58th Annual IEEE Symposium on Foundations of Computer Science

Local List Recovery of High-rate Tensor Codes & Applications

Brett Hemenway Noga Ron-Zewi Mary Wootters

***Abstract*—In this work, we give the first construction of *high-rate* locally list-recoverable codes. List-recovery has been an extremely useful building block in coding theory, and our motivation is to use these codes as such a building block. In particular, our construction gives the first *capacity-achieving* locally list-decodable codes (over constant-sized alphabet); the first *capacity achieving* globally list-decodable codes with nearly linear time list decoding algorithm (once more, over constant-sized alphabet); and a randomized construction of binary codes on the Gilbert-Varshamov bound that can be uniquely decoded in near-linear-time, with higher rate than was previously known.**

Our techniques are actually quite simple, and are inspired by an approach of Gopalan, Guruswami, and Raghavendra (Siam Journal on Computing, 2011) for list-decoding tensor codes. We show that tensor powers of (globally) list-recoverable codes are 'approximately' locally list-recoverable, and that the 'approximately' modifier may be removed by pre-encoding the message with a suitable locally decodable code. Instantiating this with known constructions of high-rate globally list-recoverable codes and high-rate locally decodable codes finishes the construction.

I. Introduction

List-recovery refers to the problem of decoding error correcting codes from "soft" information. More precisely, given a *code* $C : \Sigma^k \to \Sigma^n$, which maps length-$k$ *messages* to length-n *codewords,* an (α, ℓ, L)-list-recovery algorithm for C is provided with a sequence of lists $S_1, \ldots, S_n \subset \Sigma$ of size at most ℓ each, and is tasked with efficiently returning all messages $x \in \Sigma^k$ so that $C(x)_i \notin S_i$ for at most α fraction of the coordinates i; the guarantee is that there are no more than L such messages. The goal is to design codes C which simultaneously admit such algorithms, and which also have other desirable properties, like high *rate* (that is, the ratio k/n, which captures how much information can be sent using the code) or small *alphabet size* $|\Sigma|$. List-recovery is a generalization of *list-decoding,* which is the situation when the lists S_i have size one: we refer to $(\alpha, 1, L)$-list-recovery as (α, L)-list-decoding.

List recoverable codes were first studied in the context of list-decoding and soft-decoding. The celebrated Guruswami-Sudan list-decoding algorithm [GS99] is in fact a list-recovery algorithm, as are several more recent list-decoding algorithms [GR08], [GW11], [Kop15], [GX13]. Initially, list recoverable codes were used as stepping stones towards constructions of list decodable and uniquely decodable codes [GI01], [GI02], [GI03], [GI04]. Since then, list recoverable codes have found additional applications in the areas of compressed sensing, combinatorial group testing, and hashing [INR10], [NPR12], [GNP+13], [HIOS15].

Locality is another frequent desideratum in coding theory. Loosely, an algorithm is "local" if information about a single coordinate x_i of a message x of C can be determined locally from only a few coordinates of a corrupted version of $C(x)$. Locality, and in particular local list-decoding, has been implicit in theoretical computer science for decades: for example, local list-decoding algorithms are at the heart of algorithms in cryptography [GL89], learning theory [KM93], and hardness amplification and derandomization [STV01].

A local list-recovery algorithm returns a list $A_1, \ldots, A_L$ of randomized local algorithms, and each of these algorithms takes an index $i \in [k]$ as input, and has oracle access to the lists $S_1, \ldots, S_n$. The algorithm then makes at most Q queries to this oracle (that is, it sees at most Q different lists S_i), and must return a guess for x_i, where x is a message whose encoding $C(x)$ agrees with many of the lists. The guarantee is that for all such x—that is, for all x whose encoding $C(x)$ agrees with many of the lists—there exists (with high probability) some A_j so that for all i, $A_j(i) = x_i$ with probability at least $2/3$. The parameter Q is called the *query complexity* of the local list-recovery algorithm.

One reason to study local list-recoverability is that list-recovery is a very useful building block throughout coding theory. In particular, the problem of constructing **high rate locally list-recoverable codes** (of rate arbitrarily close to 1, and or at least non-decreasing in ℓ) has been on the radar for a while, because such codes would have implications in local list-decoding, global list-decoding, and classical unique decoding.

In this work, we give the first constructions of high-rate locally list-recoverable codes. As promised, these lead to several applications throughout coding theory. Our construction is actually quite simple, we show that the list-decoding algorithm of [GGR11] for tensor codes can be modified to provide local recovery algorithm.

0272-5428/17 $31.00 © 2017 IEEE
DOI 10.1109/FOCS.2017.27

A. Results

We highlight our main results below—we will elaborate more on these results and their context within related literature next in Section II.

High-rate local list-recovery. Our main technical contribution is the first constructions of high-rate locally list-recoverable codes: Theorem V.5 give the formal statements. If we do not require an efficient recovery procedure, we can guarantee high-rate list recovery with query complexity $n^{1/t}$ (for constant t, say $n^{0.001}$), constant alphabet size and *constant* output list size. Theorem V.5 on the other hand gives an explicit and efficient version, at the cost of a slightly super-constant output list size (which depends on $\log^* n$).

For those familiar with the area, it may be somewhat surprising that this was not known before: indeed, as discussed below in Section II, we know of locally list recoverable codes (of low rate), and we also know of high-rate (globally) list-recoverable codes. One might think that our result is lurking implicitly in those earlier works. However, it turns out that it is not so simple: as discussed below, existing techniques for locally or globally list-recoverable codes do not seem to work for this problem. Indeed, some of those prior works [HW15], [KMRS16], [GKO$^+$17] (which involve the current authors) began with the goal of obtaining high-rate locally list-recoverable codes and ended up somewhere else.

This raises the question: why might one seek high-rate locally list-recoverable error correcting codes in the first place? The motivation is deeper than a desire to add adjectives in front of "error correcting codes." As we will see below, via a number of reductions that already exist in the literature, such codes directly lead to improvements for several other fundamental problems in coding theory, including fast or local algorithms for list and unique decoding.

Capacity-achieving locally list-decodable codes. The first such reduction is an application of an expander-based technique of Alon, Edmunds, and Luby [AEL95], which allows us to turn the high-rate locally list-recoverable codes into *capacity achieving* locally list-decodable (or more generally, locally list recoverable) codes. This gives explicit and efficiently list decodable codes, and a trade-off between query complexity, alphabet size, and output list size. Specifically, these codes obtain query complexity $Q = n^{1/t}$ with an output list size and an alphabet size that grow doubly exponentially with t (and output list size depends additionally on $\log^* n$). In particular, if we choose t to be constant, we obtain query complexity $n^{1/t}$, with constant alphabet size and nearly-constant output list size. We may also choose to take t to be very slowly growing, and this yields query complexity $n^{o(1)}$, with output list and alphabet size $n^{o(1)}$ as well. Prior to this work, no construction of capacity achieving locally list decodable codes with query-complexity $o(n)$ was known. As before, if we do not require efficient recovery, we can guarantee capacity achieving locally list decodable codes with query complexity $n^{0.001}$ (say), constant alphabet size and constant output list size. The details of this construction are in the full version.

Near-linear time capacity-achieving list-decodable codes. An efficiently list decodable capacity achieving locally list-decodable code can also be globally list-decoded. Indeed, we just repeat the local decoding algorithm (which can be done in time $n^{O(1/t)}$) a few times, for all n coordinates, and take majority vote at each coordinate. Thus, our previous result implies explicit, capacity-achieving, list-decodable codes (or more generally, list recoverable codes) that can be (globally) list-decoded (or list-recovered) in time $n^{1+O(1/t)}$.

As with the previous point, this result actually allows for a trade-off: we obtain either decoding time $N^{1.001}$ (say) with constant alphabet size and near-constant output list size, or decoding time $n^{1+o(1)}$ at the cost of increasing the alphabet and output list size to $n^{o(1)}$. Previous capacity achieving list-decoding algorithms required at least quadratic time for recovery. The details of this construction are in the full version.

Near-linear time unique decoding up to the Gilbert Varshamov bound. Via a technique of Thommesen [Tho83] and Guruswami and Indyk [GI04], our near-linear time capacity-achieving list-recoverable codes give a randomized construction of low-rate (up to 0.02) binary codes approaching the *Gilbert-Varshamov* (GV) bound, which admit near-linear time ($n^{1+o(1)}$) algorithms for unique decoding up to half their distance. Previous constructions which could achieve this either required at least quadratic decoding time, or else did not work for rates larger than 10^{-4}.

Our approach (discussed more below) is modular; given as an ingredient any (globally) high-rate list-recoverable code (with a polynomial time recovery algorithm), it yields high-rate (efficiently) locally list-recoverable code. To achieve the results advertised above, we instantiate this with either a random (non-efficient) linear code or with the (efficient) Algebraic Geometry (AG) subcodes of [GK16b]. Any improvements in these ingredient codes (for example, in the output list size of AG codes, which is near-constant but not quite) would translate immediately into improvements

in our constructions. The details of this construction are in the full version.

II. Related work

As mentioned above, list decoding and recovery, local decoding, and local list decoding and recovery, have a long and rich history in theoretical computer science. We mention here the results that are most directly related to ours mentioned above.

High-rate local list recovery.: Our main technical contribution is the construction of high-rate locally list-recoverable codes. There are two lines of work that are most related to this: the first is on local list recovery, and the second on high-rate (globally) list-recoverable codes.

Local list-decoding (which is a special case of local list recovery) first arose outside of coding theory, motivated by applications in complexity theory. For example, the Goldreich-Levin theorem in cryptography and the Kushilevitz-Mansour algorithm in learning theory are local list-decoding algorithms for Hadamard codes. Later, Sudan, Trevisan and Vadhan [STV01], motivated by applications in pseudorandomness, gave an algorithm for locally list-decoding Reed-Muller codes. Neither Hadamard codes nor the Reed-Muller codes of [STV01] are high-rate. However, similar ideas can be used to locally list-decode *lifted codes* [GK16a], and *multiplicity codes* [Kop15], which can be seen as high-rate variants of Reed-Muller codes. These algorithms work up to the so-called *Johnson bound.*

Briefly, the Johnson bound says that a code of distance δ is (α, L)-list-decodable, for reasonable L, when $\alpha \leq 1-\sqrt{1-\delta}$. This allows for high rate list decodable codes when δ is small, but there exist codes which are more list-decodable: the *list-decoding capacity theorem* implies that there are codes of distance δ which are (α, L)-list-decodable for α approaching the distance δ. The "capacity-achieving" list-decodable codes that we have been referring to are those which meet this latter result, which turns out to be optimal.

Like many list-decoding algorithms, the algorithms of [STV01], [Kop15], [GK16a] can be used for list-recovery as well (indeed, this type of approach was recently used in [GKO$^+$17] to obtain a local list-recovery algorithm for Reed-Muller codes.) However, as mentioned above they only work up to the Johnson bound for list-decoding, and this holds for list-recovery as well. However, for list-recovery, the difference between the Johnson bound and capacity is much more stark. Quantitatively, for (α, ℓ, L)-list-recovery, the Johnson bound requires $\alpha \leq 1 - \sqrt{\ell(1-\delta)}$, which is meaningless unless δ is very large; this requires the rate of the code to be small, less than $1/\ell$. In particular, these approaches do not give high-rate codes for list-recovery, and the Johnson bound appears to be a fundamental bottleneck.

The second line of work relevant to high-rate local list-recovery is that on high-rate *global* list-recovery. Here, there are two main approaches. The first is a line of work on capacity achieving list-decodable codes (also discussed more below). In many cases, the capacity achieving list-decoding algorithms for these codes are also high-rate list-recovery algorithms [GR08], [GW11], [Kop15], [GX13]. These algorithms are very global: they are all based on finding some interpolating polynomial, and finding this polynomial requires querying almost all of the coordinates. Thus, it is not at all obvious how to tweak these sorts of algorithms to achieve locally list-recoverable codes. The other line of work on high-rate global list-recovery is that of [HW15], which studies high-rate list-recoverable expander codes. While that algorithm is not explicitly local, it's not as clearly global as those previously mentioned (indeed, expander codes are known to have some locality properties [HOW15]). However, that work could only handle list-recovery with no errors—that is, it returns codewords that agree with *all* of the lists S_i, rather than a large fraction of them—and adapting it to handle errors seems like a challenging task.

Capacity achieving locally list decodable codes.: As mentioned above, one reason to seek high-rate codes is because of a transformation of Alon, Edmunds, and Luby [AEL95], recently highlighted in [KMRS16], which can, morally speaking, turn any high-rate code with a given property into a capacity achieving code with the same property.[1] This allows us to obtain *capacity achieving* locally list-decodable (or more generally, locally list recoverable) codes. This technique has been used frequently over the years [GI01], [GI02], [GI03], [GI04], [HW15], [KMRS16], [GKO$^+$17], and in particular [GKO$^+$17] used it for local list recovery. We borrow this result from them, and this immediately gives our capacity achieving locally list-decodable codes. Once we have these, they straightforwardly extend to near-linear time capacity-achieving (globally) list-decodable (or more generally, locally list recoverable) codes, simply by repeatedly running the local algorithm on each coordinate.

Capacity-achieving list-decodable codes.: We defined list-decodability above as a special case of list-recovery, but it is in fact much older. List-decodability

[1] We note however that this transformation does not apply to the property of list decoding, but just list recovery, and therefore we cannot use existing constructions of high-rate locally list decodable codes [Kop15], [GK16a] as a starting point for this transformation.

has been studied since the work of Elias and Wozencraft [Eli57], [Woz58] in the late 1950s, and the combinatorial limits are well understood. The *list-decoding capacity theorem,* mentioned earlier, states that there exist codes of rate approaching $1 - H_q(\alpha)$ which are (α, L)-list-decodable for small list size L, where $H_q(\alpha)$ is the q-ary entropy function (when q is large we have $1 - H_q(\alpha) \approx 1 - \alpha$). Moreover, any code of rate larger than that must have exponentially large list size.

The existence direction of the list-decoding capacity theorem follows from a random coding argument, and it wasn't until the Folded Reed-Solomon Codes of Guruswami and Rudra [GR08] that we had explicit constructions of codes which achieved list-decoding capacity. Since then, there have been many more constructions [Gur10], [GW11], [Kop15], [DL12], [GX12], [GX13], [GK16b], aimed at reducing the alphabet size, reducing the list size, and improving the speed of the recovery algorithm. We show the state-of-the-art in Table I below, along with our results.

Unique decoding up to the Gilbert-Varshamov bound.: The *Gilbert-Varshamov* (GV) bound [Gil52], [Var57] is a classical achievability result in coding theory. It states that there exist binary codes of relative distance $\delta \in (0, 1)$ and rate ρ approaching $1 - H_2(\delta)$, where H_2 is the binary entropy function. The proof is probabilistic: for example, it is not hard to see that a random linear code will do the trick. However, finding *explicit* constructions of codes approaching the GV bound remains one of the most famous open problems in coding theory. While we cannot find explicit constructions, we may hope for randomized constructions with efficient algorithms, and indeed this was achieved in the low-rate regime through a few beautiful ideas by Thommesen [Tho83] and follow-up work by Guruswami and Indyk [GI04].

Thommesen gave an efficient randomized construction of *concatenated codes* approaching the GV bound. Starting with a Reed-Solomon code over large alphabet, the construction is to concatenate each symbol with an independent random linear code. Later, [GI04] showed that these codes could in fact be efficiently decoded up to half their distance, in polynomial time, up to rates about 10^{-4}. Their idea was to use the list recovery properties of Reed-Solomon codes. The algorithm is then to list decode the small inner codes by brute force, and to run the efficient list-recovery algorithm for Reed-Solomon on the output lists of the inner codes: the combinatorial result of Thommesen ensures that the output list will contain a message that corresponds to the transmitted codeword.

In their work, [GI04] used the Guruswami-Sudan list recovery algorithm [GS99]. After decades of work [Ale02], [BB10], [CH11], [BHNW13], [CJN$^+$15], this algorithm can now be implemented to run in near-linear time, and so we already can achieve near-linear time unique decoding near the GV bound, up to rates about 10^{-4}. The reason for the bound on the rate is that the Guruswami-Sudan algorithm only works up to the aforementioned Johnson bound, which means it cannot tolerate as much error as capacity-achieving list-recoverable codes. It was noted by Rudra [Rud07] that replacing the Reed-Solomon codes with a capacity achieving list recoverable code (such as folded Reed-Solomon codes) can improve this rate limit up to about 0.02. However, those capacity achieving list recovery algorithms were slower (as in Table I), and this increases the running time back to at best quadratic.

The recent work [GKO$^+$17] also applied these techniques to give *locally decodable codes* approaching the Gilbert-Varshamov bound. These have query complexity n^β, and so in particular can be easily adapted to give a global decoding algorithm with running time $O(n^{1+\beta})$. However, the rate up to which the construction works approaches zero exponentially quickly in $1/\beta$.

Using exactly the same approach as these previous works, we may plug in our capacity achieving near-linear-time list-recooverable codes to obtain binary codes approaching the GV bound, which are uniquely decodable up to half their distance in time $n^{1+o(1)}$, and which work with rate matching Rudra's, $\rho = 0.02$.

Remark II.1. It is natural to ask whether our result can, like [GKO$^+$17], give locally decodable codes on the GV bound of higher rate. The main barrier is that our locality guarantees are for local decoding rather than local correction. It is an interesting open question whether one can use our techniques to extend the results of [GKO$^+$17] to higher rates.

List-decodability and local properties of tensor codes.: Our codes are constructed by taking tensor products of existing constructions of globally list-recoverable codes. Our approach is inspired by that of [GGR11], who study the list-decodability of tensor codes, although they do not address locality. It should be noted that the local *testing* properties of tensor codes have been extensively studied [BS06], [Val05], [CR05], [DSW06], [GM12], [BV09], [BV15], [Vid11], [Mei09], [Vid13]. Local properties of tensor codes have also been studied in the context of derandomization [MV05]. To the best of our knowledge, ours is the first work to study the local (list) *decodability* of tensor codes, rather than

Code	Reference	Construction	Alphabet size	List size	Decoding time
Folded RS codes, derivative codes	[GR08], [GW11], [Kop15]	Explicit	$\mathrm{poly}(n)$	$\mathrm{poly}(n)$	$n^{O(1/\varepsilon)}$
Folded RS subcodes	[DL12]	Explicit	$\mathrm{poly}(n)$	$O(1)$	n^2
(Folded) AG subcodes	[GX12], [GX13]	Monte Carlo	$O(1)$	$O(1)$	n^c
AG subcodes	[GK16b]	Explicit	$O(1)$	$\exp(\exp((\log^* n)^2))$	n^c
Tensor codes	This work	Explicit	$O(1)$	$\exp(\exp(\exp(\log^* n)))$	$n^{1.001}$

Table I

CONSTRUCTIONS OF LIST-DECODABLE CODES THAT ENABLE (α, L) LIST DECODING UP TO RATE $\rho = 1 - H_q(\alpha) - \varepsilon$, FOR CONSTANT ε. WE HAVE SUPPRESSED THE DEPENDENCE ON ε, EXCEPT WHERE IT APPEARS IN THE EXPONENT ON n IN THE DECODING TIME. ABOVE, c IS AN UNSPECIFIED CONSTANT. IN THE ANALYSIS OF THESE WORKS, IT IS REQUIRED TO TAKE $c \geq 3$. IT MAY BE THAT THESE APPROACHES COULD BE ADAPTED (WITH FASTER LINEAR-ALGEBRAIC METHODS) TO USE A SMALLER CONSTANT c, BUT IT IS NOT APPARENT; IN PARTICULAR WE CANNOT SEE HOW TO TAKE $c < 2$.

local testability.

III. OVERVIEW OF TECHNIQUES

Our main technical contribution is the construction of high-rate locally list-recoverable codes. While these are powerful objects, and result in new sub-linear and near-linear time algorithms for fundamental coding theoretic tasks, our techniques are actually quite simple (at least if we take certain previous works as a black box). We outline our approach below.

Our main ingredient is *tensor codes,* and the analysis given by Gopalan, Guruswami, and Raghevendra in [GGR11]. Given a linear code $C : \mathbb{F}^k \to \mathbb{F}^n$, consider the *tensor code* $C \otimes C : \mathbb{F}^{k \times k} \to \mathbb{F}^{n \times n}$; we will define the tensor product formally in Definition IV.6, but for now, we will treat the codewords of $C \otimes C$ as $n \times n$ matrices with the constraints that the rows and columns are all codewords of the original code C.

In [GGR11], it is shown that the tensor code $C \otimes C$ is roughly as list-decodable as C is. That work was primarily focused on combinatorial results, but their techniques are algorithmic, and it is these algorithmic insights that we leverage here. The algorithm is very simple: briefly, we imagine fixing some small combinatorial rectangle $S \times T \subseteq [n] \times [n]$ of "advice." Think of this advice as choosing the symbols of the codeword indexed by those positions. By alternately list decoding rows and columns, it can be shown that this advice uniquely determines a codeword c of $C \otimes C$. Finally, iterating over all possible pieces of advice yields the final list.

Inspired by their approach, our Main Technical Lemma V.2 says that if C is list-recoverable, then not only $C \otimes C$ is also list-recoverable, but in fact it is (approximately) locally list-recoverable. To understand the intuition, let us describe the algorithm just for $C \otimes C$, although our actual codes will require a higher tensor power $C^{\otimes t}$. Suppose that C is list-recoverable with output list size L. First, imagine fixing some advice $J := (a_1, \ldots, a_m) \in [L]^m$ for some (small integer) parameter m. This advice will determine an algorithm $\widetilde{A}_J$ which attempts to locally decode some message that corresponds to a close-by codeword c of $C \otimes C$, and the list we finally return will be the list of all algorithms $\widetilde{A}_J$ obtained by iterating over all possible advice.

Now, we describe the randomized algorithm $\widetilde{A}_J$, on input $(i, i') \in [n] \times [n]$.[2] Recall, $\widetilde{A}_J$ is allowed to query the input lists at every coordinate, and must produce a guess for the codeword value indexed by (i, i'). First, $\widetilde{A}_J$ chooses m random rows of $[n] \times [n]$. These each correspond to codewords in C, and $\widetilde{A}_J$ runs C's list-recovery algorithm on them to obtain lists $\mathcal{L}_1, \ldots, \mathcal{L}_m$ of size at most L each. Notice that this requires querying mn coordinates, which is roughly the square root of the length of the code (which is n^2). Then, $\widetilde{A}_J$ will use the advice $a_1, \ldots, a_m$ to choose codewords from each of these lists, and we remember the i''th symbol of each of these codewords. Finally, $\widetilde{A}_J$ again runs C's list-recovery algorithm on the i''th column, to obtain another list $\mathcal{L}$. Notice that our advice now has the same form as it does in [GGR11]: we have chosen a few symbols of a codeword of C. Now $\widetilde{A}_J$ chooses the codeword in $\mathcal{L}$ that agrees the most with this advice. The i''th symbol of this codeword is $\widetilde{A}_J$'s guess for the (i, i') symbol of the tensor codeword.

The above idea gives a code of length n which is locally list-recoverable with query complexity on the order of $\sqrt{n}$. This algorithm for $C \otimes C$ extends straightforwardly to $C^{\otimes t}$, with query complexity $n^{1/t}$.

[2]The algorithm $\widetilde{A}_J$ we describe decodes codeword symbols instead of message symbols, but since the codes we use are systematic this algorithm can also decode message symbols.

The trade-off is that the output list-size also grows with t. Thus, as we continue to take tensor powers, the locality improves, while the output list-size degrades; this allows for the trade-off between locality and output list-size mentioned in the introduction.

One issue with this approach is that this algorithm may in fact fail on a constant fraction of coordinates (i, i') (e.g., when a whole column is corrupted). To get around this, we first encode our message with a high-rate *locally decodable code,* before encoding it with the tensor code. For this, we use the codes of [KMRS16], which have rate that is arbitrarily close to 1, and which are locally decodable with $\exp(\sqrt{\log n})$ queries. This way, instead of directly querying the tensor code (which may give the wrong answer a constant fraction of the time), we instead use the outer locally decodable code to query the tensor code: this still does not use too many queries, but now it is robust to a few errors.

The final question is what to use as a base code. Because we are after high-rate codes we require C to be high-rate (globally) list recoverable. Moreover, since the tensor operation inflates the output list size by quite a lot, we require C to have small (constant or very slowly growing) output list size. Finally, we need C to be linear to get a handle on the rate of the tensor product. One possible choice is random linear codes, and these give a non-explicit and non-efficient construction with constant output list size. Another possible choice is the Algebraic Geometry subcodes of [GX13], [GK16b] which give explicit and efficient construction but with slowly growing output list size. However, we cannot quite use these latter codes as a black box, for two reasons. First, the analysis in [GX13] only establishes list-*decodability*, rather than list-recoverability. Fortunately, list-recoverability follows from exactly the same argument as list-decodability. Second, these codes are linear over a subfield, but are not themselves linear, while our arguments require linearity over the whole alphabet. Fortunately, we can achieve the appropriate linearity by concatenating the AG subcode with a small list-recoverable linear code, which exists by a probabilistic argument.

To summarize, our high-rate locally list-recoverable code is given by these ingredients: to encode a message x, we first encode it with the [KMRS16] locally decodable code. Then we encode this with a t-fold tensor product of a random linear code or a modified AG subcode and we are done. We go through the details of the argument sketched above in Section V; but first, we introduce some notation and formally define the notions that we will require.

IV. Definitions and preliminaries

For a prime power q we denote by $\mathbb{F}_q$ the finite field of q elements. For any finite alphabet Σ and for any pair of strings $x, y \in \Sigma^n$, the relative distance between x and y is the fraction of coordinates $i \in [n]$ 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.

A. Error-correcting codes

Let Σ be an alphabet and k, n be positive integers (the message length and the block length, respectively). A code is an injective map $C : \Sigma^k \to \Sigma^n$. The elements in the domain of C are called messages and the elements in the image of C are called codewords. If $\mathbb{F}$ is a finite field and Σ is a vector space over $\mathbb{F}$, we say that C is $\mathbb{F}$-linear if it is a linear transformation over $\mathbb{F}$ between the $\mathbb{F}$-vector spaces Σ^k and Σ^n. If $\Sigma = \mathbb{F}$ and C is $\mathbb{F}$-linear, we simply say that C is linear. The generating matrix of a linear code $C : \mathbb{F}^k \to \mathbb{F}^n$ is the matrix $G \in \mathbb{F}^{n \times k}$ such that $C(x) = G \cdot x$ for any $x \in \mathbb{F}^k$. We say that a code $C : \Sigma^k \to \Sigma^n$ is systematic if any message is the prefix of its image, that is, for any $x \in \Sigma^k$ there exists $y \in \Sigma^{n-k}$ such that $C(x) = (x, y)$.

The rate of a code $C : \Sigma^k \to \Sigma^n$ is the ratio $\rho := \frac{k}{n}$. The relative distance $\mathrm{dist}(C)$ of C is the minimum $\delta > 0$ such that for every pair of distinct messages $x, y \in \Sigma^k$ it holds that $\mathrm{dist}(C(x), C(y)) \geq \delta$. For a code $C : \Sigma^k \to \Sigma^n$ 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 message $x \in \Sigma^k$ (if any) which satisfies $\mathrm{dist}(C(x), w) \leq \alpha$.

The best known general trade-off between rate and distance of codes is the Gilbert-Varshamov bound, attained by random (linear) codes. For $x \in [0, 1]$ let $H_q(x)$ denote the q-ary entropy function.

Theorem IV.1 (Gilbert-Varshamov (GV) bound, [Gil52], [Var57])**.** *For any prime power q, $0 \leq \delta < 1 - \frac{1}{q}$, $0 \leq \rho < 1 - H_q(\delta)$, and sufficiently large n, a random linear code $C : \mathbb{F}_q^{\rho n} \to \mathbb{F}_q^n$ of rate ρ has relative distance at least δ with probability at least $1 - \exp(-n)$.*

B. List decodable and list recoverable codes

List decoding is a paradigm that allows one to correct more than $\delta/2$ fraction of errors by returning a small

list of messages that correspond to close-by codewords. More formally, for $\alpha \in [0,1]$ and an integer L we say that a code $C : \Sigma^k \to \Sigma^n$ is (α, L)-list decodable if for any $w \in \Sigma^n$ there are at most L different messages $x \in \Sigma^k$ which satisfy that $\mathrm{dist}(C(x), w) \leq \alpha$.

For list decoding concatenated codes it is useful to consider the notion of list recovery where one is given as input a small list of candidate symbols for each of the codeword coordinates, and is required to output a list of messages such that the corresponding codewords are consistent with the input lists. More concretely, for $\alpha \in [0,1]$ and integers ℓ, L we say that a code $C : \Sigma^k \to \Sigma^n$ is (α, ℓ, L)-list recoverable if for any $S \in \binom{\Sigma}{\ell}^n$ there are at most L different messages $x \in \Sigma^k$ which satisfy that $\mathrm{dist}(C(x), S) \leq \alpha$.

It is well-known that $1 - H_q(\alpha)$ is the list decoding capacity, that is, any q-ary code of rate above $1 - H_q(\alpha)$ cannot be list decoded from α fraction of errors with list size polynomial in the block length, and on the other hand, a random q-ary (linear) code of rate below $1 - H_q(\alpha)$ can be list decoded from α fraction of errors with small list size. The following corollary follows from Theorem 5.3 and Lemma 9.6 in [Gur01].

Corollary IV.2. *There is a constant c so that the following holds. Choose $\rho \in [0,1]$, $\varepsilon > 0$, and a positive integer ℓ. Suppose that q is a prime power which satisfies*

$$q \geq \max\{(1-\rho-\varepsilon)^{-c(1-\rho-\varepsilon)/\varepsilon}, (\rho+\varepsilon)^{-c(\rho+\varepsilon)/\varepsilon}, \ell^{c/\varepsilon}\}.$$

Then for sufficiently large n, a random linear code $C : \mathbb{F}_q^{\rho n} \to \mathbb{F}_q^n$ of rate ρ is $(1 - \rho - \varepsilon, \ell, q^{c\ell/\varepsilon})$-list recoverable with probability at least $1 - \exp(-n)$.

C. Locally decodable codes

Intuitively, a code C is said to be locally decodable if, given a codeword $C(x)$ that has been corrupted by some errors, it is possible to decode any coordinate of the corresponding message x by reading only a small part of the corrupted version of $C(x)$. Formally, it is defined as follows.

Definition IV.3 (Locally decodable code (LDC))**.** We say that a code $C : \Sigma^k \to \Sigma^n$ is (Q, α)-locally decodable if there exists a randomized algorithm A that satisfies the following requirements:

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

Remark IV.4. The success probability of $\frac{2}{3}$ can be amplified using sequential repetition; amplifying the success probability to $1 - e^{-t}$ requires increasing the query complexity by a multiplicative factor of $O(t)$.

Locally list decodable and list recoverable codes.: The following definition generalizes the notion of locally decodable codes to the setting of list decoding / recovery.

Definition IV.5 (Locally list recoverable code)**.** We say that a code $C : \Sigma^k \to \Sigma^n$ is (Q, α, ℓ, L)-locally list recoverable if there exists a randomized algorithm A that satisfies the following requirements:

- **Preprocessing:** A outputs L randomized algorithms $A_1, \ldots, A_L$.
- **Input:** Each A_j takes as input a coordinate $i \in [k]$, and also gets oracle access to a string $S \in \binom{\Sigma}{\ell}^n$.
- **Query complexity:** Each A_j makes at most Q queries to the oracle S.
- **Output:** For every codeword $C(x)$ that is α-close to S, 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 $i \in [k]$,

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

where the probability is over the internal randomness of A_j.

We say that A has preprocessing time T_{pre} if A outputs the description of the algorithms $A_1, \ldots, A_L$ in time at most T_{pre}, and has running time T if each A_j has running time at most T. Finally, we say that C is (Q, α, L)-locally list decodable if it is $(Q, \alpha, 1, L)$-locally list recoverable.

D. Tensor codes

A main ingredient in our constructions is the tensor product operation, defined as follows.

Definition IV.6 (Tensor codes)**.** Let $C_1 : \mathbb{F}^{k_1} \to \mathbb{F}^{n_1}$, $C_2 : \mathbb{F}^{k_2} \to \mathbb{F}^{n_2}$ be linear codes, and let $G_1 \in \mathbb{F}^{n_1 \times k_1}, G_2 \in \mathbb{F}^{n_2 \times k_2}$ be the generating matrices of C_1, C_2 respectively. Then the tensor code $C_1 \otimes C_2 : \mathbb{F}^{k_1 \times k_2} \to \mathbb{F}^{n_1 \times n_2}$ is defined as $(C_1 \otimes C_2)(M) = G_1 \cdot M \cdot G_2^T$.

The codewords of $C_1 \otimes C_2$ are $n_1 \times n_2$ matrices over $\mathbb{F}$ whose columns belong to the code C_1 and whose rows belong to the code C_2.

For a linear code C, let $C^{\otimes 1} := C$ and $C^{\otimes t} := C \otimes C^{\otimes(t-1)}$. If C has rate ρ and relative distance δ then $C^{\otimes t}$ has rate ρ^t and relative distance δ^t (see e.g. [Sud01], [DSW06]).

V. High-rate locally list recoverable codes

In Section V-A we show that high-rate tensor codes are *approximately locally list recoverable*, namely there exists a short list of local algorithms that can recover *most* of the coordinates of messages that correspond to near-by codewords. We then observe in Section V-B that by pre-encoding the message with a locally decodable code, the former codes can be turned into *locally list recoverable codes* for which the local algorithms can recover *all* the coordinates of messages that correspond to near-by codewords. Finally, we show in Section V-C how to instantiate the codes used in the process in order to obtain high-rate locally list recoverable codes with good performance.

A. Approximate local list recovery

To describe our approximate local list recovery algorithm it will be more convenient to require that the local algorithms recover *codeword symbols* as opposed to *message symbols*[3].

Definition V.1 (Approximately locally list recoverable code)**.** We say that a code $C : \Sigma^k \to \Sigma^n$ is $(Q, \alpha, \varepsilon, \ell, L)$-approximately locally list recoverable if there exists a randomized algorithm A that satisfies the following requirements:

- **Preprocessing:** A outputs L deterministic algorithms $A_1, \ldots, A_L$.
- **Input:** Each A_j takes as input a coordinate $i \in [n]$, and also gets oracle access to a string $S \in \binom{\Sigma}{\ell}^n$.
- **Query complexity:** Each A_j makes at most Q queries to the oracle S.
- **Output:** For every codeword $C(x)$ that is α-close to S, with probability at least $1 - \varepsilon$ over the randomness of A the following event happens: there exists some $j \in [L]$ such that

$$\Pr_{i \in [n]}[A_j(i) = C(x)_i] \geq 1 - \varepsilon,$$

  where the probability is over the choice of uniform random $i \in [n]$.

As before, we say that A has preprocessing time T_{pre} if A outputs the description of the algorithms $A_1, \ldots, A_L$ in time at most T_{pre}, and has running time T if each A_j has running time at most T.

Our main technical lemma is the following.

Lemma V.2 (Main technical)**.** *Suppose that $C : \mathbb{F}^k \to \mathbb{F}^n$ is a linear code of relative distance δ that is (α, ℓ, L)-(globally) list recoverable. Then for any $\widetilde{\varepsilon} > 0$, the tensor product code $\widetilde{C} := C^{\otimes t} : \mathbb{F}^{k^t} \to \mathbb{F}^{n^t}$ is $(\widetilde{Q}, \widetilde{\alpha}, \widetilde{\varepsilon}, \ell, \widetilde{L})$-approximately locally list recoverable for $\widetilde{\alpha} = \alpha \cdot \widetilde{\varepsilon} \cdot \delta^{O(t)}$,*

$$\widetilde{Q} = n \cdot \frac{\log^t L}{(\alpha \cdot \widetilde{\varepsilon})^{O(t)} \cdot \delta^{O(t^2)}},$$

and

$$\widetilde{L} = \exp\left(\frac{\log^t L}{(\alpha \cdot \widetilde{\varepsilon})^{O(t)} \cdot \delta^{O(t^2)}}\right).$$

Moreover, the approximate local list recovery algorithm for $\widetilde{C}$ has preprocessing time

$$\widetilde{T}_{pre} = \log n \cdot \exp\left(\frac{\log^t L}{(\alpha \cdot \widetilde{\varepsilon})^{O(t)} \cdot \delta^{O(t^2)}}\right),$$

and if the (global) list recovery algorithm for C runs in time T then the approximate local list recovery algorithm for $\widetilde{C}$ runs in time

$$\widetilde{T} = T \cdot \frac{\log^t L}{(\alpha \cdot \widetilde{\varepsilon})^{O(t)} \cdot \delta^{O(t^2)}}.$$

The proof of the above lemma will follow from repeated application of the following technical lemma.

Lemma V.3. *Suppose that $C : \mathbb{F}^k \to \mathbb{F}^n$ is a linear code of relative distance δ that is (α, ℓ, L)-(globally) list recoverable, and $C' : \mathbb{F}^{k'} \to \mathbb{F}^{n'}$ is a linear code that is $(Q', \alpha', \varepsilon', \ell, L')$-approximately locally list recoverable. Then for any $\widetilde{\varepsilon} \geq 100\varepsilon'/\delta$, the tensor product code $\widetilde{C} := C \otimes C' : \mathbb{F}^{k \times k'} \to \mathbb{F}^{n \times n'}$ is $(\widetilde{Q}, \widetilde{\alpha}, \widetilde{\varepsilon}, \ell, \widetilde{L})$-approximately locally list recoverable for $\widetilde{\alpha} = \frac{1}{10} \cdot \min\{\alpha' \cdot \delta, \alpha \cdot \widetilde{\varepsilon}\}$,*

$$\widetilde{Q} = O\left(\frac{\log(L/\widetilde{\varepsilon})}{(\delta \cdot \alpha' \cdot \widetilde{\varepsilon})^2}\right) \cdot Q' + n,$$

and

$$\widetilde{L} = \exp\left(\frac{\log L' \cdot \log(L/\widetilde{\varepsilon})}{(\delta \cdot \alpha' \cdot \widetilde{\varepsilon})^2}\right).$$

Moreover, if the (global) list recovery algorithm for C runs in time T, and the approximate local list recovery algorithm for C' has preprocessing time T'_{pre} and runs in time T', then the approximate local list recovery algorithm for $\widetilde{C}$ has preprocessing time

$$\widetilde{T}_{pre} = O\left(\frac{\log(L/\widetilde{\varepsilon})}{(\delta \cdot \alpha' \cdot \widetilde{\varepsilon})^2}\right) \cdot (\log n + T'_{pre}) + \exp\left(\frac{\log L' \cdot \log(L/\widetilde{\varepsilon})}{(\delta \cdot \alpha' \cdot \widetilde{\varepsilon})^2}\right),$$

and runs in time

$$\widetilde{T} = O\left(\frac{\log(L/\widetilde{\varepsilon})}{(\delta \cdot \alpha' \cdot \widetilde{\varepsilon})^2}\right) \cdot T' + T.$$

[3] In our constructions we shall use systematic codes and so recovery of codeword symbols will imply also recovery of message symbols.

The proof of Lemma V.2 proceeds by repeated application of Lemma V.3 to the code C, and the details can be found in the full version of this paper.

Lemma V.3, states that there is a randomized algorithm $\widetilde{A}$ that outputs a list of (deterministic) local algorithms $\widetilde{A}_1, \ldots, \widetilde{A}_{\widetilde{L}}$ such that for any codeword $\widetilde{c} \in C \otimes C'$ that is consistent with most of the input lists, with high probability over the randomness of A, there exists some $\widetilde{A}_i$ in the output list that computes correctly most of the coordinates of $\widetilde{c}$.

The algorithm $\widetilde{A}$ first chooses a uniform random subset $R \subseteq [n]$ of rows of size $m := O\left(\frac{\log(L/\widetilde{\varepsilon})}{(\delta \cdot \alpha' \cdot \widetilde{\varepsilon})^2}\right)$. It then runs for each row $r \in R$, independently, the approximate local list recovery algorithm A' for C', let $A'^r_1, \ldots, A'^r_{L'}$ denote the output algorithms on row r. Finally, for every possible choice of a single local algorithm $A^r_{a_r}$ per each of the rows $r \in R$, the algorithm $\widetilde{A}$ outputs a local algorithm denoted $\widetilde{A}_J$ where $J := (a_r)_{r \in R} \in [L']^R$. The formal definition of the algorithm $\widetilde{A}_J$ is given in 1.

Algorithm 1 The approximate local list recovery algorithm for $C \otimes C'$.

function $\widetilde{A}_J((i, i') \in [n] \times [n'])$

▷ $\widetilde{A}_J$ receives oracle access to a matrix of lists $S \in \binom{\mathbb{F}}{\ell}^{n \times n'}$

▷ $J = (a_r)_{r \in R} \in [L']^R$

for $r \in R$ **do**

Run $A^r_{a_r}$ on input i' and oracle access to the rth row $S|_{\{r\} \times [n']}$.

Let $c'_r \leftarrow A^r_{a_r}(i')$.

▷ c'_r is a candidate for the symbol at position $(r, i') \in [n] \times [n']$.

end for

▷ At this point, we have candidate symbols for every position in $R \times \{i'\}$.

Run the (global) list recovery algorithm for C on the i'th column $S|_{[n] \times \{i'\}}$.

Let $\mathcal{L} \subseteq \mathbb{F}^n$ denote the output list.

Choose a codeword $c \in \mathcal{L}$ such that $c|_R$ is closest to $(c'_r)_{r \in R}$ (breaking ties arbitrarily).

Return: c_i

end function

The complete analysis of the correctness and complexity of Algorithm 1 can be found in the full version.

B. Local list recovery

Next we show that the approximately locally list recoverable codes of Lemma V.2 can be turned into locally list recoverable codes by pre-encoding the message with a locally decodable code.

Lemma V.4. *Suppose that $C : \mathbb{F}^{\rho n} \to \mathbb{F}^n$ is a* systematic *linear code of rate ρ and relative distance δ that is (α, ℓ, L)-(globally) list recoverable, and $\widehat{C} : \mathbb{F}^{\widehat{k}} \to \mathbb{F}^{(\rho n)^t}$ is $(\widehat{Q}, \widehat{\alpha})$-locally decodable. Then $\widetilde{C} := C^{\otimes t}(\widehat{C}) : \mathbb{F}^{\widehat{k}} \to \mathbb{F}^{n^t}$ is $(\widetilde{Q}, \widetilde{\alpha}, \ell, \widetilde{L})$-locally list recoverable for $\widetilde{\alpha} = \alpha \cdot \widehat{\alpha} \cdot \rho^t \cdot \delta^{O(t)}$,*

$$\widetilde{Q} = \widehat{Q} \cdot n \cdot \frac{\log^t L}{(\alpha \cdot \widehat{\alpha})^{O(t)} \cdot (\rho \cdot \delta)^{O(t^2)}},$$

and

$$\widetilde{L} = \exp\left(\frac{\log^t L}{(\alpha \cdot \widehat{\alpha})^{O(t)} \cdot (\rho \cdot \delta)^{O(t^2)}}\right).$$

Moreover, the local list recovery algorithm for $\widetilde{C}$ has preprocessing time

$$\widetilde{T}_{pre} = \exp\left(\frac{\log^t L}{(\alpha \cdot \widehat{\alpha})^{O(t)} \cdot (\rho \cdot \delta)^{O(t^2)}}\right) \cdot \log n,$$

and if the (global) list recovery algorithm for C runs in time T and the local decoding algorithm for $\widehat{C}$ runs in time $\widehat{T}$ then the local list recovery algorithm for $\widetilde{C}$ runs in time

$$\widetilde{T} = \widehat{T} + \widehat{Q} \cdot T \cdot \frac{\log^t L}{(\alpha \cdot \widehat{\alpha})^{O(t)} \cdot (\rho \cdot \delta)^{O(t^2)}}.$$

Intuitively, the proof works as follows: to recover the ith message symbol, x_i, run the local decoder of the inner code $\widehat{C}$ to obtain a set of $\widehat{Q}$ indices in $\mathbb{F}^{(\rho n)^t}$ that, if queried, would allow you to recover x_i. Since the code $\overline{C}$ is systematic, those symbols correspond to symbols in the big code $\widetilde{C}$. Use the approximate local list recovery algorithm for $\overline{C}$ to obtain $\overline{L}$ guesses for each of these $\widehat{Q}$ symbols. Finally, for each of these $\overline{L}$ sets of $\widehat{Q}$ "guesses" run the local decoding algorithm for $\widehat{C}$ to obtain $\overline{L}$ guesses for x_i. Since $\overline{C}$ is only approximately locally list recoverable, there will be a subset of symbols on which the approximate local list decoder fails, but by carefully choosing parameters, these errors can be handled by the local decoding procedure of $\widehat{C}$.

It is not hard to see that the query complexity of this algorithm will be $\widehat{Q}$ times the query complexity of $C^{\otimes t}$, and the output list size will be the same as that of $C^{\otimes t}$.

Below, we outline the algorithm. Let $(\overline{A}_1, \cdots, \overline{A}_{\overline{L}}) \leftarrow \overline{A}(\cdot)$ be the approximate local list recovery algorithms for $\overline{C}$. In Algorithm 2 we describe the local list recovery algorithms $\widetilde{A}_1, \ldots, \widetilde{A}_{\overline{L}}$ for the code $\widetilde{C} := \overline{C}(\widehat{C})$.

The analysis of the correctness and running time of Algorithm 2 can be found in the full version.

Algorithm 2 The local list recovery algorithm for $\widetilde{C} := C^{\otimes t}(\widehat{C})$.

function $\widetilde{A}_j(i \in [\widehat{k}])$

▷ $\widetilde{A}_j$ receives oracle access to lists $S \in \binom{\mathbb{F}}{\ell}^{n^t}$

Run the local decoding algorithm for $\widehat{C}$ on input i to obtain a set of $\widehat{Q}$ indices that the local decoder would query.

Let $R \subseteq [(\rho n)^t]$ be the subset of indices that would be queried.

Let $\overline{R} \subseteq [n^t]$ be the indices in $\overline{C}$ encoding the indices of R.

▷ $\overline{R}$ exists and $|\overline{R}| = |R| = \widehat{Q}$ because $\overline{C}$ is systematic.

for $\overline{r} \in \overline{R}$ **do**

Let $c_j^{(\overline{r})} \leftarrow \overline{A}_j(\overline{r})$ (on oracle access to S)

end for

Run the local decoder for $\widehat{C}$ on input $\{c_j^{(\overline{r})}\}_{\overline{r} \in \overline{R}}$ to obtain a guess $x_i^{(j)}$ for the ith symbol of the message

Return: $x_i^{(j)}$

end function

C. Instantiations

In what follows we shall instantiate Lemma V.4 in two ways. For both, we will use the high-rate LDCs of [KMRS16] as the code $\widehat{C}$.

To obtain efficiently encodable (in nearly-linear time) and efficiently list recoverable (in sub-linear time) codes, we use a modification of the Algebraic Geometry subcodes studied in [GX13], [GK16b] as the code C. These codes have constant alphabet size, but slightly super-constant output list size, which means that our efficient construction will as well.

If we do not require efficiency, we can use a random linear code (via Corollary IV.2) as the code C. This yields a code $\widetilde{C}$ that is not linear-time encodable, nor efficiently list recoverable, but it does achieve small locality together with constant alphabet size and constant output list size.

Theorem V.5 (High-rate locally list recoverable codes, efficient)**.** *There is a constant c so that the following holds. Choose $\varepsilon > 0$ and a positive integer ℓ. Let $\{t_n\}_n$ be a sequence of positive integers, non-decreasing with n, so that t_0 is sufficiently large and*

$$t_n \leq \sqrt{\frac{\varepsilon \log_q(n)}{c\ell}}.$$

For each choice of t choose $s = s(t)$ so that $s \geq \max\{1/\varepsilon^c, c(\log \ell)t/\varepsilon\}$ is even. Then there exists an infinite family of $\mathbb{F}_2$-linear codes $\{C_n\}_n$ such that the following holds. Below, to simplify notation we use t instead of t_n and s instead of $s(t_n)$.

1) *$C_n : \mathbb{F}_{2^s}^{(1-\varepsilon)n} \to \mathbb{F}_{2^s}^n$ has rate $1-\varepsilon$ and relative distance at least $(\Omega(\varepsilon/t))^{2t}$.*
2) *C_n is (Q, α, ℓ, L)-locally list recoverable for $\alpha = (\varepsilon/t)^{O(t)}$,*

$$Q = n^{1/t} \cdot 2^{O(\sqrt{\log n \cdot \log\log n})} \cdot e^{\left(\frac{t^2 \ell s}{\varepsilon} \cdot \exp(\log^* n) + t \log s\right)},$$

and

$$L = \exp\left(\exp\left(\frac{t^2 \ell s}{\varepsilon} \cdot \exp(\log^* n) + t \log s\right)\right).$$

3) *The local list recovery algorithm for C_n has preprocessing time*

$$T_{pre} = e^{\left(e^{\left(\frac{t^2 \ell s}{\varepsilon} \cdot \exp(\log^* n) + t \log s\right)}\right)} \cdot \log n,$$

and running time

$$T = n^{O(1/t)} \cdot 2^{O(\sqrt{\log n \cdot \log\log n})} \cdot e^{\left(\left(\frac{t\ell s}{\varepsilon} \cdot \exp(\log^* n) + \log s\right)\right)}.$$

4) *C_n can be encoded in time*

$$n \cdot 2^{O(\sqrt{\log n \cdot \log\log n})} + t \cdot n^{1+O(1/t)}.$$

In particular, when $\varepsilon, \ell, t_n = t, s$ are constant we get that $\alpha = \Omega(1)$, $Q = n^{1/t+o(1)}$, $L = \exp(\exp(\exp(\log^* n)))$, $T_{pre} = \log^{1+o(1)} n$, $T = n^{O(1/t)}$, and encoding time is $n^{1+O(1/t)}$.

The proofs of Theorem V.5 as well as the inefficient construction can be found in the full version.

The technique of [AEL95] can be used to transform the above codes into capacity-achieving locally list recoverable codes over a large (but constant) alphabet. These capacity-achieving codes can then be used to obtain codes that are encodable in near-linear time and uniquely decodable up to half the GV bound. The details are in the full version.

Acknowledgements.: The second author would like to thank Swastik Kopparty for raising the question of obtaining capacity achieving locally list decodable codes which was a direct trigger for this work as well as previous work [KMRS16], [GKO$^+$17], and Sivakanth Gopi, Swastik Kopparty, Rafael Oliveira and Shubhangi Saraf for many discussions on this topics. The current collaboration began during the Algorithmic Coding Theory Workshop at ICERM, we thank ICERM for their hospitality.

REFERENCES

[AEL95] Noga Alon, Jeff Edmonds, and Michael Luby. Linear time erasure codes with nearly optimal recovery. In *FOCS*, pages 512–519. IEEE Computer Society, 1995.

[Ale02] Michael Alekhnovich. Linear diophantine equations over polynomials and soft decoding of reed-solomon codes. In *FOCS*, pages 439–448. IEEE, 2002.

[BB10] Peter Beelen and Kristian Brander. Key equations for list decoding of Reed Solomon codes and how to solve them. *Journal of Symbolic Computation*, 45(7):773–786, 2010.

[BHNW13] Peter Beelen, Tom Hoholdt, Johan S.R. Nielsen, and Yingquan Wu. On rational interpolation-based list-decoding and list-decoding binary goppa codes. *IEEE Trans. Inf. Th.*, 59(6):3269–3281, 2013.

[BS06] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Struct. Algorithms*, 28(4):387–402, 2006.

[BV09] Eli Ben-Sasson and Michael Viderman. Tensor products of weakly smooth codes are robust. *Theory of Computing*, 5(1):239–255, 2009.

[BV15] Eli Ben-Sasson and Michael Viderman. Composition of semi-LTCs by two-wise tensor products. *Computational Complexity*, 24(3):601–643, 2015.

[CH11] Henry Cohn and Nadia Heninger. Ideal forms of Coppersmith's theorem and Guruswami-Sudan list decoding. In *ICS*, pages 298–308, 2011.

[CJN+15] Muhammad F.I. Chowdhury, Claude-Pierre Jeannerod, Vincent Neiger, Eric Schost, and Gilles Villard. Faster algorithms for multivariate interpolation with multiplicities and simultaneous polynomial approximations. *IEEE Trans. Inf. Th.*, 61(5):2370–2387, 2015.

[CR05] Don Coppersmith and Atri Rudra. On the robust testability of tensor products of codes. ECCC TR05-104, 2005.

[DL12] Zeev Dvir and Shachar Lovett. Subspace Evasive Sets. In *STOC*, pages 351–358, October 2012.

[DSW06] Irit Dinur, Madhu Sudan, and Avi Wigderson. Robust local testability of tensor products of LDPC codes. In *RANDOM*, pages 304–315, 2006.

[Eli57] Peter Elias. List decoding for noisy channels. Technical Report 335, MIT, September 1957.

[GGR11] Parikshit Gopalan, Venkatesan Guruswami, and Prasad Raghavendra. List decoding tensor products and interleaved codes. *SIAM Journal on Computing*, 40(5):1432–1462, 2011.

[GI01] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *FOCS*, pages 658–667. IEEE, October 2001.

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *STOC*, pages 812–821, 2002.

[GI03] Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *STOC*, pages 126–135, New York, NY, USA, 2003.

[GI04] Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting Gilbert-Varshamov bound for low rates. In *SODA*, pages 756–757, 2004.

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

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

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

[GKO+17] Sivakanth Gopi, Swastik Kopparty, Rafael Oliveira, Noga Ron-Zewi, and Shubhangi Saraf. Locally testable and locally correctable codes approaching the gilbert-varshamov bound. In *SODA*, pages 2073–2091. SIAM, 2017.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32. ACM, 1989.

[GM12] Oded Goldreich and Or Meir. The tensor product of two good codes is not necessarily locally testable. *Inf. Proces. Lett.*, 112(8-9):351–355, 2012.

[GNP+13] Anna C. Gilbert, Hung Q. Ngo, Ely Porat, Atri Rudra, and Martin J. Strauss. ℓ_2/ℓ_2-foreach sparse recovery with low risk. In *Automata, Languages, and Programming*, volume 7965, pages 461–472. 2013.

[GR08] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Trans. Inf. Th.*, 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), 1999.

[Gur01] Venkatesan Guruswami. *List decoding of error-correcting codes*. PhD thesis, MIT, 2001.

[Gur10] Venkatesan Guruswami. Cyclotomic function fields, artin–frobenius automorphisms, and list error correction with optimal rate. *Algebra & Number Theory*, 4(4):433–463, 2010.

[GW11] Venkatesan Guruswami and Carol Wang. Optimal rate list decoding via derivative codes. In *RANDOM '11*, 2011.

[GX12] Venkatesan Guruswami and Chaoping Xing. Folded codes from function field towers and improved optimal rate list decoding. In *STOC*, pages 339–350. ACM, 2012.

[GX13] Venkatesan Guruswami and Chaoping Xing. List decoding reed-solomon, algebraic-geometric, and gabidulin subcodes up to the singleton bound. In *STOC*, pages 843–852. ACM, 2013.

[HIOS15] Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO*, volume 9216 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2015.

[HOW15] Brett Hemenway, Rafail Ostrovsky, and Mary Wootters. Local correctability of expander codes. *Inf. Comput*, 243:178–190, 2015.

[HW15] Brett Hemenway and Mary Wootters. Linear-time list recovery of high-rate expander codes. In *ICALP '15*, volume 9134, pages 701–712, 2015.

[INR10] Piotr Indyk, Hung Q. Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *SODA*, pages 1126–1142, Philadelphia, PA, USA, 2010.

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

[KMRS16] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High rate locally-correctable and locally-testable codes with sub-polynomial query complexity. In *STOC*, pages 25–32. ACM Press, 2016.

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

[Mei09] Or Meir. Combinatorial construction of locally testable codes. *SIAM J. Comput.*, 39(2):491–544, 2009.

[MV05] Peter Bro Miltersen and N Variyam Vinodchandran. Derandomizing arthur–merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[NPR12] Hung Q. Ngo, Ely Porat, and Atri Rudra. Efficiently Decodable Compressed Sensing by List-Recoverable Codes and Recursion. In Christoph Dürr and Thomas Wilke, editors, *STACS*, volume 14, pages 230–241, Dagstuhl, Germany, 2012.

[Rud07] Atri Rudra. *List Decoding and Property Testing of Error Correcting Codes*. PhD thesis, University of Washington, 2007.

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

[Sud01] Madhu Sudan. Algorithmic introduction to coding theory (lecture notes), 2001.

[Tho83] Christian Thommesen. The existence of binary linear concatenated codes with reed - solomon outer codes which asymptotically meet the gilbert- varshamov bound. *IEEE Trans. Information Theory*, 29(6):850–853, 1983.

[Val05] Paul Valiant. The tensor product of two codes is not necessarily robustly testable. In *RANDOM*, pages 472–481. Springer, 2005.

[Var57] R. R. Varshamov. Estimate of the number of signals in error correcting codes. *Doklady Akadamii Nauk*, pages 739–741, 1957.

[Vid11] Michael Viderman. A combination of testability and decodability by tensor products. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:87, 2011.

[Vid13] Michael Viderman. Strong LTCs with inverse poly-log rate and constant soundness. In *FOCS*, pages 330–339. IEEE Computer Society, 2013.

[Woz58] J.M. Wozencraft. List decoding. Quarterly progress report, MIT, 1958.