[M^S[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})](\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \geq 1 - \gamma.$$

Claim 4.9. *Let $y_0 = Q^{(<s)}(\mathbf{x})$. With probability at least $1 - \alpha + \varepsilon$ over the random choice of $\mathrm{x} \in \mathbb{F}_q^m$, $y_0 \in S(\mathbf{x})$.*

Proof. The proof is immediate since it is given to us that

$$\Pr_{\mathbf{x}\in\mathbb{F}_q^m}[Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x})] > 1 - \alpha + \varepsilon.$$

□

Claim 4.10. *Let $y_0 = Q^{(<s)}(\mathbf{x})$. For any $y \in S(\mathbf{x})$ with $y \neq y_0$, with probability at least $1 - \frac{s}{q}$ over the random choice of $\mathbf{a} \in \mathbb{F}_q^m$, we have that*

$$y|_{\mathbf{b}_*} \neq y_0|_{\mathbf{b}_*}.$$

Proof. Recall that by definition, for an element $z \in \Sigma_{m,s}$, and a direction $\mathbf{b} \in \mathbb{F}_q^m$, $z|_{\mathbf{b}}$ is to equal $h \in \Sigma_{1,s}$, where:

$$h^{(j)} = \sum_{\mathsf{wt}(\mathbf{j})=j} z^{(\mathbf{j})}\mathbf{b}^{\mathbf{j}}$$

for each j such that $0 \leq j < s$. Note that $h^{(j)}$ can be viewed as a polynomial of degree at most j evaluated at $\mathbf{b}$, where the coefficients of the polynomial depend only on z.

Since $y \neq y_0$, the corresponding tuples of polynomials (each of degree at most s) will differ in at least one coordinate. Observe also that for any fixed choice of $\mathbf{x}$, the randomness of $\mathbf{a}$ implies that $\mathbf{b}_*$ is a uniformly random element of $\mathbb{F}_q^m$. Thus in the coordinate where the tuples of polynomials differ, the evaluations at $\mathbf{b}_*$ will be distinct with probability at least $1 - \frac{s}{q}$ by the Schwartz-Zippel Lemma.

Thus with probability at least $1 - \frac{s}{q}$ over the random choice of $\mathbf{a} \in \mathbb{F}_q^m$, we have that $y|_{\mathbf{b}_*} \neq y_0|_{\mathbf{b}_*}$. □

Claim 4.11. *Let $y_0 = Q^{(<s)}(\mathbf{x})$. Then with probability at least $1 - \frac{\ell s}{q}$ over the random choice of $\mathbf{a} \in \mathbb{F}_q^m$, y_0 is the unique element y of $S(\mathbf{x})$ for which $y|_{\mathbf{b}*} = Q \circ \lambda_{\mathbf{b}_*}^{(<s)}(0)$.*

Proof. Clearly, by definition, $y_0|_{\mathbf{b}*} = Q \circ \lambda_{\mathbf{b}_*}^{(<s)}(0)$. Also, taking a union bound over all ℓ elements of $S(\mathbf{x})$, by Claim 4.10, $y_0|_{\mathbf{b}*} \neq y|_{\mathbf{b}_*}$ for all other $y \in S(\mathbf{x})$ with probability at least $1 - \frac{\ell s}{q}$. □

Claim 4.9 and Claim 4.11 together imply that with probability at least $1 - \left(\alpha - \varepsilon + \frac{\ell s}{q}\right)$ over the random choice of $\mathbf{a}$ and $\mathbf{x} \in \mathbb{F}_q^m$, $Q^{(<s)}(\mathbf{x})$ is the unique element y of $S(\mathbf{x})$ for which $y|_{\mathbf{b}*} = Q \circ \lambda_{\mathbf{b}_*}^{(<s)}(0)$.

We will now show that with probability at least $1 - \left(\frac{1}{\varepsilon^2 q} + \frac{sL}{\tilde{s}}\right)$ over the random choice of $\mathbf{a}$ and $\mathbf{x} \in \mathbb{F}_q^m$, $P_{\mathbf{b}_*}(T) = Q \circ \lambda_{\mathbf{b}_*}(T)$. Once we will have this, then it will immediately follow that the algorithm will output $Q^{(<s)}(\mathbf{x})$ with probability at least $1 - \left(\alpha - \varepsilon + \frac{\ell s}{q} + \frac{1}{\varepsilon^2 q} + \frac{sL}{\tilde{s}}\right)$ over the random choice of $\mathbf{a}$ and $\mathbf{x} \in \mathbb{F}_q^m$

For each line λ in $\mathbb{F}_q^m$, let $\mathcal{L}_\lambda$ be the result of list recovering $S \circ \lambda$ with error-parameter α. For points $\mathbf{x}$ and $\mathbf{a}$ picked uniformly at random from $\mathbb{F}_q^m$, let $\mathbf{b}_* = \mathbf{a} - \mathbf{x}$, and let $\lambda_{\mathbf{b}_*}$ be the line $\lambda_{\mathbf{b}_*}(T) = \mathbf{x} + T\mathbf{b}_*$.

Let $B_{\lambda_{\mathbf{b}_*}}$ denote the event that $\mathcal{L}_{\lambda_{\mathbf{b}_*}}$ does not contain $Q \circ \lambda_{\mathbf{b}_*}(T)$. Let $C_{\lambda_{\mathbf{b}_*},\mathbf{a}}$ denote the event that there exists $P(T) \in \mathcal{L}_{\lambda_{\mathbf{b}_*}}$ with $P(T) \neq Q \circ \lambda_{\mathbf{b}_*}(T)$, but $P^{(<\tilde{s})}(0) = (Q \circ \lambda_{\mathbf{b}_*})^{(<\tilde{s})}(0)$. Thus $B_{\lambda_{\mathbf{b}_*}}$ is the event that there are too many errors on $\lambda_{\mathbf{b}_*}$. $C_{\lambda_{\mathbf{b}_*},\mathbf{a}}$ is the event that $\mathbf{a}$ is not a disambiguating point.

Claim 4.12.

$$\Pr\left[B_{\lambda_{\mathbf{b}_*}}\right] \leq \frac{1}{\varepsilon^2 q}.$$

Proof. The proof is identical to that of Claim 4.6, and it follows from a standard application of Chebyshev's inequality, using the fact that the points on a uniformly random line are pairwise independent. □

Claim 4.13.

$$\Pr\left[C_{\lambda_{\mathbf{b}_*},\mathbf{a}}\right] \leq \frac{sL}{\tilde{s}}.$$

Proof. Because of the way $\mathbf{x}$, $\mathbf{a}$ and the line $\lambda_{\mathbf{b}_*}$ are sampled, equivalently one could let $\mathbf{x}$ be picked uniformly at random from $\mathbb{F}_q^m$, $\lambda_{\mathbf{b}_*}$ be a uniformly random line through $\mathbf{x}$, and $\mathbf{a}$ be a uniformly random point on $\lambda_{\mathbf{b}_*}$.

Now fix any polynomial $P(T) \in \mathcal{L}_{\lambda_{\mathbf{b}_*}}$ with $P(T) \neq Q \circ \lambda_{\mathbf{b}_*}(T)$. We want to bound the probability that $P^{(<\tilde{s})}(\alpha) = (Q \circ \lambda_{\mathbf{b}_*})^{(<\tilde{s})}(\alpha)$ where α is picked uniformly at random. But P and $Q \circ \lambda_{\mathbf{b}_*}$ are fixed distinct polynomials of degree at most d. Thus the probability that they agree with multiplicity $\tilde{s}$ on a random point of $\mathbb{F}_q$ is at most $\frac{d}{\tilde{s}q} \leq \frac{s}{\tilde{s}}$, where the inequality follows by assumption that $d < sq$.

The result follows from a union bound over all $P(T) \in \mathcal{L}_{\lambda_{\mathbf{b}_*}}$. □

Claim 4.12 and Claim 4.13 together imply that with probability at least $1 - \left(\frac{1}{\varepsilon^2 q} + \frac{sL}{\tilde{s}}\right)$ over the random choice of $\mathbf{a}$ and $\mathbf{x} \in \mathbb{F}_q^m$, neither $B_{\lambda_{\mathbf{b}_*}}$ nor $C_{\lambda_{\mathbf{b}_*},\mathbf{a}}$ occurs, and hence $P_{\mathbf{b}_*}(T) = Q \circ \lambda_{\mathbf{b}_*}(T)$.

Thus the result follows.

□

4.3 Main local list recovery algorithm

Together, Lemmas 4.3 and 4.8 inspire a local-list recovery algorithm for multivariate multiplicity codes. The idea is that RecoverCandidates will first obtain a list of possibilities, Z, for $Q^{(<\tilde{s})}(\mathbf{a})$. Then for each possibility $z \in Z$, we will create an oracle machine as in Lemma 4.8 which guesses $Q^{(<\tilde{s})}(\mathbf{a}) = z$. Unfortunately, this will still have some amount of error; that is, there will be some small fraction of $\mathbf{x} \in \mathbb{F}_q^m$ so that the approach above will not be correct on $\mathbf{x}$.

To get around this, we first reduce the fraction of erroneous input lists by replacing each given input list $S(\mathbf{y})$ with the result of applying RecoverCandidates on $\mathbf{y}$ with oracle access to S and parameter $\tilde{s} = s$, to obatin a list of possibilities $\hat{S}(\mathbf{y})$ for $Q^{(<s)}(\mathbf{y})$. By Lemma 4.8, this will result in a reduced amount of error in the decoded codewords. Once the error is sufficiently small, we can wrap the whole thing in the local (unique) correction algorithm. Specifically, we use the following local correction algorithm for multivariate multiplicity codes from [KSY14].

Theorem 4.14 ([KSY14], Theorem 3.6)**.** *Let q be a prime power, let s, d, m be nonnegative integers, and let $\delta := 1 - \frac{d}{sq}$. Suppose that $q \geq \max\{10m, \frac{d+6s}{s}, 12(s+1)\}$. Then the multivariate multiplicity code $\mathsf{MULT}_{q,s}^{(m)}(d)$ is locally correctable from $\frac{\delta}{10}$-fraction of errors with $(O(s)^m \cdot q)$ queries.*

Assuming the above local-correction algorithm LocalCorrect for multivariate multiplicity codes in hand, we define our final local-list recovery algorithm as follows.

Algorithm LocalListRecoverMULT.

- Oracle access to $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$.

1. Pick $\mathbf{a}, \mathbf{b}, \mathbf{b}' \in \mathbb{F}_q^m$ uniformly at random.
2. Set $\tilde{s} = \frac{100 \cdot L' \cdot s}{\alpha}$.
3. Let Z be the output of $\mathsf{RecoverCandidates}^S(\mathbf{a}, \tilde{s})$, where the random choice of the procedure RecoverCandidates is fixed to $\mathbf{b}$.
4. For $\mathrm{y} \in \mathbb{F}_q^m$, let $\hat{S}(\mathbf{y}) = \mathsf{RecoverCandidates}^S(\mathbf{y}, s)$, where the random choice of the procedure RecoverCandidates is fixed to $\mathbf{b}'$.
5. For $z \in Z$, define $\mathcal{A}_z$ by:
   - **INPUT:** $\mathbf{x} \in \mathbb{F}_q^m$

(a) Let M denote the oracle machine $M^{\hat{S}}[\mathbf{a}, z]$.

(b) Return $\mathsf{LocalCorrect}^{M}(\mathbf{x})$

6. Return $\mathcal{L} = \{\mathcal{A}_z : z \in Z\}$.

The following lemma shows that this algorithm works, assuming a (global) list recovery algorithm for univariate multiplicity codes. This lemma implies in turn our main Lemma 4.2.

Lemma 4.15. *Let q be a prime power, let s, d, m be nonnegative integers, and let $\delta := 1 - \frac{d}{sq}$. Let $\alpha \in (0, \delta)$ and $\ell \in \mathbb{N}$, and suppose that the univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(d)$ is (α, ℓ, L)-(globally) list recoverable and $(\alpha, 100L, L')$-(globally) list recoverable. Let $\varepsilon > 0$ be a parameter, and suppose that $q \geq 100^2 \frac{s \cdot L \cdot L'}{\alpha^3 \cdot \varepsilon^2}$.*

Let $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$, and suppose that $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ is a polynomial of degree at most d such that:

$$\Pr_{\mathbf{x} \in \mathbb{F}_q^m} \left[Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x}) \right] > 1 - \alpha + \varepsilon.$$

Then with probability at least $\frac{2}{3}$ over the choice of uniform random $\mathrm{a}, \mathrm{b}, \mathrm{b}' \in \mathbb{F}_q^m$, the following holds. There exists an oracle machine $\mathcal{A}_z \in \mathcal{L}$ so that for all $\mathrm{x} \in \mathbb{F}_q^m$,

$$\Pr \left[\mathcal{A}_z(\mathbf{x}) = Q^{(<s)}(\mathbf{x}) \right] \geq \frac{2}{3},$$

where the probability is over the internal randomness of $\mathcal{A}_z$.

Moreover, the output list $\mathcal{L}$ has size $|\mathcal{L}| = O(L)$; and $\mathsf{LocalListRecoverMULT}$ makes $q \cdot \left(\frac{s \cdot L \cdot L'}{\alpha} \right)^{O(m)}$ queries to S, and each $\mathcal{A}_z$ makes $q^3 \cdot (s \cdot L)^{O(m)}$ queries to S.

The rest of this section is devoted to the proof of the above lemma. We first establish the correctness of the algorithm $\mathsf{LocalListRecoverMULT}$ given above.

Let E_1 be the event that $Q^{(<\tilde{s})}(\mathbf{a}) \in Z$. By Lemma 4.3, we have that,

Claim 4.16.

$$\Pr_{\mathbf{a}, \mathbf{b}}[E_1] \geq 1 - \frac{1}{\varepsilon^2 q}.$$

Next, let E_2 be the event that $Q^{(<s)}(\mathbf{y}) \in \hat{S}(\mathbf{y})$ for at least $\left(1 - \frac{\alpha}{100}\right)$-fraction of $\mathbf{y} \in \mathbb{F}_q^m$.

Claim 4.17.

$$\Pr_{\mathbf{b}'}[E_2] \geq 1 - \frac{100}{\alpha \varepsilon^2 q}.$$

Proof. By Lemma 4.3,

$$\mathbb{E}_{\mathbf{y}, \mathbf{b}'} \left[\mathbf{1} \left\{ Q^{(<s)}(\mathbf{y}) \in \hat{S}(\mathbf{y}) \right\} \right] \geq 1 - \frac{1}{\varepsilon^2 q},$$

and so by Markov's inequality,

$$\Pr_{\mathbf{b}'}\left[\Pr_{\mathbf{y}}\left[Q^{(<s)}(\mathbf{y}) \in \hat{S}(\mathbf{y})\right] \geq 1 - \frac{\alpha}{100}\right] \geq 1 - \frac{100}{\alpha\varepsilon^2 q}.$$

□

Finally, let E_3 be the event that $M^{\hat{S}}\left[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})\right](\mathbf{x}) = Q^{(<s)}(\mathbf{x})$ for at least $\left(1 - \frac{\alpha}{10}\right)$-fraction of inputs $\mathbf{x} \in \mathbb{F}_q^m$.

Claim 4.18.

$$\Pr_{\mathbf{a}}\left[E_3 | E_2\right] \geq 1 - \frac{10\gamma}{\alpha}$$

for

$$\gamma := \frac{\alpha}{100} + \frac{100Ls}{q} + \frac{1}{0.99^2\alpha^2 q} + \frac{sL'}{\tilde{s}}. \tag{5}$$

Proof. By Lemma 4.8, assuming event E_2 holds,

$$\mathbb{E}_{\mathbf{x},\mathbf{a}}\left[\mathbf{1}\left\{M^{\hat{S}}\left[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})\right](\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right\}\right] \geq 1 - \gamma,$$

recalling that by Lemma 4.3 $\hat{S}(\mathbf{y})$ has size at most $100L$ for all $\mathrm{y} \in \mathbb{F}_q^m$, and our assumption that the univariate multiplicity code is $(\alpha, 100L, L')$-list recoverable. By Markov's inequality, this implies in turn that

$$\Pr_{\mathbf{a}}\left[\Pr_{\mathbf{x}}\left[M^{\hat{S}}\left[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})\right](\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \geq 1 - \frac{\alpha}{10}\right] \geq 1 - \frac{10\gamma}{\alpha}.$$

□

Thus, by the union bound, the above Claims 4.16, 4.17, and 4.18 imply that with probability at least $1 - \frac{1}{\varepsilon^2 q} - \frac{100}{\alpha\varepsilon^2 q} - \frac{10\gamma}{\alpha}$ over the choice of $\mathbf{a}, \mathbf{b}, \mathbf{b}'$, all of the events E_1, E_2, E_3 occur, where γ is as given in (5). Recalling our choice of $\tilde{s} := \frac{100 \cdot L' \cdot s}{\alpha}$ and $q \geq 100^2 \frac{s \cdot L \cdot L'}{\alpha^3 \cdot \varepsilon^2}$, we can ensure that this probability is at least $2/3$.

So with probability at least $2/3$ over the choice of $\mathrm{a}, \mathrm{b}, \mathrm{b}'$, there exists $z = Q^{(<\tilde{s})}(\mathbf{a}) \in Z$ so that $M = M^{\hat{S}}[\mathbf{a}, z]$ recovers correctly all but at most $\frac{\alpha}{10}$-fraction of the inputs $\mathbf{x} \in \mathbb{F}_q^m$. Recalling our assumption that $\alpha \in (0, \delta)$, we may now apply the local correction algorithm from Theorem 4.14 to the oracle machine $M = M^{\hat{S}}[\mathbf{a}, z]$, and conclude that for all $\mathbf{x} \in \mathbb{F}_q^m$, with probability at least $2/3$, $\mathcal{A}_z(\mathbf{x}) = Q^{(<s)}(\mathbf{x})$, as desired.

Now that we have established that the algorithm is correct, we quickly work out the list size and query complexity. The list size is clearly $O(L)$, because this is the list size returned by RecoverCandidates. For the query complexity, the algorithm LocalListRecoverMULT has the same query complexity as RecoverCandidates, while each $\mathcal{A}_z$ has query complexity which is the product of the query complexities of the oracle machines $M^S[\mathbf{a}, z]$ (which is q), RecoverCandidates with parameter $\tilde{s} = s$, and LocalCorrect, and together these give the reported values.

5 Capacity-achieving locally list decodable codes

In this section we use the results from the previous section to construct capacity-achieving locally list decodable codes with low (sub-polynomial) query complexity. For this, we first show that *high-rate* multivariate multiplicity codes are *locally list recoverable* with small query complexity, and then use the AEL distance amplification [AEL95, AL96] to transform these codes into capacity-achieving locally list decodable codes, while roughy preserving the query complexity. These two steps are described in Sections 5.1 and 5.2 below.

5.1 High-rate locally list recoverable codes

In this section we use our results from the previous section to show that high-rate multivariate multiplicity codes are locally list recoverable with low query complexity.

Theorem 5.1. *For any $\gamma > 0$ and $\ell \in \mathbb{N}$ there exists an infinite family $\{C_N\}_N$ of codes that satisfy the following.*

1. *C_N is a code of block length N and rate at least $1 - \gamma$.*
2. *C_N is (t, α, ℓ, L)-locally list recoverable for*

$$t = \exp_{\gamma,\ell}\left((\log N)^{5/6} \cdot (\log\log N)^{1/3}\right), \qquad \alpha = \Omega_{\gamma,\ell}\left(\frac{(\log\log N)^{1/3}}{(\log N)^{1/6}}\right),$$

$$\text{and} \qquad L = \exp_{\gamma,\ell}\left((\log N)^{2/3} \cdot (\log\log N)^{2/3}\right).$$

3. *The alphabet size of C_N is $\exp_{\gamma,\ell}\exp\left((\log N)^{1/6} \cdot (\log\log N)^{2/3}\right)$.*

While in principle we could have used Theorem 4.1 as our starting point for the proof of the above theorem, this will lead to pretty high query complexity since the field size is exponential in L, which is in turn exponential in s.[10] The reason for this exponential dependency of the field size on L is the fact that in our main local list recovery algorithm LocalListRecoverMULT (in Section 4.3) we have first run the algorithm RecoverCandidates on each point $\mathbf{y}$ to reduce the amount of errors in the input lists. This step was necessary in order to be able to locally list recover up to the minimum distance.

Here we observe that if we are not interested in locally list recovering up to the minimum distance, but just up to some small fraction of the minimum distance then we can eliminate this step, which will result in turn in polynomial dependency of the field size on L. This in turn will suffice for applying the AEL transformation, and obtaining capacity-achieving locally list decodable codes.

[10] To achieve high-rate we shall need to set s to be larger than the number of variables m, and so the field size would be doubly-exponential in m. But this means that $m \leq O(\log\log N)$ where $N = q^m$ is the block length, and so the query complexity is at least $q = N^{1/m} \geq N^{\Omega(1/\log\log N)}$.

Theorem 5.2. *There exist absolute constants $c_0, c_1 > 0$ so that the following holds. Let q be a prime, let s, d, m be nonnegative integers, let $\delta := 1 - \frac{d}{sq}$, and assume that $\delta < \frac{1}{2}$. Let $\ell, L \in \mathbb{N}$ be such that $s \geq \frac{64\ell}{\delta^2}$, $L \geq 2^{c_0(s \log s)^2}$, and $q \geq (m \cdot L)^{c_1}$. Then the multivariate multiplicity code $\mathsf{MULT}_{q,s}^{(m)}(d)$ is $\left(t, \frac{\delta}{100}, \ell, L\right)$-locally list recoverable for $t = q^{O(1)} \cdot L^{O(m)}$.*

We sketch the proof of the above theorem in Section 5.1.1, but first we show how this theorem implies Theorem 5.1.

Proof of Theorem 5.1. We shall let $C_N := \mathsf{MULT}_{q,s}^{(m)}(d)$, as per the following choice of parameters. Let m be a parameter to be determined later on. Let

$$s := \frac{300\ell}{\gamma^2} \cdot m^2 = \Theta_{\gamma,\ell}(m^2). \tag{6}$$

Let q be a prime such that

$$L := 2^{c_0(s \log s)^2} = \exp_{\gamma,\ell}\left(m^4 \log^2 m\right), \tag{7}$$

where c_0 is the constant guaranteed by Theorem 5.2.

Let

$$q := L^{c_1 \cdot m} = \exp_{\gamma,\ell}(m^5 \log^2 m), \tag{8}$$

where c_1 is the constant guaranteed by Theorem 5.2, and note that $q \geq (m \cdot L)^{c_1}$. Jumping ahead, the reason for this choice of q is that this is the largest we may take q so that the query complexity expression $q^{O(1)} \cdot L^{O(m)}$ from Theorem 5.2 does not substantially grow. Finally, let $d := \left(1 - \frac{\gamma}{2m}\right) sq$, and note that this choice implies that

$$\delta := \frac{\gamma}{2m}, \tag{9}$$

and that under this choice we have that $\delta < \frac{1}{2}$ and $s \geq \frac{64\ell}{\delta^2}$.

Finally, to guarantee block length N we must have that $N = q^m = \exp_{\gamma,\ell}(m^6 \log^2 m)$, which implies in turn that

$$m = \Theta_{\gamma,\ell}\left(\frac{(\log N)^{1/6}}{(\log \log N)^{1/3}}\right). \tag{10}$$

First observe that as per Claim 2.5, the rate of C is at least

$$\begin{aligned} R &\geq \left(1 - \frac{m^2}{s}\right)(1-\delta)^m \\ &= \left(1 - \frac{\gamma^2}{300\ell}\right)\left(1 - \frac{\gamma}{2m}\right)^m \\ &\geq \left(1 - \frac{\gamma^2}{300\ell}\right)\left(1 - \frac{\gamma}{2}\right) \\ &\geq 1 - \gamma, \end{aligned}$$

where the first equality follows by choices of s and δ in (6) and (9), respectively, and the last two inequalities hold for sufficiently small γ.

By Theorem 5.2, the query complexity for locally list recovering C_N is

$$t = q^{O(1)} \cdot L^{O(m)} = \exp_{\gamma,\ell}(m^5 \log^2 m) = \exp_{\gamma,\ell}\left((\log N)^{5/6} \cdot (\log\log N)^{1/3}\right),$$

the decoding radius is

$$\alpha = \frac{\delta}{100} = \frac{\gamma}{2m} = \Omega_{\gamma,\ell}\left(\frac{(\log\log N)^{1/3}}{(\log N)^{1/6}}\right),$$

and the output list size is

$$L = \exp_{\gamma,\ell}(m^4 \log^2 m) = \exp_{\gamma,\ell}\left((\log N)^{2/3} \cdot (\log\log N)^{2/3}\right),$$

where the equalities follow by our choice of q, L, δ, and m in (8), (7), (9), and (10).

Finally, the alphabet size is

$$q^{\binom{s+m-1}{m}} = \exp_{\gamma,\ell}\left(m^{O(m)}\right) = \exp_{\gamma,\ell}\exp\left((\log N)^{1/6} \cdot (\log\log N)^{2/3}\right),$$

where the equalities follow by our choice of q, s, and m in (8), (6), and (10), respectively. □

5.1.1 Proof sketch of Theorem 5.2

As before, Theorem 5.2 follows from the following lemma which relates the parameters of the global list recovery algorithm for univariate multiplicity codes to that of the local list recovery algorithm for the corresponding multivariate multiplicity codes. Theorem 5.2 follows by instantiating this lemma with the global list recovery algorithm for univariate multiplicity codes from Theorem 3.8.

Lemma 5.3. *Let q be a prime power, let s, d, m be nonnegative integers, and let $\delta := 1 - \frac{d}{sq}$. Let $\ell \in \mathbb{N}$, and suppose that the univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(d)$ is $\left(\frac{\delta}{2}, \ell, L\right)$-(globally) list recoverable. Suppose that $q \geq \max\{10m, 100^2 \frac{s \cdot \ell \cdot L^2}{\delta^3}\}$. Then the multivariate multiplicity code $\mathsf{MULT}^{(m)}_{q,s}(d)$ is $\left(t, \frac{\delta}{100}, \ell, O(L)\right)$-locally list recoverable for $t = q^{O(1)} \cdot \left(\frac{s \cdot L}{\delta}\right)^{O(m)}$.*

The main advantage of the above lemma over Lemma 4.2 is that we do not need to assume that the univariate code is $(\alpha, 100L, L')$-list recoverable, and the dependency on L' is eliminated. However, we only get that the multivariate code is locally list recoverable from at most $\frac{\delta}{100}$-fraction of errors. Since the proof of the above lemma is very similar to that of Lemma 4.2, we just sketch the differences below.

Proof of Lemma 5.3 (sketch). We use the same algorithm as before, except that in algorithm $\mathsf{LocalListRecoverMULT}$ (in Section 4.3) we simply replace the input lists $\hat{S}$ with the original input lists S (and we do not need to sample $\mathbf{b}'$). We also choose $\tilde{s} := \frac{100 \cdot L \cdot s}{\delta}$.

As before, we let E_1 be the event that $Q^{(<\tilde{s})}(\mathbf{a}) \in Z$, and by Lemma 4.3, we have that

$$\Pr_{\mathbf{a},\mathbf{b}}[E_1] \geq 1 - \frac{16}{\delta^2 q}.$$

Next, note that by our assumption, it holds that $Q^{(<s)}(\mathbf{y}) \in S(\mathbf{y})$ for at least $\left(1 - \frac{\delta}{100}\right)$-fraction of $\mathbf{y} \in \mathbb{F}_q^m$. Finally, let E_3 be the event that $M^S\left[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})\right](\mathbf{x}) = Q^{(<s)}(\mathbf{x})$ for at least $\left(1 - \frac{\delta}{10}\right)$-fraction of inputs $\mathbf{x} \in \mathbb{F}_q^m$. Then as before, by Lemma 4.8 and Markov's inequality, we have that

$$\Pr_{\mathbf{a}}[E_3] \geq 1 - \frac{10\gamma}{\delta}$$

for

$$\gamma := \frac{\delta}{100} + \frac{\ell s}{q} + \frac{16}{\delta^2 q} + \frac{sL}{\tilde{s}}.$$

Thus, by the union bound, we get that with probability at least $2/3$ over the choice of $\mathrm{a}, \mathrm{b} \in \mathbb{F}_q^m$, there exists $z = Q^{(<\tilde{s})}(\mathbf{a}) \in Z$ so that $M = M^S[\mathbf{a}, z]$ recovers correctly all but at most $\frac{\delta}{10}$-fraction of the inputs $\mathbf{x} \in \mathbb{F}_q^m$. As before, we may then apply the local correction algorithm from Theorem 4.14 to the oracle machine, and conclude that for all $\mathbf{x} \in \mathbb{F}_q^m$, with probability at least $2/3$, $\mathcal{A}_z(\mathbf{x}) = Q^{(<s)}(\mathbf{x})$, as desired. Finally, it can be verified that query complexity and output list size are as stated. □

5.2 Capacity-achieving locally list decodable codes

In this section we apply the Alon-Edmonds-Luby (AEL) distance amplification method [AEL95, AL96] on the codes given by Theorem 5.1 to obtain capacity-achieving locally list decodable (in fact, recoverable) codes with small (sub-polynomial) query complexity.

Theorem 5.4. *For any $R \in (0,1)$, $\gamma > 0$ and $\ell \in \mathbb{N}$ there exists an infinite family $\{C_N\}_N$ of codes that satisfy the following.*

1. *C_N is a code of block length N and rate at least R.*
2. *C_N is $(t, 1 - R - \gamma, \ell, L)$-locally list recoverable for*

$$t = \exp_{R,\gamma,\ell}\left((\log N)^{5/6} \cdot (\log\log N)^{1/3}\right) \qquad \text{and} \qquad L = \exp_{R,\gamma,\ell}\left((\log N)^{2/3} \cdot (\log\log N)^{2/3}\right).$$

3. *The alphabet size of C_N is $O_{R,\gamma,\ell}(1)$.*

To prove the above theorem one can use the following version of the AEL transformation for local list recovery from [GKO+18] which roughly says the following. Given an "outer" code C of rate approaching 1 that is locally list recoverable from a tiny fraction of errors, and a small "inner" code C' that is a capacity-achieving (globally) list recoverable code, they can be combined to get a new code C_{AEL} that on the one hand, inherits the tradeoff between rate and error correction that C' enjoys, and on the other hand, inherits the locality of C.

Lemma 5.5 (Distance amplification for local list recovery, [GKO+18], Lemma 5.4.)**.** *There exists an absolute constant b_0 so that the following holds for any $\alpha, \gamma > 0$ and $s \geq (\alpha \cdot \gamma)^{-b_0}$. Suppose that $C \subseteq (\Sigma^{R \cdot s})^n$ is an outer code of rate $1 - \gamma$ that is (t, α, ℓ, L)-locally list recoverable, and $C' \subseteq \Sigma^s$ is an inner code of rate R that is $(1 - R - \gamma, \ell', \ell)$-globally list recoverable. Then there exists a code $C_{AEL} \subseteq (\Sigma^s)^n$ of rate $R - \gamma$ that is $(t \cdot \mathsf{poly}(s), 1 - R - 2\gamma, \ell', L)$-locally list recoverable.*

Since the AEL transforation has been applied in a similar way in a sequence of recent papers [KMRS17, GKO+18, HRW17, KRR+19], we only sketch the proof of Theorem 5.4. The reader is referred to the previous papers for full details. First note that one can apply the above distance amplification Lemma 5.5 with the outer code being the high-rate locally list recoverable codes given by Theorem 5.1, and the inner code being a capacity-achieving (globally) list recoverable code with constant output list size (depending only on R, γ, ℓ; e.g., a random code). This will give the code family advertised in Theorem 5.4, except for the alphabet size that would be very large $\exp_{R,\gamma,\ell} \exp\left((\log N)^{1/6} \cdot (\log\log N)^{2/3}\right)$.

To reduce the alphabet size to a constant one can then concatenate the resulting code with a high-rate globally list recoverable code with constant decoding radius, alphabet size, and output list size (once more depending only on R, γ, ℓ; e.g, a random code). This will give a high-rate locally list recoverable code with essentially the same parameters as in Theorem 5.1 but with constant decoding radius and alphabet size. Finally, one can apply once more the distance amplification Lemma 5.5 to obtain the decoding radius advertised by Theorem 5.4.

Acknowledgements

We would like to thank Atri Rudra and Venkatesan Guruswami for helpful discussions.

References

[AEL95] Noga Alon, Jeff Edmonds, and Michael Luby. Linear time erasure codes with nearly optimal recovery. In *proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 512–519. IEEE Computer Society, 1995.

[AL96] Noga Alon and Michael Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory*, 42(6):1732–1736, 1996.

[AS03] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–31. ACM Press, 1991.

[BK09] Kristian Brander and Swastik Koparty. List-decoding reed-muller over large fields upto the johnson radius. *Manuscript*, 2009.

[BL18] Abhishek Bhowmick and Shachar Lovett. The list decoding radius for reed-muller codes over small fields. *IEEE Transactions on Information Theory*, 64(6):4382–4391, 2018.

[DKSS13] Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the method of multiplicities, with applications to kakeya sets and mergers. *SIAM Journal on Computing*, 42(6):2305–2328, 2013.

[DL12] Zeev Dvir and Shachar Lovett. Subspace evasive sets. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC)*, pages 351–358. ACM Press, 2012.

[GI04] Venkatesan Guruswami and Piotr Indyk. Linear-time list decoding in error-free settings. In *proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *Lecture Notes in Computer Science*, pages 695–707. Springer, 2004.

[GK16a] Alan Guo and Swastik Kopparty. List-decoding algorithms for lifted codes. *IEEE Transactions on Information Theory*, 62(5):2719–2725, 2016.

[GK16b] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016.

[GKO+18] Sivakanth Gopi, Swastik Kopparty, Rafael Oliveira, Noga Ron-Zewi, and Shubhangi Saraf. Locally testable and locally correctable codes approaching the gilbert-varshamov bound. *IEEE Transactions on Information Theory*, 64(8):5813–5831, 2018.

[GKZ08] Parikshit Gopalan, Adam R. Klivans, and David Zuckerman. List-decoding reed-muller codes over small fields. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 265–274. ACM Press, 2008.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32. ACM Press, 1989.

[Gop13] Parikshit Gopalan. A fourier-analytic approach to reed-muller decoding. *IEEE Transactions on Information Theory*, 59(11):7747–7760, 2013.

[GR08] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[Gur04] Venkatesan Guruswami. *List decoding of error-correcting codes: winning thesis of the 2002 ACM doctoral dissertation competition*, volume 3282. Springer Science & Business Media, 2004.

[GW12] Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed-solomon codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:73, 2012.

[GW13] Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed-solomon codes. *IEEE Transactions on Information Theory*, 59(6):3257–3268, 2013.

[HRW17] Brett Hemenway, Noga Ron-Zewi, and Mary Wootters. Local list recovery of high-rate tensor codes and applications. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 2017.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[KMRS17] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *Journal of ACM*, 64(2):11:1–11:42, 2017.

[Kop15] Swastik Kopparty. List-decoding multiplicity codes. *Theory of Computing*, 11(5):149–182, 2015.

[KRR+19] Swastik Kopparty, Nicolas Resch, Noga Ron-Zewi, Shubhangi Saraf, and Shashwat Silas. On list recovery of high-rate tensor codes. In *proceedings of the 23rd International Conference on Randomization and Computation (RANDOM)*, pages 68:1–68:22. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[KRSW18] Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved decoding of folded reed-solomon and multiplicity codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:91, 2018.

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *Journal of ACM*, 61(5):28, 2014.

[KT00] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 80–86. ACM Press, 2000.

[Lip90] Richard J. Lipton. Efficient checking of computations. In *Proceedings of the 7th Annual ACM Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 207–215. Springer, 1990.

[Nie01] R. R. Nielsen. *List decoding of linear block codes*. PhD thesis, Technical University of Denmark, 2001.

[PV05] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the Guruswami–Sudan radius in polynomial time. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 285–294. IEEE Computer Society, 2005.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[RT97] M. Yu. Rosenbloom and M. A. Tsfasman. Codes for the m-metric. *Problemy Peredachi Informatsii*, 33(1):55–63, 1997.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[SY11] Shubhangi Saraf and Sergey Yekhanin. Noisy interpolation of sparse polynomials, and applications. In *Proceedings of the IEEE 26th Annual Conference on Computational Complexity (CCC)*, pages 86–92. IEEE Computer Society, 2011.

A Constant-dimensional output list for univariate multiplicity codes

In this section we prove the following lemma which implies Theorem 3.7.

Lemma A.1. *Let q be a prime power, and let s, d, n be nonnegative integers such that $n \leq q$. Let r be a nonnegative integer such that $r \leq \min\{s, \mathsf{char}(\mathbb{F}_q)\}$. Then the univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(n,d)$ is (α, ℓ, L)-list recoverable for*

$$\alpha < 1 - \frac{\ell}{r+1} - \frac{r}{r+1} \cdot \frac{s}{s-r+1} \cdot \frac{d}{sq},$$

where the output list is contained in an $\mathbb{F}_q$-affine subspace $v_0 + V$ of dimension at most $r \cdot \left(1 + \frac{d}{\mathsf{char}(\mathbb{F}_q)}\right)$. Moreover, there is a (deterministic) algorithm that outputs a basis for V in time $\mathsf{poly}(\log q, s, d, n)$.

Theorem 3.7 follows by setting $s \geq \frac{16\ell}{\varepsilon^2}$ and $r = \frac{4\ell}{\varepsilon}$, and noting that in this setting of parameters we have that $\alpha \geq \delta - \varepsilon$.

The proof of Lemma A.1 above adapts Theorem 17 of [GW12] to our setting. We refer the reader to [GW12] for more context and intuition.

We begin by giving the algorithm.

Algorithm FindSubspace.

- **INPUT:** Access to input lists $S \subseteq \binom{\mathbb{F}_q^s}{\ell}^n$
- **OUTPUT:** (A basis for) an $\mathbb{F}_q$-affine subspace $v_0 + V$ containing all codewords $c \in \mathsf{MULT}^{(1)}_{q,s}(n,d)$ with $\mathrm{dist}(c, S) \leq \alpha$.

1. Set $D = (s - r + 1)(1 - \alpha)q - 1$.
2. By solving a linear system of equations over $\mathbb{F}_q$, find nonzero polynomials

$$A(X), B_0(X), \ldots, B_{r-1}(X) \in \mathbb{F}_q[X]$$

such that:

(a) $\deg(A) \leq D$, and for all $i = 0, \ldots, r-1$, $\deg(B_i) \leq D - d$.

(b) For each λ with $0 \leq \lambda \leq s - r$, for each evaluation point $a_i \in \mathbb{F}_q$, and for each $\beta = (\beta_0, \ldots, \beta_{s-1}) \in S_i$:

$$A^{(\lambda)}(a_i) + \sum_{i=0}^{r-1} \sum_{j=0}^{\lambda} \binom{i+j}{i} \beta_{i+j} B_i^{(\lambda-j)}(a_i) = 0.$$

3. Let $v_0 + V$ be the affine space containing the encoding of all polynomials $f(X)$ satisfying

$$A(X) + \sum_{i=0}^{r-1} f^{(i)}(X) B_i(X) = 0.$$

We need to show:

1. The linear system has a nonzero solution,
2. Any codeword $c \in \mathsf{MULT}^{(1)}_{q,s}(n, d)$ with $\mathrm{dist}(c, S) \leq \alpha$ is contained in $v_0 + V$.

We show these two items below.

Item (1): To see that the linear system has a nonzero solution, we show that the homogeneous system of linear equations in Step 2 of the algorithm has more variables than constraints. The total number of free coefficients in $A(X), B_0(X), \ldots, B_{r-1}(X)$ equals:

$$\begin{aligned}
(D+1) + r(D-d+1) &= (D+1)(r+1) - d \cdot r \\
&= (s-r+1)(1-\alpha)q(r+1) - d \cdot r \\
&> (s-r+1)\left(\frac{\ell}{r+1} + \frac{r}{r+1} \cdot \frac{s}{s-r+1} \cdot \frac{d}{sq}\right) q(r+1) - d \cdot r \\
&= (s-r+1)\ell q + d \cdot r - d \cdot r \\
&= (s-r+1)\ell q.
\end{aligned}$$

The total number of constraints equals:

$$q \cdot (s - r + 1) \cdot \ell.$$

Thus the number of free coefficients is larger than the number of constraints, and this proves that the algorithm can find a nonzero solution in Step 2.

Item (2): Let c be a codeword with $\mathrm{dist}(c, S) \leq \alpha$ that is the encoding of a polynomial $g(X)$. We will show that $c \in v_0 + V$. Define $Q(X) = A(X) + \sum_{i=0}^{r-1} g^{(i)}(X) B_i(X)$. Observe that $\deg(Q) \leq D$.

Now take any evaluation point $a_i \in \mathbb{F}_q$ and $\beta = (\beta_0, \ldots, \beta_{s-1}) \in S_i$ for which

$$g^{(<s)}(a_i) = \beta \tag{11}$$

Let λ be an integer with $0 \leq \lambda \leq s - r$. Then by the chain rule for Hasse derivatives:

$$\begin{aligned}
Q^{(\lambda)}(a_i) &= A^{(\lambda)}(a_i) + \sum_{i=0}^{r-1} \left(g^{(i)} \cdot B_i\right)^{(\lambda)}(a_i) \\
&= A^{(\lambda)}(a_i) + \sum_{i=0}^{r-1} \sum_{j=0}^{\lambda} \left(g^{(i)}\right)^{(j)}(a_i) B_i^{(\lambda-j)}(a_i) \\
&= A^{(\lambda)}(a_i) + \sum_{i=0}^{r-1} \sum_{j=0}^{\lambda} \binom{i+j}{i} g^{(i+j)}(a_i) B_i^{(\lambda-j)}(a_i) \\
&= A^{(\lambda)}(a_i) + \sum_{i=0}^{r-1} \sum_{j=0}^{\lambda} \binom{i+j}{i} \beta_{i+j} B_i^{(\lambda-j)}(a_i) \\
&= 0
\end{aligned}$$

Since this holds for every λ with $0 \leq \lambda \leq s - r$, we get that:

$$\mathsf{mult}(Q, a_i) \geq s - r + 1.$$

By assumption on g, there are at least $(1-\alpha)q$ evaluation points $a_i \in \mathbb{F}_q$ such that there exists some $\beta \in S_i$ for which Equation (11) holds. Thus there are at least $(1-\alpha)q$ points where Q vanishes with multiplicity at least $s - r + 1$. Since $\deg(Q) \leq D < (s - r + 1)(1 - \alpha)q$, we conclude that $Q(X) = 0$.

By definition of $Q(X)$ and $v_0 + V$, this implies that $c \in v_0 + V$, as desired.

Finally, we turn to bound the dimension of V.

Dimension of affine subspace: Let $f = \sum_{i=0}^{d} f_i x^i$ be a polynomial satisfying $Q(x) := A(X) + \sum_{i=0}^{r-1} f^{(i)}(X) B_i(X) = 0$. Then for each $i = r-1, \ldots, d$, the coefficient of X^{i-r+1} in $Q(x)$ is $\binom{i}{r-1} \cdot f_i + \ell_i$, where ℓ_i is an affine linear combination of $f_1, \ldots, f_{i-1}$. Thus, whenever $\binom{i}{r-1} = \frac{i \cdot (i-1) \cdots (i-r+1)}{(r-1)!} \neq 0$ we get that the coefficient f_i is determined by prior coefficients. By assumption that $r \leq \mathsf{char}(\mathbb{F}_q)$ we get that $(r-1)! \neq 0$. Moreover, $i \cdot (i-1) \cdots (i-r+1)$ can be non-zero for at most $r \cdot \frac{d+1}{\mathsf{char}(\mathbb{F}_q)}$ values in $\mathbb{F}_q$. We conclude that the number of undetermined coefficients is at most $r - 1 + r \cdot \frac{d+1}{\mathsf{char}(\mathbb{F}_q)} \leq r \cdot \left(1 + \frac{d}{\mathsf{char}(\mathbb{F}_q)}\right)$, and this also gives a bound on the dimension of V.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il