Linear-Time List Recovery of High-Rate Expander Codes

Brett Hemenway[1(✉)] and Mary Wootters[2]

[1] University of Pennsylvania, Philadelphia, USA
fbrett@cis.upenn.edu
[2] Carnegie Mellon University, Pittsburgh, USA
marykw@cs.cmu.edu

Abstract. We show that expander codes, when properly instantiated, are high-rate list recoverable codes with linear-time list recovery algorithms. List recoverable codes have been useful recently in constructing efficiently list-decodable codes, as well as explicit constructions of matrices for compressive sensing and group testing. Previous list recoverable codes with linear-time decoding algorithms have all had rate at most 1/2; in contrast, our codes can have rate $1 - \varepsilon$ for any $\varepsilon > 0$. We can plug our high-rate codes into a framework of Alon and Luby (1996) and Meir (2014) to obtain linear-time list recoverable codes of arbitrary rates R, which approach the optimal trade-off between the number of non-trivial lists provided and the rate of the code.

While list-recovery is interesting on its own, our primary motivation is applications to list-decoding. A slight strengthening of our result would imply linear-time and optimally list-decodable codes for all rates. Thus, our result is a step in the direction of solving this important problem.

1 Introduction

In the theory of error correcting codes, one seeks a code $\mathcal{C} \subset \mathbb{F}^n$ so that it is possible to recover any *codeword* $c \in \mathcal{C}$ given a corrupted version of that codeword. The most standard model of corruption is from errors: some constant fraction of the symbols of a codeword might be adversarially changed. Another model of corruption is that there is some uncertainty: in each position $i \in [n]$, there is some small list $S_i \subset \mathbb{F}$ of possible symbols. In this model of corruption, we cannot hope to recover c exactly; indeed, suppose that $S_i = \{c_i, c'_i\}$ for some codewords $c, c' \in \mathcal{C}$. However, we can hope to recover a short list of codewords that contains c. Such a guarantee is called *list recoverability.*

While this model is interesting on its own—there are several settings in which this sort of uncertainty may arise—one of our main motivations for studying list-recovery is *list-decoding.* We elaborate on this more in Section 1.1 below.

We study the list recoverability of *expander codes.* These codes—introduced by Sipser and Spielman in [29]—are formed from an expander graph and an

M. Wootters–Research funded by NSF MSPRF grant DMS-1400558.

M.M. Halldórsson et al. (Eds.): ICALP 2015, Part I, LNCS 9134, pp. 701–712, 2015.
DOI: 10.1007/978-3-662-47672-7_57

inner code $\mathcal{C}_0$. One way to think about expander codes is that they preserve some property of $\mathcal{C}_0$, but have some additional useful structure. For example, [29] showed that if $\mathcal{C}_0$ has good distance, then so does the the expander code; the additional structure of the expander allows for a linear-time decoding algorithm. In [20], it was shown that if $\mathcal{C}_0$ has some good (but not great) locality properties, then the larger expander code is a good locally correctable code. In this work, we extend this list of useful properties to include list recoverability. We show that if $\mathcal{C}_0$ is a list recoverable code, then the resulting expander code is again list recoverable, but with a linear-time list recovery algorithm.

1.1 List Recovery

List recoverable codes were first studied in the context of list-decoding and soft-decoding: a list recovery algorithm is at the heart of the celebrated Guruswami-Sudan list-decoder for Reed-Solomon codes [17] and for related codes [16]. Guruswami and Indyk showed how to use list recoverable codes to obtain good list- and uniquely-decodable codes [12–14]. More recently, list recoverable codes have been studied as interesting objects in their own right, and have found several algorithmic applications, in areas such as compressed sensing and group testing [7,23,28].

We consider list recovery from erasures, which was also studied in [8,14]. That is, some fraction of symbols may have no information; equivalently, $S_i = \mathbb{F}$ for a constant fraction of $i \in [n]$. Another, stronger guarantee is list recovery from *errors*. That is, $c_i \notin S_i$ for a constant fraction of $i \in [n]$. We do not consider this stronger guarantee here, and it is an interesting question to extend our results for erasures to errors. It should be noted that the problem of list recovery is interesting even when there are neither errors nor erasures. In that case, the problem is: given $S_i \subset \mathbb{F}$, find all the codewords $c \in \mathcal{C}$ so that $c_i \in S_i$ for all i.

There are two parameters of interest. First, the rate $R := \log_q(|\mathcal{C}|)/n$ of the code: ideally, we would like the rate to be close to 1. Second, the efficiency of the recovery algorithm: ideally, we would be able to perform list-recovery in time linear in n. We survey the relevant results on list recoverable codes in Figure 1. While there are several known constructions of list recoverable codes with high rate, and there are several known constructions of list recoverable codes with linear-time decoders, there are no known prior constructions of codes which achieve both at once.

In this work, we obtain the best of both worlds, and give constructions of high-rate, linear-time list recoverable codes. Additionally, our codes have constant (independent of n) list size and alphabet size. As mentioned above, our codes are actually expander codes—in particular, they retain the many nice properties of expander codes: they are explicit linear codes which are efficiently (uniquely) decodable from a constant fraction of errors.

We can use these codes, along with a construction of Alon and Luby [1], recently highlighted by Meir [26], to obtain linear-time list recoverable codes of any rate R, which obtain the optimal trade-off between the fraction $1 - \alpha$ of erasures and the rate R. More precisely, for any $R \in [\mathbf{?}\ \mathbf{?}\ ]$, $\ell \in \mathbb{N}$, and $\eta > 0$,

there is some $L = L(\eta, \ell)$ so that we can construct rate R codes which are $(R + \eta, \ell, L)$-list recoverable in linear time. The fact that our codes from the previous paragraph have rate approaching 1 is necessary for this construction. To the best of our knowledge, linear-time list-decodable codes obtaining this trade-off were also not known.

It is worth noting that if our construction worked for list recovery from *errors*, rather than erasures, then the reduction above would obtain linear-time list decodable codes, of rate R and tolerating $1 - R - \eta$ errors. (In fact, it would yield codes that are list-recoverable from errors, which is a strictly stronger notion). So far, all efficiently list-decodable codes in this regime have polynomial-time decoding algorithms. In this sense, our work is a step in the direction of linear-time optimal list decoding, which is an important open problem in coding theory.[1]

1.2 Expander Codes

Our list recoverable codes are actually properly instantiated *expander codes.* Expander codes are formed from a d-regular expander graph, and an *inner code* $\mathcal{C}_0$ of length d, and are notable for their extremely fast decoding algorithms. We give the details of the construction below in Section 2. The idea of using a graph to create an error correcting code was first used by Gallager [6], and the addition of an inner code was suggested by Tanner [30]. Sipser and Spielman introduced the use of an expander graph in [29]. There have been several improvements over the years by Barg and Zemor [2–4,31].

Recently, Hemenway, Ostrovsky and Wootters [20] showed that expander codes can also be *locally corrected,* matching the best-known constructions in the high-rate, high-query regime for locally-correctable codes. That work showed that as long as the inner code exhibits suitable locality, then the overall expander code does as well. This raised a question: what other properties of the inner code does an expander code preserve? In this work, we show that as long as the inner code is list recoverable (even without an efficient algorithm), then the expander code itself is list recoverable, but with an extremely fast decoding algorithm.

It should be noted that the works of Guruswami and Indyk cited above on linear-time list recovery are also based on expander graphs. However, that construction is different from the expander codes of Sipser and Spielman. In particular, it does not seem that the Guruswami-Indyk construction can achieve a high rate while maintaining list recoverability.

[1] In fact, adapting our construction to handle errors, even if we allow polynomial-time decoding, is interesting. First, it would give a new family of efficiently-decodable, optimally list-decodable codes, very different from the existing algebraic constructions. Secondly, there are no known uniformly constructive explicit codes (that is, constructible in time $\text{poly}(n) \cdot C_\eta$) with both constant list-size and constant alphabet size—adapting our construction to handle errors, even with polynomial-time recovery, could resolve this.

1.3 Our Contributions

We summarize our contributions below:

1. **The first construction of linear-time list-recoverable codes with rate approaching 1.** As shown in Figure 1, existing constructions have either low rate or substantially super-linear recovery time. The fact that our codes have rate approaching 1 allows us to plug them into a construction of [26], to achieve the next bullet point:
2. **The first construction of linear-time list-recoverable codes with optimal rate/erasure trade-off.** We will show in Section 3.2 that our high-rate codes can be used to construct list-recoverable codes of arbitrary rates R, where we are given information about only an $R + \varepsilon$ fraction of the symbols. As shown in Figure 1, existing constructions which achieve this trade-off have substantially super-linear recovery time.
3. **A step towards linear-time, optimally list decodable codes.** Our results above are for list-recovery from *erasures*. While this has been studied before [14], it is a weaker model than a standard model which considers *errors*. As mentioned above, a solution in this more difficult model would lead to algorithmic improvements in list decoding (as well as potentially in compressed sensing, group testing, and related areas). It is our hope that understanding the erasure model will lead to a better understanding of the error model, and that our results will lead to improved list decodable codes.
4. **New tricks for expander codes.** One take-away of our work is that expander codes are extremely flexible. This gives a third example (after unique- and local- decoding) of the expander-code construction taking an inner code with some property and making that property efficiently exploitable. We think that this take-away is an important observation, worthy of its own bullet point. It is a very interesting question what other properties this may work for.

2 Definitions and Notation

An error correcting code is (α, ℓ, L) *list recoverable* (from errors) if given lists of ℓ possible symbols at every index, there are at most L codewords whose symbols lie in a α fraction of the lists. We will use a slightly different definition of list recoverability, matching the definition of [14]; to distinguish it from list recovery from errors, we will call it list recoverability from *erasures*.

Definition 1 (List recoverability from erasures). *An error correcting code $\mathcal{C} \subset \mathbb{F}_q^n$ is (α, ℓ, L)-list recoverable from erasures if the following holds. Fix any sets $S_1, \ldots, S_n$ with $S_i \subset \mathbb{F}_q$, so that $|S_i| \leq \ell$ for at least αn of the i's and $S_i = \mathbb{F}_q$ for all remaining i. Then there are most L codewords $c \in \mathcal{C}$ so that $c \in S_1 \times S_2 \times \cdots \times S_n$.*

Source	Rate	List size L	Alphabet size	Agreement α	Recovery time	Explicit Linear
Random code	$1-\gamma$	$O(\ell/\gamma)$	$\ell^{O(1/\gamma)}$	$1-O(\gamma)$		
Random pseudolinear code [11]	$1-\gamma$	$O\left(\frac{\ell \log(\ell)}{\gamma^2}\right)$	$\ell^{O(1/\gamma)}$	$1-O(\gamma)$		
Random linear code [9]	$1-\gamma$	$\ell^{O(\ell/\gamma^2)}$	$\ell^{O(1/\gamma)}$	$1-O(\gamma)$		L
Folded Reed-Solomon codes [16]	$1-\gamma$	$n^{O(\log(\ell)/\gamma)}$	$n^{O(\log(\ell)/\gamma^2)}$	$1-O(\gamma)$	$n^{O(\log(\ell)/\gamma^2)}$	EL
Folded RS subcodes: evaluation points in an explicit subspace-evasive set [5]	$1-\gamma$	$(1/\gamma)^{O(\ell/\gamma)}$	$n^{O(\ell/\gamma^2)}$	$1-O(\gamma)$	$n^{O(\ell/\gamma^2)}$	E
Folded RS subcodes: evaluation points in a non-explicit subspace-evasive set [10]	$1-\gamma$	$O\left(\frac{\ell}{\gamma^2}\right)$	$n^{O(\ell/\gamma^2)}$	$1-O(\gamma)$	$n^{O(\ell/\gamma^2)}$	
(Folded) AG subcode [18,19]	1 - γ	$O(\ell/\gamma)$	$\exp(\tilde{O}(\ell/\gamma^2))$	$1-O(\gamma)$	$C_{\ell,\gamma} n^{O(1)}$	
[13]	$2^{-2^{O(\ell)}}$	ℓ	$2^{2^{2^{O(\ell)}}}$	$1-2^{-2^{\ell^{O(1)}}}$	$O(n)$	E
[14]	$\ell^{-O(1)}$	ℓ	$2^{\ell^{O(1)}}$	.999 $(\star)$	$O(n)$	E
This work	$1-\gamma$	$\ell^{\gamma^{-4}\ell^{C\ell/\gamma^2}}$	$\ell^{O(1/\gamma)}$	$1-O(\gamma^3)$ $(\star)$	$O(n)$	EL

Fig. 1. Results on high-rate list recoverable codes and on linear-time decodable list recoverable codes. Above, n is the block length of the (α, ℓ, L)-list recoverable code, and $\gamma > 0$ is sufficiently small and independent of n. Agreement rates marked $(\star)$ are for erasures, and all others are from errors. An empty "recovery time" field means that there are no known efficient algorithms. We remark that [19], along with the explicit subspace designs of [15], also give explicit constructions of high-rate AG subcodes with polynomial time list-recovery and somewhat complicated parameters; the list-size L becomes super-constant.

The results listed above of [5,10,16,18,19] also apply for any rate R and agreement $R+\gamma$. In Section 3.2, we show how to achieve the same trade-off (for erasures) in linear time using our codes.

Our construction will be based on *expander graphs*. We say a d-regular graph H is a *spectral expander* with parameter λ, if λ is the second-largest eigenvalue

of the normalized adjacency matrix of H. Intuitively, the smaller λ is, the better connected H is—see [22] for a survey of expanders and their applications. We will take H to be a *Ramanujan graph,* that is, so that $\lambda \leq \frac{2\sqrt{d-1}}{d}$; explicit constructions of Ramanujan graphs are known [24,25,27] for arbitrarily large values of d. For a graph H with vertices $V(H)$ and edges $E(H)$, we use the following notation. For a set $S \subset V(H)$, we use $\Gamma(S)$ to denote the neighborhood

$$\Gamma(S) = \{v : \exists u \in S, (u,v) \in E(H)\}.$$

For a set of edges $F \subset E(H)$, we use $\Gamma_F(S)$ to denote the neighborhood restricted to F:

$$\Gamma_F(S) = \{v : \exists u \in S, (u,v) \in F\}.$$

Given a d-regular H and an inner code $\mathcal{C}_0$, we define the *Tanner code* $\mathcal{C}(H, \mathcal{C}_0)$ as follows.

Definition 2 (Tanner code [30]). *If H is a d-regular graph on n vertices and $\mathcal{C}_0$ is a linear code of block length d, then the* Tanner code *created from $\mathcal{C}_0$ and H is the linear code $\mathcal{C} \subset \mathbb{F}_q^{E(H)}$, where each edge H is assigned a symbol in $\mathbb{F}_q$ and the edges adjacent to each vertex form a codeword in $\mathcal{C}_0$.*

$$\mathcal{C} = \{c \in \mathbb{F}_q^{E(H)} : \forall v \in V(H), c|_{\Gamma(v)} \in \mathcal{C}_0\}$$

Because codewords in $\mathcal{C}_0$ are *ordered* collections of symbols whereas edges adjacent to a vertex in H may be unordered, creating a Tanner code requires choosing an ordering of the edges at each vertex of the graph. Although different orderings lead to different codes, our results (like all previous results on Tanner codes) work for all orderings. As our constructions work with any ordering of the edges adjacent to each vertex, we assume that some arbitrary ordering has been assigned, and do not discuss it further.

When the underlying graph H is an expander graph,[2] we call the resulting Tanner code an *expander code*. Sipser and Spielman showed that expander codes are efficiently uniquely decodable from about a δ_0^2 fraction of errors. We will only need unique decoding from erasures; the same bound of δ_0^2 obviously holds for erasures as well, but for completeness we state the following lemma, which we prove in the full version [21].

Lemma 1. *If $\mathcal{C}_0$ is a linear code of block length d that can recover from an $\delta_0 d$ number of erasures, and H is a d-regular expander with normalized second eigenvalue λ, then the expander code $\mathcal{C}$ can be recovered from a $\frac{\delta_0}{k}$ fraction of erasures in linear time whenever $\lambda < \delta_0 - \frac{2}{k}$.*

Throughout this work, $\mathcal{C}_0 \subset \mathbb{F}_q^d$ will be (α_0, ℓ, L)-list recoverable from erasures, and the distance of $\mathcal{C}_0$ is δ_0. We choose H to be a Ramanujan graph, and $\mathcal{C} = \mathcal{C}(H, \mathcal{C}_0)$ will be the expander code formed from H and $\mathcal{C}_0$.

[2] Although many expander codes rely on bipartite expander graphs (e.g. [31]), we find it notationally simpler to use the non-bipartite version.

3 Results and Constructions

In this section, we give an overview of our constructions and state our results. Our main result (Theorem 1) is that list recoverable inner codes imply list recoverable expander codes. We then instantiate this construction to obtain the high-rate list recoverable codes claimed in Figure 1. Next, in Theorem 3 we show how to combine our codes with a construction of Meir [26] to obtain linear-time list recoverable codes which approach the optimal trade-off between α and R.

3.1 High-Rate Linear-Time List Recoverable Codes

Our main theorem is that list recoverable codes imply list recoverable expander codes:

Theorem 1. *Suppose that $\mathcal{C}_0$ is (α_0, ℓ, L)-list recoverable from erasures, of rate R_0, length d, and distance δ_0, and suppose that H is a d-regular expander graph with normalized second eigenvalue λ, if*

$$\lambda < \frac{\delta_0^2}{12\ell^L}$$

Then the expander code $\mathcal{C}$ formed from $\mathcal{C}_0$ and H has rate at least $2R_0 - 1$ and is (α, ℓ, L')-list recoverable from erasures, where

$$L' \leq \exp_\ell\left(\frac{72\,\ell^{2L}}{\delta_0^2(\delta_0 - \lambda)^2}\right)$$

and α satisfies

$$1 - \alpha \geq (1 - \alpha_0)\left(\frac{\delta_0(\delta_0 - \lambda)}{6}\right).$$

Further, the running time of the list recovery algorithm is $O_{L,\ell,\delta_0,d}(n)$.

Above, the notation $\exp_\ell(\cdot)$ means $\ell^{(\cdot)}$. Before we discuss the proof of Theorem 1 and the recovery algorithm, we show how to instantiate these codes to give the parameters claimed in Figure 1.

We will use a random linear code as the inner code. A probabilistic argument shows that there exist inner codes with $R_0 = 1 - \gamma$, distance $\delta_0 = \gamma(1 + \mathcal{O}(\gamma))$ that are (α_0, ℓ, L)-list recoverable, over an alphabet of size $q^{O(1/\gamma)}$. (See the full version of this paper [21]). Plugging all this into Theorem 1, we get explicit codes of rate $1 - 2\gamma$ which are (α, ℓ, L')-list recoverable in linear time, for

$$L' = \exp_\ell\left(\gamma^{-4}\exp_\ell\left(\exp_\ell\left(C\ell/\gamma^2\right)\right)\right)$$

and for $\alpha = 1 - C'\gamma^3$ for some constants C, C'. This recovers the parameters claimed in Figure 1. Above, we can choose $d = O\left(\frac{\ell^{2L}}{\gamma^4}\right)$ so that the Ramanujan graph would have parameter λ obeying the conditions of Theorem 1. Thus, when ℓ, γ are constant, so is the degree d, and the running time of the recovery algorithm is linear in n, and thus in the block length nd of the expander code. By Lemma 1, because the distance of the inner code is $\delta_0 = \gamma(1 + O(\gamma))$, the distance of our construction is $\delta = \Omega(\gamma^2)$.

Remark 1. Both the alphabet size and the list size L' are constant, if ℓ and γ are constant. However, L' depends rather badly on ℓ, even compared to the other high-rate constructions in Figure 1. This is because the bound on random linear codes that we use for our inner code is likely not tight; it would be interesting to either improve this bound or to give an inner code with better list size L. The key restrictions for such an inner code are that (a) the rate of the code must be close to 1; (b) the list size L must be constant, and (c) the code must be linear. Notice that (b) and (c) prevent the use of either Folded Reed-Solomon codes or their restriction to a subspace evasive set, respectively.

3.2 List Recoverable Codes Approaching Capacity

We can use our list recoverable codes, along with a construction of Alon and Luby [1] (which has also been used for similar purposes by Guruswami and Indyk [12], and was recently used and highlighted by Meir [26]), to construct codes which approach the optimal trade-off between the rate R and the agreement α. To quantify this, we state the following analog of the list-decoding capacity theorem.

Theorem 2 (List recovery capacity theorem). *For every $R > 0$, and $L \geq \ell$, there is some code $\mathcal{C}$ of rate R over $\mathbb{F}_q$ which is $(R+\eta(\ell, L), \ell, L)$-list recoverable from erasures, for any*

$$\eta(\ell, L) \geq \frac{4\ell}{L} \qquad \text{and} \qquad q \geq \ell^{2/\eta}.$$

Further, for any constant $R > 0$, any integer ℓ, and any sufficiently small $\eta > 0$, any code of rate R which is $(R-\eta, \ell, L)$-list recoverable from erasures must have $L = q^{\Omega(n)}$.

The proof is a straightforward probabilistic argument and is given in the full version [21]. Although Theorem 2 ensures the existence of certain list-recoverable codes, the proof of Theorem 2 is probabilistic, and does not provide a means of efficiently identifying (or list recovering) these codes. Using the approach of [26] we can turn our construction of linear-time list recoverable codes into linear-time list recoverable codes approaching capacity.

Theorem 3. *For any $R > 0$, $\ell > 0$, and for all sufficiently small $\eta > 0$, there is some L, depending only on ℓ and η, and some constant d, depending only on η, so that whenever $q \geq \ell^{6/\eta}$ there is a family of (α, ℓ, L)-list recoverable codes $\mathcal{C} \subset \mathbb{F}_{q^d}^n$ with rate at least R, for $\alpha = R + \eta$. Further, these codes can be list-recovered in linear time.*

We follow the approach of [26], which adapts a construction of [1] to take advantage of high-rate codes with a desirable property. Informally, the takeaway of [26] is that, given a family of codes with any nice property and rate approaching 1, one can make a family of codes with the same nice property that acheives the Singleton bound. The proof of Theorem 3, as well as the construction, can be found in the full version [21].

4 Recovery Procedure and Proof of Theorem 1

In this section, we outline at a high level the ideas and techniques in our list recovery algorithm. A detailed description of the recovery algorithm and the proof of correctness can be found in the full version [21]. The complete list recovery algorithm is presented in the full version. The algorithm proceeds in three steps, which we describe below. Due to space constraints, we omit the details of these steps, which can be found in the full version of the paper [21].

1. First, we list recover locally at each vertex, using the list recoverability of the inner code.
   This step yields a list of L codewords at each vertex.
2. Next, we choose an edge, and one of the ℓ possible symbols on that edge. The crux of the decoding algorithm is identifying how this choice propagates through the graph.
   This propagation will cover a constant fraction of the edges in the graph. We repeat this propagation for each of the ℓ choices of symbol for the chosen edge. This yields a collection of ℓ possible partial codewords.
3. Step 2 yields partial assignments (that assign values to a constant fraction of the symbols in the expander code). To turn these partial assignments into full assignments, we repeat Step 2 a constant number of times until we have partial assignments that cover essentially the entire graph. We stop once we have covered enough edges, and we use the minimum distance of the expander code to uniquely fill in the unknown edges. Since each iteration of Step 2 yields ℓ possible partial assignments (all to the same set of edges), if we repeat Step 2 t times, we can stitch them together to obtain ℓ^t possible assignments.

The difficulty in analyzing this algorithm comes from determining how a choice of a symbol in Step 2 propagates through the graph. We sketch the intuition below; the formal discussion can be found in [21].

For simplicity, suppose there are no erasures—our final algorithm can recover from a constant fraction of erasures, but the intuition is cleaner if there are no erasures—and suppose that each edge of H holds a list of ℓ possible symbols. Suppose (v, u) is the edge chosen at Step 2. We might hope that a choice of a symbol on this edge (or even the choice of a codeword at vertex u) would determine the codeword at v. This is unfortunately not likely to be true because $L > \ell$: a choice of one of ℓ symbols on (v, u) is not sufficient to uniquely determine one of the L codewords on vertex v. Instead of analyzing propagation at a vertex level, we focus on propagation at the edge level.

To do this, we introduce the notion of equivalence classes of *edges*. Suppose that the neighbors of v are $u_1, \ldots, u_d$. There are L possible codewords at v, and there are ℓ possible choices of symbol at (v, u_i); thus there are at most ℓ^L possible maps from codeword at v to symbol at (v, u_i). If $d \gg \ell^L$, then by the pigeonhole principle some of these maps must be identical. We call edges (v, u_i) and (v, u_j) equivalent with respect to v if their maps are identical. In particular, this means

that a choice of symbol of (v, u_i) defines the choice of symbol on (v, u_j). Thus a choice of symbol on (v, u_1) (say), will determine symbols of (v, u_i) for all (v, u_i) in the same equivalence class as (v, u_1).

We can then repeat this logic at each of these vertices u_i: the choice of symbol on (u_i, v) will determine symbols on edges (u_i, w) that are equivalent to (u_i, v) with respect to u_i. In this way, the choice of a single symbol propagates through a large portion of the graph. We use the expansion of the graph to show that this propagation ends up covering a constant fraction of the graph. Thus, after making a constant number of choices (and using the distance of the expander code to take care of the small fraction of untouched edges), we will have recovered every assignment of symbols which is consistent with the given lists.

There are several details omitted from the sketch above. For example, we argued above that *some* equivalence classes are large. Of course, some may also be small. What if (v, u) belongs to a small equivalence class and our choice does not propagate? We show in the appendix that there is a large subgraph H' of H so that every equivalence class in H' is large. The full details, and a complete description of the recovery algorithm, can be found in the full version of the paper [21].

5 Conclusion and Open Questions

We have shown that expander codes, properly instantiated, are high-rate list recoverable codes with constant list size and constant alphabet size, which can be list recovered in linear time. To the best of our knowledge, no such construction was known. Our work leaves several open questions. Most notably, our algorithm can handle *erasures,* but it seems much more difficult to handle errors. As mentioned above, handling list recovery from errors would open the door for many of the applications of list recoverable codes, to list-decoding and other areas. Extending our results to errors with linear-time recovery would be most interesting, as it would immediately lead to optimal linear-time list-decodable codes. However, even polynomial-time recovery would be interesting: in addition to given a new, very different family of efficient locally-decodable codes, this could lead to explicit (uniformly constructive), efficiently list-decodable codes with constant list size and constant alphabet size, which is (to the best of our knowledge) currently an open problem. Second, the parameters of our construction could be improved: our choice of inner code (a random linear code), and its analysis, is clearly suboptimal. Our construction would have better performance with a better inner code. As mentioned in Remark 1, we would need a high-rate linear code which is list recoverable with constant list-size (the reason that this is not begging the question is that this inner code need not have a fast recovery algorithm). We are not aware of any such constructions.

Acknowledgments. We thank Venkat Guruswami for raising the question of obtaining high-rate linear-time list-recoverable codes, and for very helpful conversations. We also thank Or Meir for pointing out [26].

References

1. Alon, N., Luby, M.: A linear time erasure-resilient code with nearly optimal recovery. IEEE Transactions on Information Theory **42**(6), 1732–1736 (1996)
2. Barg, A., Zemor, G.: Error exponents of expander codes. IEEE Transactions on Information Theory **48**(6), 1725–1729 (2002)
3. Barg, A., Zemor, G.: Concatenated codes: serial and parallel. IEEE Transactions on Information Theory **51**(5), 1625–1634 (2005)
4. Barg, A., Zemor, G.: Distance properties of expander codes. IEEE Transactions on Information Theory **52**(1), 78–90 (2006)
5. Dvir, Z., Lovett, S.: Subspace evasive sets. In: Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC), pp. 351–358. ACM (2012)
6. Gallager, R.G.: Low Density Parity-Check Codes. Technical report. MIT (1963)
7. Gilbert, A.C., Ngo, H.Q., Porat, E., Rudra, A., Strauss, M.J.: ℓ_2/ℓ_2-foreach sparse recovery with low risk. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 461–472. Springer, Heidelberg (2013)
8. Guruswami, V.: List decoding from erasures: Bounds and code constructions. IEEE Transactions on Information Theory **49**(11), 2826–2833 (2003)
9. Guruswami, V.: List decoding of error-correcting codes. LNCS, vol. 3282. Springer, Heidelberg (2004)
10. Guruswami, V.: Linear-algebraic list decoding of folded reed-solomon codes. In: Proceedings of the 26th Annual Conference on Computational Complexity (CCC), pp. 77–85. IEEE (2011)
11. Guruswami, V., Indyk, P:. Expander-based constructions of efficiently decodable codes. In: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 658–667. IEEE (October 2001)
12. Guruswami, V., Indyk, P.: Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In: Proceedings of the 34th Annual ACM Aymposium on Theory of computing (STOC), pp. 812–821. ACM (2002)
13. Guruswami, V., Indyk, P.: Linear time encodable and list decodable codes. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC), pp. 126–135. ACM, New York (2003)
14. Guruswami, V., Indyk, P.: Linear-time list decoding in error-free settings. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 695–707. Springer, Heidelberg (2004)
15. Guruswami, V., Kopparty, S.: Explicit subspace designs. In: Proceedings of the 54th Annual IEEE Symposium on Foundations of Computing (FOCS), pp. 608–617. IEEE (2013)
16. Guruswami, V., Rudra, A.: Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. IEEE Transactions on Information Theory **54**(1), 135–150 (2008)
17. Guruswami, V., Sudan, M.: Improved decoding of Reed-Solomon and algebraic-geometry codes. IEEE Transactions on Information Theory 45(6) (1999)
18. Guruswami, V., Xing, C.: 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), pp. 339–350. ACM (2012)
19. Guruswami, V., Xing, C.: 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), pp. 843–852. ACM (2013)

20. Hemenway, B., Ostrovsky, R., Wootters, M.: Local correctability of expander codes. Information and Computation (2014)
21. Hemenway, B., Wootters, M.: Linear-time list recovery of high-rate expander codes. ArXiv preprint 1503.01955 (2015)
22. Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bulletin of the American Mathematical Society **43**(4), 439–561 (2006)
23. Indyk, P., Ngo, H.Q., Rudra, A.: Efficiently decodable non-adaptive group testing. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1126–1142. Society for Industrial and Applied Mathematics (2010)
24. Lubotzky, A., Phillips, R., Sarnak, P.: Ramanujan graphs. Combinatorica **8**(3), 261–277 (1988)
25. Margulis, G.A.: Explicit Group-Theoretical Constructions of Combinatorial Schemes and Their Application to the Design of Expanders and Concentrators. Probl. Peredachi Inf. **24**(1), 51–60 (1988)
26. Meir, O.: Locally correctable and testable codes approaching the singleton bound, ECCC Report TR14-107 (2014)
27. Morgenstern, M.: Existence and Explicit Constructions of q + 1 Regular Ramanujan Graphs for Every Prime Power q. Journal of Combinatorial Theory, Series B **62**(1), 44–62 (1994)
28. Ngo, H.Q., Porat, E., Rudra, A.: Efficiently decodable compressed sensing by list-recoverable codes and recursion. In: Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS), vol. 14, pp. 230–241 (2012)
29. Sipser, M., Spielman, D.A.: Expander codes. IEEE Transactions in Information Theory 42(6) (1996)
30. Tanner, R.: A recursive approach to low complexity codes. IEEE Transactions on Information Theory **27**(5), 533–547 (1981)
31. Zemor, G.: On expander codes. IEEE Transactions on Information Theory **47**(2), 835–837 (2001)