Linear-time Erasure List-decoding of Expander Codes

Noga Ron-Zewi
Department of Computer Science
University of Haifa
noga@cs.haifa.ac.il

Mary Wootters
Department of Computer Science
Department of Electrical Engineering
Stanford University
marykw@stanford.edu.

Gilles Zémor
Institut de Mathématiques de Bordeaux
UMR 5251
Université de Bordeaux
zemor@math.u-bordeaux.fr

***Abstract*—We give a linear-time erasure list-decoding algorithm for expander codes. More precisely, let $r > 0$ be any integer. Given an inner code $\mathcal{C}_0$ of length d, and a d-regular bipartite expander graph G with n vertices on each side, we give an algorithm to list-decode the expander code $\mathcal{C} = \mathcal{C}(G, \mathcal{C}_0)$ of length nd from approximately $\delta\delta_r nd$ erasures in time $n \cdot \mathrm{poly}(d2^r/\delta)$, where δ and δ_r are the relative distance and the r'th generalized relative distance of $\mathcal{C}_0$, respectively. To the best of our knowledge, this is the first linear-time algorithm that can list-decode expander codes from erasures beyond their (designed) distance of approximately $\delta^2 nd$.**

To obtain our results, we show that an approach similar to that of (Hemenway and Wootters, *Information and Computation*, 2018) can be used to obtain such an erasure-list-decoding algorithm with an exponentially worse dependence of the running time on r and δ; then we show how to improve the dependence of the running time on these parameters.

A full version of this paper is accessible at: https://arxiv.org/abs/2002.08579

I. Introduction

In coding theory, the problem of *list-decoding* is to return all codewords that are close to some received word z; in *algorithmic list-decoding,* the problem is to do so efficiently. While there has been a great deal of progress on algorithmic list-decoding in the past two decades [1]–[10], most work has relied crucially on algebraic constructions, and thus it is interesting to develop combinatorial tools to construct efficiently list-decodable codes with good parameters.

In this work, we consider the question of list-decoding *expander codes,* introduced by Sipser and Spielman in [11]. The expander code $\mathcal{C}(G, \mathcal{C}_0)$ is a linear code constructed from a d-regular bipartite expander graph G and a linear inner code $\mathcal{C}_0 \subseteq \mathbb{F}_2^d$. A codeword of $\mathcal{C}(G, \mathcal{C}_0)$ is a vector in $\mathbb{F}_2^{E(G)}$ which is a labeling of edges in G. The constraints are that, for each vertex v of G, the labels on the d edges incident to v form a codeword in $\mathcal{C}_0$.

Expander codes are notable for their efficient unique decoding algorithms [11]–[20]. However, little is known about the algorithmic list-decodability of expander codes, and it is an open problem to find a family of expander codes that admit fast linear-time list-decoding algorithms with good parameters. Motivated by this open problem, our main contribution is a linear-time algorithm for list decoding expander codes *from erasures.*

Erasure-list-decoding is a variant of list-decoding where the received word z may have some symbols which are "$\perp$" (erasures). More formally, let $\mathcal{C} \subseteq \mathbb{F}_2^N$ be a binary code of length N. For $z \in (\mathbb{F}_2 \cup \perp)^N$, define

$$\mathrm{List}_{\mathcal{C}}(z) := \{c \in \mathcal{C} \,:\, c_i = z_i \text{ whenever } z_i \neq \perp\}.$$

We say that $\mathcal{C}$ is *erasure-list-decodable* from e erasures with list size L if for any $z \in (\mathbb{F}_2 \cup \{\perp\})^N$ with at most e symbols equal to $\perp$, $|\mathrm{List}_{\mathcal{C}}(z)| \leq L$.

Erasure list-decoding has been studied before [21], [23]–[28], motivated both by standard list-decoding and as an interesting combinatorial and algorithmic question in its own right. It is known that the erasure-list-decodability of a linear code is precisely captured by its generalized distances. The *r'th (relative)*[1] *generalized distance* δ_r of a linear code $\mathcal{C} \subseteq \mathbb{F}_2^N$ is

$$\delta_r := \frac{1}{N} \min_V \big| \{i \,:\, \exists v \in V, v_i \neq 0\} \big|,$$

where the minimum is taken over all linear subspaces $V \subseteq \mathcal{C}$ of dimension r. Thus, δ_1 coincides with the traditional *(relative) distance* δ of the code, which for linear codes equals the minimum relative weight of any nonzero codeword. The generalized distances of a linear code $\mathcal{C}$ characterize its erasure list-decodability:

Lemma 1 ([21]): Let $\mathcal{C} \subseteq \mathbb{F}_2^N$ be a linear code. Then $\mathcal{C}$ is erasure-list-decodable from e erasures with list size L if and only if $\delta_r(\mathcal{C}) > e/N$, where $r = 1 + \lfloor \log_2(L) \rfloor$.

If $\mathcal{C}$ is linear, then it can be erasure list-decoded in polynomial time by solving a linear system. Thus, the combinatorial result of Lemma 1 comes with a polynomial-time algorithm.

Our main result is a linear-time erasure list-decoding algorithm for expander codes beyond the (designed) minimum distance.

Theorem 2: Let $\mathcal{C}_0 \subseteq \mathbb{F}_2^d$ be a linear code with distance δ and r'th generalized distance δ_r. Let $G = (L \cup R, E)$ be a bipartite d-regular expander graph on $2n$ vertices with expansion $\lambda = \max\{|\lambda_i|, |\lambda_i| \neq d\}$. Let $\mathcal{C} = \mathcal{C}(G, \mathcal{C}_0)$ be the expander code that results. Let $\varepsilon > 0$, and suppose that $\frac{\lambda}{d} \leq \frac{\varepsilon^2 \delta^2}{2^{r+4}}$.

[1]We will work with the relative generalized distances (that is, measured as a fraction of coordinates). We will omit the adjective "relative" to describe these quantities in the future.

Then there is an algorithm LISTDECODE which, given a received word $z \in (\mathbb{F}_2 \cup \{\perp\})^E$ with at most $(1-\varepsilon)\delta\delta_r dn$ erasures, runs in time $n \cdot \text{poly}\left(\frac{2^r d}{\varepsilon\delta}\right)$, and returns a matrix $L \in \mathbb{F}_2^{nd \times a}$ and a vector $\ell \in \mathbb{F}_2^{nd}$ so that $\mathcal{L} := \text{List}_{\mathcal{C}}(z) = \{Lx + \ell : x \in \mathbb{F}_2^a\}$ where $a := \dim(\mathcal{L})$ satisfies $a \leq \frac{2^{2r+7}}{\varepsilon^4\delta^4}$.

Because $\delta_r > \delta$ for any non-trivial linear code (any code of dimension > 1), the radius that Theorem 2 achieves is beyond the (designed) minimum distance of $\mathcal{C}$, which is approximately $\delta^2 dn$. To the best of our knowledge, this is the first linear-time list-decoding algorithm for expander codes that achieves this with a non-trivial list size.

In light of Lemma 1, our algorithm would ideally be able to list-decode an expander code $\mathcal{C}$ from up to a $\delta_r(\mathcal{C})$ fraction of erasures with list size 2^{r-1} for any $r \geq 1$. The r'th generalized distance $\delta_r(\mathcal{C})$ is not well-understood[2] so we are unable to make such a guarantee for general r. However, under some conditions on $\mathcal{C}_0$, we are able to make such a guarantee for $r = 2$. In the full version of the paper [22], we show that, in the setting of Theorem 2, $\delta_2(cC) \geq (1-\varepsilon)\delta\delta_2$, provided that $\delta_2(\mathcal{C}_0) \leq 2\delta(\mathcal{C}_0)$. This implies that our algorithm can erasure list-decode $\mathcal{C}$ in linear time up to a $\delta(\mathcal{C}_0)\delta_2(\mathcal{C}_0)$ fraction of erasures, with an output list of size 2.

More generally, our algorithm can still achieve this for some values of r because it is guaranteed to return a list of the "correct" size. That is, if r' is such that $\delta_{r'}(\mathcal{C}) < \delta(\mathcal{C}_0)\delta_r(\mathcal{C}_0)$ for some $r = O(1)$, our algorithm will run in linear time and return a list of size $2^{r'-1}$ given a $\delta_{r'}(\mathcal{C})$ fraction of erasures.

A. Notation and Definitions

Let $G = (L \cup R, E)$ be a bipartite graph.[3] For a vertex $v \in L \cup R$, let $\Gamma(v)$ denote the set of vertices adjacent to v. For $S \subseteq L$ and $T \subseteq R$, let $E(S,T)$ denote the set of edges with endpoints in $S \cup T$, and for $A \subseteq L \cup R$, let $E(A) := E(A \cap L, A \cap R)$.

Let G be a d-regular graph on n vertices, and let $\lambda_1 = d \geq \lambda_2 \geq \ldots \geq \lambda_n$ be the eigenvalues of the adjacency matrix of G. For $n \geq 3$ we define the *expansion* of G to be $\lambda := \max\{|\lambda_i|, \lambda_i \neq \pm d\}$. The bipartite graph version of the *Expander Mixing Lemma* reads:

Theorem 3 (Expander Mixing Lemma, see e.g. *[32]):* Suppose that $G = (L \cup R, E)$ is a d-regular expander graph on $2n$ vertices with expansion λ. Then for any $S \subseteq L$ and $T \subseteq R$,

$$\left| E(S,T) - \frac{d}{n}|S||T| \right| \leq \lambda\sqrt{|S||T|}.$$

We now define expander codes. Let $G = (L \cup R, E)$ be a bipartite d-regular expander graph on $2n$ vertices. Let $\mathcal{C}_0 \subseteq \mathbb{F}_2^d$ be a linear code, called the *inner code.* Fix an order on the edges incident to each vertex of G, and let $\Gamma_i(v)$ denote the i'th neighbor of v. The expander code $\mathcal{C} := \mathcal{C}(G, \mathcal{C}_0)$ is defined as the set of all labelings of the edges of G that respect the inner code $\mathcal{C}_0$.

[2]We note that our results imply a weak bound on this distance, namely that $\delta_r(\mathcal{C})$ is at least (approximately) $\delta(\mathcal{C}_0) \cdot \delta_{\Theta(\log(r))}(\mathcal{C}_0)$.

[3]In this paper we only consider undirected connected graphs.

Definition 4 (Expander Code): Let $\mathcal{C}_0 \subseteq \mathbb{F}_2^d$ be a linear code, and let $G = (L \cup R, E)$ be a bipartite d-regular expander graph on $2n$ vertices. The expander code $\mathcal{C}(G, \mathcal{C}_0) \subseteq \mathbb{F}_2^E$ is the linear code of length nd, such that for $c \in \mathbb{F}_2^E$, $c \in \mathcal{C}$ if and only if, for all $v \in L \cup R$,

$$\left(c_{(v,\Gamma_1(v))}, c_{(v,\Gamma_2(v))}, \ldots, c_{(v,\Gamma_d(v))}\right) \in \mathcal{C}_0.$$

It is not hard to see that if $\mathcal{C}_0 \subseteq \mathbb{F}_2^d$ is a linear code of rate R, then $\mathcal{C}(G, \mathcal{C}_0) \subseteq \mathbb{F}_2^E$ is a linear code of rate at least $2R - 1$. Moreover, it is known that expander codes have distance at least $\delta(\mathcal{C}_0)(\delta(\mathcal{C}_0) - \lambda/d)$ [11], [12], and that they can be uniquely decoded from this fraction of erasures in linear time; see the full version for details.

B. Technical Overview

In this section, we give a brief overview of our approach. The basic idea is similar to the approach in [27]; however, as we discuss more in Section I-C below, in that work the goal was *list-recovery,* a generalization of list-decoding. In this work we can do substantially better by restricting our attention to list-decoding, as well as by tightening the analysis of [27].

Let $G = (L \cup R, E)$ be a bipartite d-regular expander graph, and let $\mathcal{C}_0 \subseteq \mathbb{F}_2^d$ be a linear code with distance δ and r-th generalized distance δ_r. Since the inner code $\mathcal{C}_0$ is linear and has r'th generalized distance δ_r, there is an $O(d^3)$-time algorithm to erasure list-decode $\mathcal{C}_0$ from up to $\delta_r d$ erasures. Our first step will be to do this at every vertex $v \in L \cup R$ that we can, to produce a list L_v at each such vertex.

In order to "stitch together" these lists, we define a notion of *equivalence* between edges, similar to the notion in [27]. Suppose that (u,v) and (w,v) are edges incident to a vertex v, and that there is some $b \in \mathbb{F}_2$ such that for any $c \in L_v$, $c_{(u,v)} = b + c_{(w,v)}$. Then, even if we have not pinned down a symbol for (u,v) or (w,v), we know that for any legitimate codeword $c \in \text{List}_{\mathcal{C}}(z)$, assigning a symbol for one of these edges implies an assignment for the other. In this case, we say that $(u,v) \sim (w,v)$. Because the lists L_v are actually affine subspaces, there are not many equivalence classes at each vertex (and in particular substantially fewer equivalence classes than in the approach used in [27]).

For intuition, we first describe an algorithm that is slower than our final algorithm. The main idea is to choose $s = \text{poly}(2^r, 1/\varepsilon, 1/\delta)$ large equivalence classes and generate a list of all 2^s possible labelings for these equivalence classes. For each such labeling, we now hope to uniquely fill in the rest of the codeword, to arrive at a list of size 2^s. One might hope that labeling these s large equivalence classes would leave a fraction of unlabeled symbols less than the designed distance of $\mathcal{C}$, allowing us to immediately use the known linear-time erasure unique decoding algorithm for the expander code. Unfortunately, this is not in general the case. However, we show that there are many vertices v for which the number of unlabeled edges incident to v is at most $\delta(\mathcal{C}_0)d$. Thus, we may run the unique decoder for $\mathcal{C}_0$ (in time $O(d^3)$) at each such vertex to generate yet more labels. It turns out that at

this point, there *are* enough labels to run $\mathcal{C}$'s unique decoding algorithm and finish off the labeling.

Naively, the algorithm described above runs in time at least 2^s, since we must loop over all 2^s possibilities. This is exponential in ε and δ and doubly-exponential in r. The idea behind our final algorithm LISTDECODE is to take advantage of the linear structure of the lists L_v to find a short description of all of the legitimate labelings. We will do this in time $n \cdot \mathrm{poly}(d2^r/\varepsilon\delta)$ by leveraging the sparsity of $\mathcal{C}$'s parity-check matrix.

C. Related Work

The work that is perhaps the most related to ours is [27], which seeks to *list-recover* expander codes in the presence of erasures in linear time.[4] List-recovery is a variant of list-decoding which applies to codes over a large alphabet Σ: instead of receiving as input a vector $z \in \{0,1\}^N$, the decoder receives a collection of lists, $S_1, \ldots, S_N \subseteq \Sigma$, and the goal is to return all codewords $c \in \Sigma^N$ such that $c_i \in S_i$ for all i. In the setting of erasures, some lists have size $|\Sigma|$, in which case we may as well replace the whole list by a $\perp$ symbol. List decoding from erasures is a special case of list-recovery with erasures, where the S_i that are not $\perp$ have size one. However, existing list-recovery algorithms will not immediately work in our setting, as we consider binary codes: list-recovery is only possible for codes with large alphabets.

Our first observation is that the approach of [27] for erasure list-recovery can be used to obtain an algorithm for erasure list-decoding in linear time, even for binary codes. As described above, our first step is to erasure list-decode $\mathcal{C}_0$ at each vertex, leaving us with lists L_v that need to be "stitched together." The approach of [27] does precisely this, although in their context the lists that they are stitching together come from list-recovering the inner code.

However, the results of [27] about stitching together lists do not immediately yield anything meaningful for erasure list-decoding. More precisely, those results imply that an expander code $\mathcal{C}(G, \mathcal{C}_0)$ formed from a graph G with expansion λ and an inner code $\mathcal{C}_0$ with distance δ and r'th generalized distance δ_r is list-decodable from up to a $\delta\delta_r\left(\frac{\delta-\lambda/d}{6}\right)$ fraction of erasures in time $N \cdot \exp(\exp(\exp(r)))$. In particular, the fraction of erasures that those results tolerate is smaller than the distance of the expander code, yielding only trivial results in this setting.

Thus, while we use the same ideas as [27], our analysis is different and significantly tighter. This allows us to obtain a meaningful result in our setting, corresponding to the slower algorithm described above. Moreover, as described above, we are able to take advantage of the additional linear structure in our setting to improve the dependence on r in the running time.

To the best of our knowledge, there is no algorithmic work on list-decoding expander codes from errors (rather than erasures) in linear time with good parameters. We note that [31] recently showed that there are expander codes which are *combinatorially* near-optimally list-decodable from errors, but this work is non-constructive and does not provide efficient list decoding algorithms.

Finally, we discuss other work on erasure list-decoding. It is known that, non-constructively, there are erasure-list-decodable codes of rate $\Omega(\varepsilon)$ which can list-decode up to a $1-\varepsilon$ fraction of erasures, with list sizes $O(\log(1/\varepsilon))$ [21]. However, this proof is non-constructive and does not provide efficient algorithms, and it has been a major open question to achieve these results efficiently. Recent progress has been made by [28], who provided a construction (although no decoding algorithm) with parameters close to this for ε which is polynomially small in n.

Our work is somewhat orthogonal to this line of work on erasure list-decoding for several reasons. First, that line of work is mostly concerned with low-rate codes that are list-decodable from a large fraction of erasures (approaching 1), while expander codes tend to perform best at high rates. Second, we are less concerned with the trade-off between rate and erasure tolerance and more concerned with efficiently erasure-list-decoding an arbitrary expander code as far beyond its (designed) distance as possible. Finally, much of the line of work described above has focused on getting the list size down to $O(\log(1/\varepsilon))$, which is known to be impossible for linear codes, where the best list size possible is $\Omega(1/\varepsilon)$ [21]. Since the expander codes we consider are linear, we do not focus on that goal in our work.

II. PROOF OF THEOREM 2

Our main algorithm, LISTDECODE, is given in Figure 1. We describe each of the steps below. Many details are omitted due to space constraints, and can be found in the full version of the paper. In what follows, suppose that $z \in (\mathbb{F}_2 \cup \{\perp\})^E$ is a received word with at most $(1-\varepsilon)\delta\delta_r dn$ symbols that are $\perp$, and let $\mathcal{L} = \mathrm{List}_{\mathcal{C}}(z)$ be the set of codewords of $\mathcal{C}$ that are consistent with z.

The first step is to list decode all inner codes with not too many erasures. Let $B \subseteq L \cup R$ be the set of *bad* vertices v so that z has more than $\delta_r d$ erasures incident to v:

$$B = \left\{ v \in L \cup R : \, z_{(v,u)} = \perp \text{ for more than } \delta_r d \text{ vertices } u \right\}. \tag{1}$$

The first step of the algorithm is to list-decode $\mathcal{C}_0$ at every vertex $v \notin B$. For all such v, let

$$L_v := \mathrm{List}_{\mathcal{C}_0}\left((z_{(v,\Gamma_1(v))}, z_{(v,\Gamma_2(v))}, \ldots, z_{(v,\Gamma_d(v))}) \right). \tag{2}$$

Next, we shall use the following notion of *local equivalence relation* to assign labels to many of the edges. To define this notion, note that since $\mathcal{C}_0$ has r'th generalized distance δ_r, for any $v \notin B$, L_v is an affine subspace of $\mathbb{F}_2^d$ of dimension $r_v \leq r-1$. Let $G_v \in \mathbb{F}_2^{d \times r_v}$ and $b_v \in \mathbb{F}_2^d$ be such that $L_v = \{G_v x + b_v : x \in \mathbb{F}_2^{r_v}\}$.

Definition 5 (Local equivalence relation): Suppose that $v \notin B$. For $(u,v),(w,v) \in E$, say that $(u,v) \sim_v (w,v)$ if the row

[4] We note that other works, such as [24], have also had this goal, but to the best of our knowledge [27] obtains the best known results, so we focus on that work here.

Algorithm: LISTDECODE
Inputs: A description of $G = (L \cup R, E)$ and $\mathcal{C}_0 \subseteq \mathbb{F}_2^d$, and $z \in (\mathbb{F}_2 \cup \{\perp\})^E$.
Output: A matrix $L \in \mathbb{F}_2^{nd \times a}$ and a vector $\ell \in \mathbb{F}_2^{nd}$ so that

$$\mathrm{List}_{\mathcal{C}}(z) = \{Lx + \ell \,:\, x \in \mathbb{F}_2^a\}$$

for some integer a (which does not depend on n), or else $\perp$ if $\mathrm{List}_{\mathcal{C}}(z)$ is empty.

1) Let $B \subseteq L \cup R$ be as in (1). For each $v \notin B$, run $\mathcal{C}_0$'s erasure list-decoding algorithm to obtain the lists L_v as in (2). For each v, this entails finding the kernel of a sub-matrix of G_v, which can be done in time $O(d^3)$. Thus, the time for this step is $n \cdot \mathrm{poly}(d)$.
2) Find the set E' as described in the prose. Find the partition of E' into s global equivalence classes. This can be done in time $O(nd)$ using Breadth-First-Search.
3) Run FINDLIST, as in Lemma 9, to obtain $A, b, \hat{A}, \hat{b}$. (If FINDLIST returns $\perp$, then return $\perp$).
4) Compute $L = A\hat{A}$ and $\ell = A\hat{b} + b$, and return L, ℓ.

Fig. 1: LISTDECODE: Returns a description of $\mathrm{List}_{\mathcal{C}}(z)$ in time $n \cdot \mathrm{poly}(d, 2^r, 1/\delta, 1/\varepsilon)$.

of G_v corresponding to (u, v) is the same as the row of G_v corresponding to (w, v).
Observe that for $v \notin B$, If $(u, v) \sim_v (w, v)$, then for any $c \in \mathcal{L}$, $c_{(u,v)}$ is determined by $c_{(w,v)}$.

The next step is to assign labels to large *global* equivalence classes, defined below. For this, we first define a new edge set $E' \subseteq E$. We initialize this set to be the set of all edges that do not touch B. Then we repeatedly remove all local equivalence classes of size at most $\frac{\varepsilon^2\delta^2}{2^{r+3}} \cdot d$. We call the resulting set E'.

Definition 6 (Global equivalence relation): Suppose that $e, e' \in E'$. We say that $e \sim e'$ if there is a path $e = e_1, e_2, \ldots, e_t = e'$ so that $e_1, e_2, \ldots, e_t \in E'$, and for any pair of adjacent edges $e_i = (u, v)$, $e_{i+1} = (v, w)$ on the path it holds that $(u, v) \sim_v (v, w)$.

The following lemma (proved in the full version) shows that E' is partitioned into a small number of large global equivalence classes.

Lemma 7: Any global equivalence class in E' has size at least $\frac{\varepsilon^4\delta^4}{2^{2r+7}}dn$. In particular, E' is partitioned into at most $s := \frac{2^{2r+7}}{\varepsilon^4\delta^4}$ different equivalence classes.

Consequently, one can assign labels to all edges in E' by iterating over all possible assignments for a small number of representatives from these classes. As observed above, choosing a symbol on an edge determines all the symbols in that edge's equivalence class. Thus, we will exhaust over all choices of symbols for the equivalence classes in E'; this leads to 2^s possibilities. We show that for each way of choosing symbols on the edges in E', there is a unique way to complete the codeword of $\mathcal{C}$. To this end, define

$$B' = \{v \in L \cup R \,:\, (v, u) \notin E' \text{ for more than } \delta d \text{ vertices } u\}. \tag{3}$$

The next lemma (proved in the full version) bounds the size of B', and the number of edges in $E(B')$.

Lemma 8: The following hold:

1) $|B' \cap L|, |B' \cap R| \leq \left(1 - \frac{\varepsilon}{2}\right)\delta n$.
2) $|E(B')| \leq \left(1 - \frac{\varepsilon}{4}\right)\left(\delta - \frac{\lambda}{d}\right)\delta nd$.

Observe that for any vertex $v \notin B'$, the choices for symbols on E' uniquely determine the codeword of $\mathcal{C}_0$ that belongs at the vertex v. This is because $\mathcal{C}_0$ has distance δ, and at least $(1-\delta)d$ edges incident to v have been labeled. Note that since $\mathcal{C}_0$ is a linear code of length d, this unique codeword can be found in time $O(d^3)$ by solving a system of linear equations. Once this is done, the only edges that do not have labels are those in $E(B')$. By Item (2) of Lemma 8, there are at most $\left(1 - \frac{\varepsilon}{4}\right)\left(\delta - \frac{\lambda}{d}\right)\delta nd$ such edges. Finally, these edges can be recovered using global unique decoding in time $n \cdot \mathrm{poly}(d)/\varepsilon$.

This approach gives rise to the slow algorithm described above, where we exhaust over all 2^s possibilities for the equivalence classes in E', and then for each choice, append the resulting unique codeword to our list. However, we can do better. Instead of exhausting over all possible ways to assign values to the equivalence classes in E', we will set up and solve a linear system to find a description of the ways to assign these values that will lead to legitimate codewords. In the full version we prove the following.

Lemma 9: There is an algorithm FINDLIST which, given the state of LISTDECODE (Fig. 1) at the end of Step 2, runs in time $n \cdot \mathrm{poly}(d, s)$ and returns $A \in \mathbb{F}_2^{nd \times s}$, $b \in \mathbb{F}_2^{nd}$, $\hat{A} \in \mathbb{F}_2^{s \times a}$, and $\hat{b} \in \mathbb{F}_2^s$ so that

$$\mathrm{List}_{\mathcal{C}}(z) = \left\{Ax + b \,:\, x = \hat{A}\hat{x} + \hat{b} \text{ for some } \hat{x} \in \mathbb{F}_2^a\right\},$$

where $a := \dim(\mathrm{List}_{\mathcal{C}}(z))$ satisfies $a \leq s$.

This lemma explains the third and fourth steps of LISTDECODE, and proves Theorem 2.

ACKNOWLEDGEMENTS

Most of this work was done while the authors were participating in the Summer Cluster on Error-correcting Codes and High-dimensional Expansion at the Simons Institute for the Theory of Computing at UC Berkeley. NR is supported in part by BSF grant 2017732. MW is supported in part by NSF CAREER award CCF-1844628 and by NSF-BSF award CCF-1814629, and by a Sloan Research Fellowship.

REFERENCES

[1] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometry codes," *IEEE Trans. Information Theory*, vol. 45, no. 6, pp. 1757–1767, 1999. [Online]. Available: http://dx.doi.org/10.1109/18.782097

[2] F. Parvaresh and A. Vardy, "Correcting errors beyond the Guruswami-Sudan radius in polynomial time," in *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*. Washington, DC, USA: IEEE, Oct. 2005, pp. 285–294. [Online]. Available: http://dx.doi.org/10.1109/sfcs.2005.29

[3] V. Guruswami and A. Rudra, "Achieving list decoding capacity using folded Reed-Solomon codes," in *Allerton '06*, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.132.6808

[4] V. Guruswami and C. Wang, "Deletion codes in the high-noise and high-rate regimes," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 1961–1970, 2017.

[5] V. Guruswami and C. Xing, "Folded codes from function field towers and improved optimal rate list decoding," in *Proceedings of the 44th annual ACM symposium on Theory of computing (STOC)*. ACM, 2012, pp. 339–350.

[6] ——, "List decoding Reed-Solomon, algebraic-geometric, and Gabidulin subcodes up to the Singleton bound," in *Proceedings of the 45th annual ACM symposium on Theory of Computing (STOC)*. ACM, 2013, pp. 843–852.

[7] S. Kopparty, "List-decoding multiplicity codes," *Theory of Computing*, vol. 11, no. 5, pp. 149–182, 2015.

[8] V. Guruswami and S. Kopparty, "Explicit subspace designs," *Combinatorica*, vol. 36, no. 2, pp. 161–185, 2016. [Online]. Available: http://dx.doi.org/10.1007/s00493-014-3169-1

[9] B. Hemenway, N. Ron-Zewi, and M. Wootters, "Local list recovery of high-rate tensor codes and applications," *SIAM Journal on Computing*, no. 0, pp. FOCS17–157, 2019.

[10] S. Kopparty, N. Ron-Zewi, S. Saraf, and M. Wootters, "Improved decoding of folded Reed-Solomon and multiplicity codes," in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018, pp. 212–223.

[11] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996.

[12] G. Zémor, "On expander codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 835–837, Feb. 2001. [Online]. Available: http://dx.doi.org/10.1109/18.910593

[13] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

[14] V. Skachek and R. M. Roth, "Generalized minimum distance iterative decoding of expander codes," in *Proceedings 2003 IEEE Information Theory Workshop (Cat. No. 03EX674)*. IEEE, 2003, pp. 245–248.

[15] A. Ashikhmin and V. Skachek, "Decoding of expander codes at rates close to capacity," in *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*. IEEE, 2005, pp. 317–321. [Online]. Available: http://dx.doi.org/10.1109/isit.2005.1523346

[22] N. Ron-Zewi, M. Wootters, and G. Zémor, *full version of the present paper,* available at https://arxiv.org/abs/2002.08579

[16] A. Barg and G. Zémor, "Error exponents of expander codes," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1725–1729, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1109/tit.2002.1003853

[17] ——, "Concatenated codes: serial and parallel," *IEEE Transactions on Information Theory*, vol. 51, no. 5, pp. 1625–1634, May 2005. [Online]. Available: http://dx.doi.org/10.1109/tit.2005.846392

[18] ——, "Distance properties of expander codes," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 78–90, Jan. 2006. [Online]. Available: http://dx.doi.org/10.1109/tit.2005.860415

[19] R. M. Roth and V. Skachek, "Improved nearly-MDS expander codes," *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3650–3661, 2006.

[20] B. Hemenway, R. Ostrovsky, and M. Wootters, "Local correctability of expander codes," *Inf. Comput*, vol. 243, pp. 178–190, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.ic.2014.12.013

[21] V. Guruswami, "List decoding from erasures: Bounds and code constructions," *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 2826–2833, 2003.

[23] V. Guruswami and P. Indyk, "Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets," in *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '02. New York, NY, USA: ACM, 2002, pp. 812–821. [Online]. Available: http://dx.doi.org/10.1145/509907.510023

[24] ——, "Linear-Time List Decoding in Error-Free Settings," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, Eds. Springer Berlin Heidelberg, 2004, vol. 3142, pp. 695–707. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-27836-8_59

[25] P. Gaborit and O. Ruatta, "Efficient erasure list-decoding of Reed-Muller codes," in *2006 IEEE International Symposium on Information Theory*. IEEE, 2006, pp. 148–152.

[26] Y. Ding, L. Jin, and C. Xing, "Erasure list-decodable codes from random and algebraic geometry codes," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3889–3894, 2014.

[27] B. Hemenway and M. Wootters, "Linear-time list recovery of high-rate expander codes," *Information and Computation*, vol. 261, pp. 202–218, 2018.

[28] A. Ben-Aroya, D. Doron, and A. Ta-Shma, "Near-optimal erasure list-decodable codes," Electronic Colloquium on Computational Complexity, Tech. Rep. TR18-065, 2018.

[29] V. K.-W. Wei and K. Yang, "On the generalized Hamming weights of product codes," *IEEE Trans. Information Theory*, vol. 39, no. 5, pp. 1709–1713, 1993.

[30] H. G. Schaathun, "The weight hierarchy of product codes," *IEEE Trans. Information Theory*, vol. 46, no. 7, pp. 2648–2651, 2000.

[31] J. Mosheiff, N. Resch, N. Ron-Zewi, S. Silas, and M. Wootters, "LDPC codes achieve list decoding capacity," *arXiv preprint arXiv:1909.06430*, 2019.

[32] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bulletin of AMS*, vol. 43, no. 4, pp. 439–561, 2006.