Local Correctability of Expander Codes

Brett Hemenway* Rafail Ostrovsky† Mary Wootters‡

January 8, 2015

Abstract

In this work, we present the first local-decoding algorithm for expander codes. This yields a new family of constant-rate codes that can recover from a constant fraction of errors in the codeword symbols, and where any symbol of the codeword can be recovered with high probability by reading N^ϵ symbols from the corrupted codeword, where N is the block-length of the code.

Expander codes, introduced by Sipser and Spielman, are formed from an expander graph $G = (V, E)$ of degree d, and an inner code of block-length d over an alphabet Σ. Each edge of the expander graph is associated with a symbol in Σ. A string in Σ^E will be a codeword if for each vertex in V, the symbols on the adjacent edges form a codeword in the inner code.

We show that if the inner code has a smooth reconstruction algorithm in the noiseless setting, then the corresponding expander code has an efficient local-correction algorithm in the noisy setting. Instantiating our construction with inner codes based on finite geometries, we obtain novel locally decodable codes with rate approaching one. This provides an alternative to the multiplicity codes of Kopparty, Saraf and Yekhanin (STOC '11) and the lifted codes of Guo, Kopparty and Sudan (ITCS '13).

1 Introduction

Expander codes, introduced in [32], are linear codes which are notable for their efficient decoding algorithms. In this paper, we show that when appropriately instantiated, expander codes are also *locally decodable,* and we give a sublinear time local-decoding algorithm.

In standard error correction, a sender encodes a message $x \in \{0,1\}^k$ as a codeword $c \in \{0,1\}^N$, and transmits it to a receiver across a noisy channel. The receiver's goal is to recover x from the corrupted codeword w. Decoding algorithms typically process all of w and in turn recover all of x. The goal of local decoding is to recover only a single bit of x, with the benefit of querying only a few bits of w. The number of bits of w needed to recover a single bit x is known as the *query complexity*, and is denoted q. The important trade-off in local decoding is between query complexity and the rate $r = k/N$ of the code. When q is constant or even logarithmic in k, the best known codes have rates which tend to zero as N grows. The first locally decodable codes to achieve sublinear locality and rate approaching one were the multiplicity codes of Kopparty, Saraf and Yekhanin [25]. Prior to this work, only two constructions of locally decodable codes were known with sublinear locality and rate approaching one [25, 20]. In this paper, we show that expander codes provide a third construction of efficiently locally decodable codes with rate approaching one.

*Department of Computer and Information Science, University of Pennsylvania, fbrett@cis.upenn.edu.

†Department of Computer Science and Department of Mathematics, UCLA, rafail@cs.ucla.edu. Research supported in part by NSF grants CNS-0830803; CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

‡Department of Computer Science, Carnegie Mellon University, marykw@cs.cmu.edu. Research supported in part by NSF grant CCF-1161233

1.1 Notation and preliminaries

Before we state our main results, we set notation and give a few definitions. We will construct linear codes $\mathcal{C}$ of length N and message length k, over an alphabet $\Sigma = \mathbb{F}$, for some finite field $\mathbb{F}$. That is, $\mathcal{C} \subset \mathbb{F}^N$ is a linear subspace of dimension k. The *rate* of $\mathcal{C}$ is the ratio $r = k/N$. We will also use expander graphs: we say a d-regular graph G is a *spectral expander* with parameter λ, if λ is the second-largest eigenvalue of the normalized adjacency matrix of G. Intuitively, the smaller λ is, the better connected G is—see [21] for a survey of expanders and their applications. For $n \in \mathbb{Z}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For $x, y \in \Sigma^N$, $\Delta(x, y)$ denotes relative Hamming distance, $x[i]$ denotes the i^{th} symbol of x, and $x|_S$ denotes x restricted to symbols indexed by $S \subset [N]$.

A code (along with an encoding algorithm) is *locally decodable* if there is an algorithm which can recover a symbol $x[i]$ of the message, making only a few queries to the received word.

Definition 1 (Locally Decodable Codes (LDCs))**.** *Let $\mathcal{C} \subset \Sigma^N$ be a code of size $|\Sigma|^k$, and let $E : \Sigma^k \to \Sigma^N$ be an encoding map. Then $(\mathcal{C}, E)$ is (q, ρ)-locally decodable with error probability η if there is a randomized algorithm R, so that for any $w \in \Sigma^N$ with $\Delta(w, E(x)) < \rho$, for each $i \in [k]$,*

$$\mathbb{P}\{R(w, i) = x[i]\} \geq 1 - \eta,$$

and further R accesses at most q symbols of w. Here, the probability is taken over the internal randomness of the decoding algorithm R.

In this work, we will actually construct *locally correctable codes,* which we will see below imply locally decodable codes.

Definition 2 (Locally Correctable Codes (LCCs))**.** *Let $\mathcal{C} \subset \Sigma^N$ be a code, and let $E : \Sigma^k \to \Sigma^N$ be an encoding map. Then $\mathcal{C}$ is (q, ρ)-locally correctable with error probability η if there is a randomized algorithm, R, so that for any $w \in \mathbb{F}^N$ with $\Delta(w, E(x)) < \rho$, for each $j \in [N]$,*

$$\mathbb{P}\{R(w, j) = w[j]\} \geq 1 - \eta,$$

and further R accesses at most q symbols of w. Here, the probability is taken over the internal randomness of the decoding algorithm R.

Thus the only difference between locally correctable codes and locally decodable codes is that locally correctable codes can recover symbols of the *codeword* while locally decodable codes recover symbols of the *message*.

When there is a constant $\rho > 0$ and a failure probability $\eta = o(1)$ so that $\mathcal{C}$ is (q, ρ)-locally correctable with error probability η, we will simply say that $\mathcal{C}$ is locally correctable with query complexity q (and similarly for locally decodable).

When $\mathcal{C}$ is a linear code, writing the generator matrix in systematic form gives an encoding function $E : \mathbb{F}^k \to \mathbb{F}^N$ so that for every $x \in \mathbb{F}^k$ and for all $i \in [k]$, $E(x)[i] = x[i]$. In particular, if $\mathcal{C}$ is a (q, ρ) linear LCC, then $(E, \mathcal{C})$ is a (q, ρ) LDC. Because of this connection, we will focus our attention on creating locally correctable linear codes.

Many LCCs work on the following principle: suppose, for each $i \in [N]$, there is a set of q query positions $Q(i)$, which are *smooth*—that is, each query is almost uniformly distributed within the codeword—and a method to determine $c[i]$ from $\{c[j] : j \in Q(i)\}$ for any uncorrupted codeword $c \in \mathcal{C}$. If q is constant, this *smooth local reconstruction algorithm* yields a local correction algorithm: with high probability none of the locations queried are corrupted. In particular, by a union bound, the smooth local reconstruction algorithm is a local correction algorithm that fails with probability at most $\rho \cdot q$. This argument is effective when $q = \mathcal{O}(1)$, however, when q is merely sublinear in N, as is the case for us, this reasoning fails. This paper demonstrates how to turn codes which only possess a local reconstruction procedure (in the noiseless setting) into LCCs with constant rate and sublinear query complexity.

Definition 3 (Smooth reconstruction)**.** *For a code $\mathcal{C} \subset \Sigma^N$, consider a pair of algorithms (Q, A), where Q is a randomized query algorithm with inputs in $[N]$ and outputs in 2^N, and $A : \Sigma^q \times [N] \to \Sigma$ is a deterministic reconstruction algorithm. We say that (Q, A) is a* s-smooth local reconstruction algorithm *with query complexity q if the following hold.*

1. *For each $i \in [N]$, the query set $Q(i)$ has $|Q(i)| \leq q$.*
2. *For each $i \in [N]$, there is some set $S \subset [N]$ of size s, so that each query in $Q(i)$ is uniformly distributed in S.*
3. *For all $i \in [N]$ and for all codewords $c \in \mathcal{C}$, $A(c|_{Q(i)}, i) = c[i]$.*

If $s = N$, then we say the reconstruction is perfectly smooth, since all symbols are equally likely to be queried. Notice that the queries need not be independent. The codes we consider in this work decode a symbol indexed by $x \in \mathbb{F}^m$ by querying random subspaces through x (but not x itself), and thus will have $s = N - 1$.

1.2 Related work

The first local-decoding procedure for an error-correcting code was the majority-logic decoder for Reed-Muller codes proposed by Reed [31]. Local-decoding procedures have found many applications in theoretical computer science including proof-checking [26, 4, 30], self-testing [10, 11, 17, 18] and fault-tolerant circuits [33]. While these applications implicitly used local-decoding procedures, the first explicit definition of locally decodable codes did not appear until later [24]. An excellent survey is available [38]. The study of locally decodable codes focuses on the trade-off between rate (the ratio of message length to codeword length) and query complexity (the number of queries made by the decoder). Research in this area is separated into two distinct areas: the first seeks to minimize the query complexity, while the second seeks to maximize the rate. In the low-query-complexity regime, Yekhanin was the first to exhibit codes with a constant number of queries and a subexponential rate [36]. Following Yekhanin's work, there has been significant progress in constructing locally decodable codes with constant query-complexity [37, 14, 13, 9, 23, 12, 8, 15]. On the other hand, in the high-rate regime, there has been less progress. In 2011, Kopparty, Saraf and Yekhanin introduced *multiplicity codes*, the first codes with a sublinear local-decoding algorithm [25] and rate approaching one. Like Reed-Muller codes, multiplicity codes treat the message as a multivariate polynomial, and create codewords by evaluating the polynomial at a sequence of points. Multiplicity codes are able to improve on the performance of Reed-Muller codes by also including evaluations of the partial derivatives of the message polynomial in the codeword. A separate line of work has developed high-rate locally decodable codes by "lifting" shorter codes [20]. The work of Guo, Kopparty and Sudan takes a short code $\mathcal{C}_0$ of length $|\mathbb{F}|^t$, and lifts it to a longer code $\mathcal{C}$, of length $|\mathbb{F}|^m$ for $m > t$ over $\mathbb{F}$, such that every restriction of a codeword in $\mathcal{C}$ to an affine subspace of dimension t yields a codeword in $\mathcal{C}_0$. The definition provides a natural local-correcting procedure for the outer code: to decode a symbol of the outer code, pick a random affine subspace of dimension t that contains the symbol, read the coordinates and decode the resulting codeword using the code $\mathcal{C}_0$. Guo, Kopparty and Sudan show how to lift explicit inner codes so that the outer code has constant rate and query complexity N^ε.

In this work, we show that *expander codes* can also give locally decodable codes with rate approaching one, and with query complexity N^ε. Expander codes, introduced by Sipser and Spielman [32], are formed by choosing a d-regular expander graph, G on n vertices, and a code $\mathcal{C}_0$ of length d (called the *inner code*), and defining the codeword to be all assignments of symbols to the edges of G so that for every vertex in G, its edges form a codeword in $\mathcal{C}_0$. The connection between error-correcting codes and graphs was first noticed by Gallager [16] who showed that a random bipartite graph induces a good error-correcting code. Gallager's construction was refined by Tanner [35], who suggested the use of an inner code. Sipser and Spielman [32] were the first to consider this type of code with an expander graph, and Spielman [34] showed that these *expander codes* could be encoded and decoded in linear time. Spielman's work provided the first family of error-correcting code with linear-time encoding and decoding procedures. The decoding procedure has since been improved by Barg and Zemor [39, 5, 6, 7].

1.3 Our approach and contributions

We show that certain expander codes can be efficiently locally decoded, and we instantiate our results to obtain novel families of (N^ε, ρ)-LCCs of rate $1-\alpha$, for any positive constants α, ε and some positive constant ρ. Our decoding algorithm runs in time linear in the number of queries, and hence sublinear in the length of the message. We provide a general method for turning codes with smooth local reconstruction algorithms into LCCs: our main result, Theorem 5, states that as long as the inner code $\mathcal{C}_0$ has rate at least $1/2$ and possesses a smooth local reconstruction algorithm, then the corresponding family of expander codes are constant rate LCCs. In Section 3, we give some examples of appropriate inner codes, leading to the parameters claimed above.

In addition to providing a sublinear time local decoding algorithm for an important family of codes, our constructions are only the third known example of LDCs with rate approaching one, after multiplicity codes [25] and lifted Reed-Solomon codes [20]. Our approach (and the resulting codes) are very different from earlier approaches. Both multiplicity codes and lifted Reed-Solomon codes use the same basic principle, also at work in Reed-Muller codes: in these schemes, for any two codewords c_1 and c_2 which differ at index i, the corresponding queries $c_1|_{Q(i)}$ and $c_2|_{Q(i)}$ differ in many places. Thus, if the queries are smooth, with high probability they will not have too many errors, and the correct symbol can be recovered. In contrast, our decoder works differently: while our queries are smooth, they will not have this distance property. In fact, changing a mere $\log(q)$ out of our q queries may change the correct answer. The trick is that these problematic error patterns must have a lot of structure, and we will show that they are unlikely to occur.

Finally, our results port a typical argument from the low-query regime to the high-rate regime. As mentioned above, when the query complexity q is constant, a smooth local reconstruction algorithm is sufficient for local correctability. However, this reasoning fails when q grows with N. In this paper, we show how to make this argument go through: via Theorem 5, any family of codes $\mathcal{C}_0$ with good rate and a smooth local decoder can be used to obtain a family of LCCs with similar parameters.

2 Local correctability of expander codes

In this section, we give an efficient local correction algorithm for expander codes with appropriate inner codes. We use a formulation of expander codes due to [39]. Let G be a d-regular expander graph on n vertices with expansion parameter λ. We will take G to be a *Ramanujan graph,* that is, so that $\lambda \leq \frac{2\sqrt{d-1}}{d}$; explicit constructions of Ramanujan graphs are known [27, 28, 29] for arbitrarily large values of d. Let H be the double cover of G. That is, H is a bipartite graph whose vertices $V(H)$ are two disjoint copies V_0 and V_1 of $V(G)$, and so that

$$E(H) = \{(u_0, v_1) \,:\, (u, v) \in E(G)\},$$

where u_i denotes the copy of u in V_i. Fix a linear inner code $\mathcal{C}_0$ over Σ of rate r_0 and relative distance δ_0. Let $N = nd$. For $v_i \in V(H)$, let $E(v_i)$ denote the edges attached to v. The expander code $\mathcal{C} \subset \Sigma^N$ of length N arising from G and $\mathcal{C}_0$ is given by

$$\mathcal{C} = \mathcal{C}_N(\mathcal{C}_0, G) = \left\{ x \in \Sigma^N \,:\, x|_{E(v_i)} \in \mathcal{C}_0 \text{ for all } v_i \in V(H) \right\} \tag{1}$$

The following theorem states that as long as the inner code $\mathcal{C}_0$ has good rate and distance, so does the resulting code $\mathcal{C}$.

Theorem 4 ([35, 32])**.** *The code $\mathcal{C}$ has rate $r \geq 2r_0 - 1$, and as long as $2\lambda \leq \delta_0$, the relative distance of $\mathcal{C}$ is at least $\delta_0^2/2$.*

Notice that when $r_0 < \frac{1}{2}$, Theorem 4 is meaningless. The rate in Theorem 4 comes from the fact that $\mathcal{C}_0$ has rate r_0, so each vertex induces $(1-r_0)d$ linear constraints, and there are n vertices, so the outer code has $nd(1-r_0)$ constraints. Since the outer code has length $N = nd/2$, its rate is at least $2r_0 - 1$. This naïve lower bound on the rate ignores the possibility that the constraints induced by the different vertices may not all be independent. It is an interesting question whether for certain inner codes, a more careful counting of

constraints could yield a better lower bound on the rate. The ability to use inner codes of rate less than $\frac{1}{2}$ would permit much more flexibility in the choice of inner code in our constructions.

The difficulty of a more sophisticated lower bound on the rate was noticed by Tanner, who pointed out that simply permuting the codewords associated with a given vertex could drastically alter the parameters of the outer code [35].

2.1 Local Correction

If the inner code $\mathcal{C}_0$ has a smooth local reconstruction procedure, then not only does $\mathcal{C}$ have good distance, but we show it can also be efficiently locally corrected. Our main result is the following theorem.

Theorem 5. *Let $\mathcal{C}_0$ be a linear code over Σ of length d and rate $r_0 > 1/2$. Suppose that $\mathcal{C}_0$ has a s_0-smooth local reconstruction procedure with query complexity q_0. Let $\mathcal{C} = \mathcal{C}_N(\mathcal{C}_0, G)$ be the expander code of length N arising from the inner code $\mathcal{C}_0$ and a Ramanujan graph G. Choose any $\gamma < 1/2$ and any $\zeta > \gamma$ satisfying $\gamma \left(e^{\zeta} q_0\right)^{-1/\gamma} > 8\lambda$. Then $\mathcal{C}$ is (q, ρ)-locally correctable, for any error rate ρ, with $\rho < \gamma \left(e^{\zeta} q_0\right)^{-1/\gamma} - 2\lambda$. The success probability is*

$$1 - \left(\frac{N}{d}\right)^{-1/\ln(d/4)}$$

and the query complexity is

$$q = \left(\frac{N}{d}\right)^{\varepsilon} \qquad \text{where} \qquad \varepsilon = \left(1 + \frac{\ln(q_0') + 1}{\zeta - \gamma}\right) \cdot \frac{\ln(q_0')}{\ln(d/4)}.$$

Further, when the length of the inner code, d, is constant, the correction algorithm runs in time $O(|\Sigma|^{q_0'+1} q)$, where $q_0' = q_0 + (d - s_0)$.

Remark 1. *We will choose d (and hence $q_0' < d$) and $|\Sigma|$ to be constant. Thus, the rate of $\mathcal{C}$, as well as the parameters ρ and ε, will be constants independent of the block length N. The parameter ζ trades off between the query complexity and the allowable error rate. When q_0 is much smaller than d (for example, $q_0 = 3$ and d is reasonably large), we will want to take $\zeta = O(1)$. On the other hand, if $q_0 = d^{\varepsilon}$ and d is chosen to be a sufficiently large constant, we should take ζ on the order of $\ln(q_0)$.*

Before diving into the details, we outline the correction algorithm. First, we observe that it suffices to consider the case when Q_0 is perfectly smooth: that is, the queries of the inner code are uniformly random. Otherwise, if Q_0 is s_0-smooth with q_0 queries, we may modify it so that it is d-smooth with $q_0 + (d - s_0)$ queries, by having it query extra points and then ignore them. Thus, we set $q_0' = q_0$ and assume in the following that Q_0 makes q_0 perfectly smooth queries.

Suppose that $\mathcal{C}_0$ has local reconstruction algorithm (Q_0, A_0), and we receive a corrupted codeword, w, which differs from a correct codeword c^* in at most a ρ fraction of the entries. Say we wish to determine $c^*[(u_0, v_1)]$, for $(u_0, v_1) \in E(H)$. The algorithm proceeds in two steps. The first step is to find a set of about $N^{\varepsilon/2}$ query positions which are nearly uniform in $[N]$, and whose correct values together determine $c^*[(u_0, v_1)]$. The second step is to correct each of *these* queries with very high probability—for each, we will make another $N^{\varepsilon/2}$ or so queries.

Step 1. By construction, $c^*[(u_0, v_1)]$ is a symbol in a codeword of the inner code, $\mathcal{C}_0$, which lies on the edges emanating from u_0. By applying Q_0, we may choose q_0 of these edges, $S = \left\{(u_0, s_1^{(i)}) : i \in [q_0]\right\}$, so that

$$A_0\left(c^*|_S, (u_0, v_1)\right) = c[(u_0, v_1)].$$

Now we repeat on each of these edges: each $(u_0, s_1^{(i)})$ is part of a codeword emanating from $s_1^{(i)}$, and so q_0 more queries determine each of those, and so on. Repeating this L_1 times yields a q_0-ary tree T of depth L_1, whose nodes are labeled by of edges of H. This tree-making procedure is given more precisely below in

Algorithm 2. Because the queries are smooth, each path down this tree is a random walk in H; because G is an expander, this means that the leaves themselves, while not independent, are each close to uniform on $E(H)$. Note that at this point, we have not made any queries, merely documented a tree, T, of edges we could query.

Step 2. Our next step is to actually make queries to determine the correct values on the edges represented in the leaves of T. By construction, these values determine $c^*[(u_0, v_1)]$. Unfortunately, in expectation a ρ fraction of the leaves are corrupted, and without further constraints on $\mathcal{C}_0$, even one corrupted leaf is enough to give the wrong answer. To make sure that we get all of the leaves correct, we use the fact that each leaf corresponds to a position in the codeword that is nearly uniform (and in particular nearly independent of the location we are trying to reconstruct). For each edge, e, of H that shows up on a leaf of T, we repeat the tree-making process beginning at this edge, resulting in new q_0-ary trees T_e of depth L_2. This time, we make all the queries along the way, resulting in an evaluated tree τ_e, whose nodes are labeled by elements of Σ; the root of τ_e is the e-th position in the corrupted codeword, $w[e]$, and we hope to correct it to $c^*[e]$.

For a fixed edge, e, on a leaf of T, we will correct the root of $\tau = \tau_e$ with very high probability, large enough to tolerate a union bound over all the trees τ_e. For two labelings σ and ν of the same tree by elements of Σ, we define the distance

$$D(\sigma, \nu) = \max_P \Delta\left(\sigma|_P, \nu|_P\right), \tag{2}$$

where the maximum is over all paths P from the root to a leaf, and $\sigma|_P$ denotes the restriction of σ to P. We will show below in Section 2.2 that it is very unlikely that τ contains a path from the root to a leaf with more than a constant fraction $\gamma < 1/2$ of errors. Thus, in the favorable case, the distance between the correct tree τ^* arising from c^* and the observed tree τ is at most $D(\tau^*, \tau) \leq \gamma$. In contrast, we will show that if σ^* and τ^* are both trees arising from legitimate codewords with distinct roots, then σ^* and τ^* must differ on an entire path P, and so $D(\sigma^*, \tau) > 1 - \gamma$. To take advantage of this, we show in Algorithm 3 how to efficiently compute

$$\mathsf{Score}(a) = \min_{\sigma^*:\mathrm{root}(\sigma^*)=a} D(\sigma^*, \tau)$$

for all a, where $\mathrm{root}(\sigma^*)$ denotes the label on the root of σ^*. The above argument (made precise below in Section 2.2) shows that there will be a unique $a \in \Sigma$ with score less than γ, and this will be the correct symbol $c^*[e]$.

Finally, with all of the leaves of T correctly evaluated, we may use A_0 to work our way back up T and determine the correct symbol corresponding to the edge at the root of T. The complete correction algorithm is given below in Algorithm 1.

Algorithm 1: correct: Local correcting protocol.

Input: An index $e_0 \in E(H)$, and a corrupted codeword $w \in \Sigma^{E(H)}$.
Output: With high probability, the correct value of the e_0'th symbol.
Set $L_1 = \log(n)/\log(d/4)$ and fix a parameter L_2
$T = \mathsf{makeTree}(e_0, L_1)$
for *each edge e of H that showed up on a leaf of T* **do**
  $T_e = \mathsf{makeTree}(e, L_2)$
  Let $\tau_e = T_e|_w$ be the tree of symbols from w
  $w^*[e] = \mathsf{correctSubtree}(\tau_e)$
Initialize a q_0-ary tree τ^* of depth L_1
Label the leaves of τ^* according to T and w^*: if a leaf of T is labeled e, label the corresponding leaf of τ^* with $w^*[e]$.
Use the local reconstruction algorithm A_0 of $\mathcal{C}_0$ to label all the nodes in τ^*
return The label on the root of τ^*

Algorithm 2: makeTree: Uses the local correction property of $\mathcal{C}_0$ to construct a tree of indices.

Input: An initial edge $e_0 = (u_0, v_1) \in E(H)$, and a depth L.
Output: A q_0-ary tree T of depth L, whose nodes are indexed by edges of H, with root e_0
Initialize a tree T with a single node labeled e_0
$s = 0$
for $\ell \in [L]$ **do**
  Let leaves be the current leaves of T
  for $e = (u_s, v_{1-s}) \in$ *leaves* **do**
    Let $\left\{v_{1-s}^{(i)} : i \in [d]\right\}$ be the neighbors of u_s in H
    Choose queries $Q_0(e) \subset \left\{(u_s, v_{1-s}^{(i)}) : i \in [d]\right\}$, and add each query in T as a child at e.
  $s = 1 - s$
return T

Algorithm 3: correctSubtree: Correct the root of a fully evaluated tree τ.

Input: τ, a q_0-ary tree of depth L whose nodes are labeled with elements of Σ.
Output: A guess at the root of the correct tree τ.
For a node x of τ, let $\tau[x]$ denote the label on x.
for *leaves* x *of* τ *and* $a \in \Sigma$ **do**
  $\mathsf{best}_a(x) = \begin{cases} 1 & \tau[x] \neq a \\ 0 & \tau[x] = a \end{cases}$
for $\ell = L-1, L-2, \ldots, 0$ **do**
  for *nodes* x *at level* ℓ *in* τ *and* $a \in \Sigma$ **do**
    Let $y_1, \ldots, y_{q_0}$ be the children of x
    Let $S_a \subset \Sigma^{q_0}$ be the set of query responses for the children of x so that A_0 returns a on those responses
    $\mathsf{best}_a(x) = \min_{(a_0,\ldots,a_{q_0}) \in S_a} \max_{r \in [q_0]} \left(\mathsf{best}_{a_r}(y_r) + \mathbf{1}_{\tau(y_r) \neq a_r}\right)$
Let r be the root of τ
for $a \in \Sigma$ **do**

$$\mathsf{Score}(a) = \frac{\mathsf{best}_a(r) + \mathbf{1}_{\tau(r) \neq a}}{L}$$

return $a \in \Sigma$ *with the smallest* Score(a)

The number of queries made by Algorithm 1 is

$$q = q_0^{L_1 + L_2} \tag{3}$$

and the running time is $O(t_d |\Sigma|^{q_0+1} q)$, where t_d is the time required to run the local correction algorithm of $\mathcal{C}_0$. For us, both d and $|\Sigma|$ will be constant, and so the running time is $O(q)$.

2.2 Proof of Theorem 5

Suppose that $c^* \in \mathcal{C}$, and Algorithm 1 is run on a received word w with $\Delta(c^*, w) \leq \rho$. To prove Theorem 5, we must show that Algorithm 1 returns $c^*[e_0]$ with high probability. As remarked above, we assume that Q_0 is perfectly smooth.

We follow the proof outline sketched in Section 2.1, which rests on the following observation.

Proposition 6. *Let $c_1, c_2 \in \mathcal{C}$ and let $e \in E(H)$ so that $c_1[e] \neq c_2[e]$. Let the distance D between trees with labels in Σ be as in (2). Let $T = \mathsf{makeTree}(e)$, and let $\tau = T|_{c_1}$ and $\sigma = T|_{c_2}$ be the labeled trees corresponding to c_1 and c_2 respectively. Then $D(\tau, \sigma) = 1$. That is, there is some path from the root to the leaf of T so that τ and σ disagree on the entire path.*

Proof. Since $c_1[e] \neq c_2[e]$, τ and σ have different symbols at their root. Since the labels on the children of any node determine the label on the node itself (via the local correction algorithm), it must be that τ and σ differ on some child of the root. Repeating the argument proves the claim. □

Let τ_e be the tree arising from the received word w, starting at e, as in Algorithm 1. Let

$$\mathcal{T}_e = \{\, \mathsf{makeTree}(e)|_c \,:\, c \in \mathcal{C}\}$$

be the set of query trees arising from uncorrupted codewords, and let $\tau_e^* \in \mathcal{T}_e$ be the "correct" tree, corresponding to the original uncorrupted codeword c^*. Suppose that

$$D(\tau_e, \tau_e^*) \leq \gamma \tag{4}$$

for some $\gamma \in [0, 1/2)$. Then Proposition 6 implies that for any $\sigma_e^* \in \mathcal{T}_e$ with a different root from τ_e^* has

$$D(\tau_e, \sigma_e^*) \geq 1 - \gamma. \tag{5}$$

Indeed, there is some path along which τ_e^* and σ_e^* differ in every place, and along this path, τ_e agrees with τ_e^* in at least a $1 - \gamma$ fraction of the places. Thus, τ_e disagrees with σ_e^* in those same places, establishing (5). Consider the quantity

$$\mathsf{Score}(a) = \min_{\sigma_e^* \in \mathcal{T}_e : \mathrm{root}(\sigma_e^*) = a} D(\tau_e, \sigma_e^*). \tag{6}$$

Equations (4) and (5) imply that if a^* is the label on the root of τ_e^*, then $\mathsf{Score}(a) \leq \gamma$, and otherwise, $\mathsf{Score}(a) \geq 1 - \gamma$. Thus, to establish the correctness of Algorithm 1, it suffices to argue first that Algorithm 3 correctly computes $\mathsf{Score}(a)$ for each a, and second that (4) holds for all trees τ_e in Algorithm 1.

The first claim follows by inspection. Indeed, for a node $x \in \tau_e$, let $(\tau_e)_x$ denote the subtree below x. Let $\mathcal{T}_e^{(x,a)}$ denote the set of trees in $\mathcal{T}_e$ so that the node x is labeled a. Throughout Algorithm 1, the quantity $\mathsf{best}_a(x)$ gives the distance from the observed tree rooted at x to the best tree in $\mathcal{T}_e$, rooted at x, with the additional restriction that the label at x should be a. That is,

$$\mathsf{best}_a(x) = \min_{\sigma_e^* \in \mathcal{T}_e^{(x,a)}} \tilde{D}\left((\sigma_e^*)_x, (\tau_e)_x\right), \tag{7}$$

where $\tilde{D}$ is the same as D except it does not count the root, and it is not normalized. It is easy to see that (7) is satisfied for leaves x of τ_e. Then for each node, Algorithm 3 updates $\mathsf{best}_a(x)$ by considering the best labeling on the children of x consistent with $\tau(x) = a$, taking the distance of the worst of those children, and adding one if necessary.

To establish the second claim, that (4) holds for all trees τ_e, we will need the following lemma about random walks on H.

Lemma 7. *Let G and H be as above, and suppose $\rho > 6\lambda$. Let $v_0, \ldots, v_L$ be a random walk of length L on H, starting from the left side at a vertex chosen from a distribution ν with $\left\|\nu - \frac{1}{n}\mathbf{1}_n\right\|_2 \leq \frac{1}{\sqrt{n}}$. Let X denote the number of corrupted edges included in the walk, and let $\rho + 2\lambda < \gamma < 1/2$. Then*

$$\mathbb{P}\{X \geq \gamma L\} \leq \exp\left(-L\, D\left(\gamma || \rho + 2\lambda\right)\right).$$

Lemma 7 says that a random walk on H will not hit too many corrupted edges, which is very much like the expander Chernoff bound [22, 19]. In this case, H is the double cover of an expander, not an expander itself, and the edges, rather than vertices, are corrupted, but the proof remains basically the same. For

completeness, we include the proof of Lemma 7 in the appendix. The conditions on ρ and λ in the statement of Theorem 5 implies that $\rho > 6\lambda$, and so Lemma 7 applies to random walks on H.

Suppose that L_1 is even, and consider any leaf of T. This leaf has label $(u_0, v_1) \in E(H)$, where u is the result of a random walk of length L_1 on G and v is a randomly chosen neighbor of u. Because G is a Ramanujan graph, the distribution μ on u satisfies

$$\left\| \mu - \frac{1}{n}\mathbf{1}_n \right\|_2 \leq \lambda^{L_1} \leq \frac{1}{\sqrt{n}}$$

as long as

$$L_1 \geq \frac{\log(n)}{\log(d/4)}.$$

Thus, Lemma 7 applies to random walks in H starting at e. Fix a leaf of τ_e; by the smoothness of the query algorithm Q_0, each path from the root to the leaf of each tree τ_e is a uniform random walk, and so with high probability, the number of corrupted edges on this walk is not more than γL_2, which was the desired outcome. The failure probability guaranteed by Lemma 7 is at most

$$\begin{aligned}
\exp(-L_2 D(\gamma || \rho + 2\lambda)) &= \left(\frac{\rho + 2\lambda}{\gamma}\right)^{\gamma L_2} \left(\frac{1 - \rho - 2\lambda}{1 - \gamma}\right)^{(1-\gamma)L_2} \\
&\leq \left(e^{\zeta} q_0\right)^{-L_2} \left(\frac{1}{1-\gamma}\right)^{(1-\gamma)L_2} \\
&\leq \left(e^{\zeta} q_0\right)^{-L_2} e^{\gamma L_2}.
\end{aligned}$$

Above, we used the assumption that $\rho + 2\lambda < \gamma \left(e^{\zeta} q_0\right)^{-1/\gamma}$ from the statement of Theorem 5.

Finally, we union bound over $q_0^{L_1}$ trees τ_e and $q_0^{L_2}$ paths in each tree. We will set $L_2 = CL_1$, for a constant C to be determined. Thus, (4) holds (and hence Algorithm 1 is correct) except with probability at most

$$\begin{aligned}
\mathbb{P}\left\{\text{Algorithm 1 fails}\right\} &\leq q_0^{L_1 + L_2} \left(e^{\zeta} q_0\right)^{-L_2} e^{\gamma L_2} \\
&= \exp\left((C+1)L_1 \ln(q_0) - CL_1(\zeta + \ln(q_0)) + C\gamma L_1\right). \qquad (8)
\end{aligned}$$

Our goal is to show that $\mathbb{P}\left\{\text{Algorithm 1 fails}\right\} \leq \exp(-L_1)$, which is equivalent to showing

$$(C+1)\ln(q_0) - C(\zeta + \ln(q_0)) + C\gamma < -1.$$

This holds if we choose

$$C < \frac{1 + \ln(q_0)}{\zeta - \gamma}.$$

From (3), $q = q_0^{(C+1)L_1}$, which completes the proof of Theorem 5.

3 Examples

In this section, we provide two examples of choices for $\mathcal{C}_0$, both of which result in (N^{ε}, ρ)-LCCs of rate $1 - \alpha$ for any constants $\varepsilon, \alpha > 0$ and for some constant $\rho > 0$. Our first and main example is a generalization of Reed-Muller codes, based on finite geometries. With these codes as $\mathcal{C}_0$, we provide LCCs over $\mathbb{F}_p$—unlike multiplicity codes, these codes work naturally over small fields.

Our second example comes from the observation that if the $\mathcal{C}_0$ is itself an LCC (of a fixed length) our construction provides a new family of (N^{ε}, ρ)-LCCs. In particular, plugging the multiplicity codes of [25] into our construction yields a novel family of LCCs. This new family of LCCs has a very different structure than the underlying multiplicity codes, but achieves roughly the same rate and locality.

Codes from Affine Geometries. One advantage of our construction is that the inner code $\mathcal{C}_0$ need not actually be a good locally decodable or correctable code. Rather, we only need a smooth reconstruction procedure, which is easier to come by. One example comes from affine geometries; in this example, we will show how use Theorem 5 to make LCCs of length N, rate $1-\alpha$ and query complexity N^{ε}, for any $\alpha, \varepsilon > 0$.

For a prime power $h = p^{\ell}$ and parameters r and m, consider the r-dimensional affine subspaces $L_1, \ldots, L_t$ of the vector space $\mathbb{F}_h^m$. let H be the $t \times h^m$ incidence matrix of the L_i and the points of $\mathbb{F}_h^m$, and let $\mathcal{A}^*(r, m, h)$ be the code over $\mathbb{F}_p$ whose parity check matrix is H. These codes, examples of *finite geometry codes,* are well-studied, and their ranks can be exactly computed—see [2, 3] for an overview.

The definition of of $\mathcal{A}^*(r, m, h)$ gives a reconstruction procedure: we may query all the points in a random r-dimensional affine subspace of $\mathbb{F}_h^m$ and use the corresponding parity check. In particular, if we index the positions of the codeword by elements of $\mathbb{F}_h^m$. Then given the position $x \in \mathbb{F}_h^m$, the query set $Q(x)$ is all the points other than x in a random r-flat L that passes through x. Given a codeword $c \in \mathcal{A}^*(r, m, h)$, we may reconstruct c_x by

$$A\left(c|_{Q(x)}\right) = -\sum_{y \in Q(x)} c_y.$$

By definition, (A, Q) is a smooth reconstruction procedure which makes h^r queries.

The locality of $\mathcal{A}^*(r, m, h)$ has been noticed before, for example in [20], where it was observed that these codes could be viewed as lifted parity check codes. However, as they note, these codes do not themselves make good LCCs—the reconstruction procedure cannot tolerate any errors in the chosen subspace, and thus the error rate ρ must tend to zero as the block length grows. Even though these codes are not good LCCs, we can use them in Theorem 5 to obtain good LCCs with sublinear query complexity, which can correct a constant fraction of errors. We will use the bound on the rate of $\mathcal{A}^*(1, m, h)$ from [20]:

Lemma 8 (Lemma 3.7 in [20])**.** *Choose $\ell = \varepsilon m$, with $h = p^{\ell}$ as above. The dimension of $\mathcal{A}^*(1, m, h)$ is at least $h^m - h^{m(1-\beta)}$, for $\beta = \beta(\varepsilon') = \Omega(2^{-2/\varepsilon'})$.*

We will apply Lemma 8 with

$$\varepsilon' = \frac{\varepsilon}{2} \qquad \text{and} \qquad m = \sqrt{\frac{\ln(2/\alpha)}{\varepsilon' \beta(\varepsilon') \ln(p)}},$$

to obtain a p-ary code $\mathcal{C}_0$ of length $d = p^{\varepsilon' m^2}$ with rate r_0 at least $1 - \alpha/2$ and which has a $(d-1)$-smooth reconstruction algorithm with query complexity $q_0 = d^{\varepsilon'}$. To apply Theorem 5, fix any $\varepsilon, \alpha > 0$, sufficiently small. We set $\zeta = 2\ln(q_0)$, and choose $\gamma = 1/4$ in Theorem 5, and use $\mathcal{C}_0$: the resulting expander code $\mathcal{C}$ has rate $1 - \alpha$ and query complexity

$$q \leq \left(\frac{N}{d}\right)^{\varepsilon}$$

for sufficiently large d. Finally, using the fact that $\lambda \leq 2/\sqrt{d}$, we see that $\mathcal{C}$ corrects against a ρ fraction of errors, where

$$\rho = \frac{1}{5} d^{-6\varepsilon'}$$

again for sufficiently large d, as long as $\varepsilon < 1/12$. Assuming ε and α are small enough that d is a suitably large constant, this rate ρ is a positive constant, and we achieve the advertised results.

Multiplicity codes. Multiplicity codes [25] are themselves a family of constant-rate locally decodable codes. We can, however, use a multiplicity code of constant length as the inner code $\mathcal{C}_0$ in our construction. This results in a new family of constant-rate locally decodable codes. The parameters we obtain from this construction are slightly worse than the original multiplicity codes, and the main reason we include this example is novelty—these new codes have a very different structure than the original multiplicity codes.

For constants $\alpha', \varepsilon' > 0$, the multiplicity codes of [25] have length d and rate $r_0 = 1 - \alpha'$ and a $(d-1)$-smooth local reconstruction algorithm with query complexity $q_0 = O(d^{\varepsilon'})$. To apply Theorem 5, we will choose $\zeta = C \ln(q_0)$ for a sufficiently large constant C, and so the query complexity of $\mathcal{C}$ will be

$$q = \left(\frac{N}{d}\right)^{(1+\beta)\varepsilon'}$$

for an arbitrarily small constant β. Thus, setting $\varepsilon = \varepsilon'(1+\beta)$, and $\alpha = 2\alpha'$, we obtain codes $\mathcal{C}$ with rate $1-\varepsilon$ and query complexity $(N/d)^{\varepsilon}$. As long as ε is sufficiently small, $\mathcal{C}$ can tolerate errors up to $\rho = C' d^{-C''\varepsilon}$ for constants C' and C'' (depending on the constants in the constructions of the multiplicity code, as well as on C above). Multiplicity codes require sufficiently large block length d, on the order of

$$d \approx \left(\frac{1}{\alpha^2 \varepsilon^3}\right)^{1/\varepsilon} \log\left(\frac{1}{\alpha\varepsilon}\right).$$

Choosing this d results in a requirement $\rho \leq 1/\mathrm{poly}(\alpha\varepsilon)$. We remark that the distance of the multiplicity codes is on the order of $\delta_0 = \Omega(\alpha^2\varepsilon)$, and so the distance of the resulting expander code $\mathcal{C}$ is $\Omega(\alpha^4\varepsilon^2)$.

4 Conclusion

In the constant-rate regime, all known LDCs work by using a smooth local reconstruction algorithm. When the locality is, say, three, then with very high probability none of the queried positions will be corrupted. This reasoning fails for constant rate codes, which have larger query complexity: we expect a ρ fraction of errors in our queries, and this is often difficult to deal with. In this work, we have shown how to make the low-query argument valid in a high-rate setting—any code with large enough rate and with a good local reconstruction algorithm can be used to make a full-blown locally correctable code.

The payoff of our approach is the first sublinear time algorithm for locally correcting expander codes. More precisely, we have shown that as long as the inner code $\mathcal{C}_0$ admits a smooth local reconstruction algorithm with appropriate parameters, then the resulting expander code $\mathcal{C}$ is a (N^{ε}, ρ)-LCC with rate $1-\alpha$, for any $\alpha, \varepsilon > 0$ and some constant ρ. Further, we presented a decoding algorithm with runtime linear in the number of queries.

There are only two other constructions known in this regime, and and our constructions are substantially different. Expander codes are a natural construction, and it is our hope that the additional structure of our codes, as well as the extremely fast decoding time, will lead to new applications of local decodability.

References

[1] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized graph products. *Computational Complexity*, 5(1):60–75, 1995.

[2] E.F. Assmus and J.D. Key. *Designs and their Codes*, volume 103. Cambridge University Press, 1994.

[3] E.F. Assmus and J.D. Key. Polynomial codes and finite geometries. *Handbook of coding theory*, 2(part 2):1269–1343, 1998.

[4] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC '91, pages 21–32, New York, NY, USA, 1991. ACM.

[5] A. Barg and G. Zemor. Error exponents of expander codes. *Information Theory, IEEE Transactions on*, 48(6):1725–1729, June 2002.

[6] A. Barg and G. Zemor. Concatenated codes: serial and parallel. *IEEE Trans. Inf. Theor.*, 51(5):1625–1634, May 2005.

[7] A. Barg and G. Zemor. Distance properties of expander codes. *Information Theory, IEEE Transactions on*, 52(1):78–90, January 2006.

[8] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Ilan Orlov. Share Conversion and Private Information Retrieval. In *CCC '12*, volume 0, pages 258–268, Los Alamitos, CA, USA, 2012. IEEE Computer Society.

[9] A. Ben-Aroya, K. Efremenko, and A. Ta-Shma. Local List Decoding with a Constant Number of Queries. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 715–722. IEEE, October 2010.

[10] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, STOC '90, pages 73–83, New York, NY, USA, 1990. ACM.

[11] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, December 1993.

[12] Yeow M. Chee, Tao Feng, San Ling, Huaxiong Wang, and Liang F. Zhang. Query-Efficient Locally Decodable Codes of Subexponential Length. *Computational Complexity*, pages 1–31, August 2011.

[13] Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching Vector Codes. *SIAM Journal on Computing*, 40(4):1154–1178, January 2011.

[14] Klim Efremenko. 3-query locally decodable codes of subexponential length. In *STOC '09*, pages 39–44. ACM, 2009.

[15] Klim Efremenko. From irreducible representations to locally decodable codes. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 327–338, New York, NY, USA, 2012. ACM.

[16] R. G. Gallager. Low Density Parity-Check Codes. Technical report, MIT, 1963.

[17] Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *STOC '91*, pages 33–42, New York, NY, USA, 1991. ACM.

[18] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169–174, September 1992.

[19] D. Gillman. A chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998.

[20] A. Guo, S. Kopparty, and M. Sudan. New affine-invariant codes from lifting. In *ITCS*, 2013.

[21] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–562, 2006.

[22] R. Impagliazzo and V. Kabanets. Constructive proofs of concentration bounds. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 617–631, 2010.

[23] Toshiya Itoh and Yasuhiro Suzuki. New Constructions for Query-Efficient Locally Decodable Codes of Subexponential Length. *IEICE Transactions on Information and Systems*, E93-D(2):263–270, October 2010.

[24] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *STOC '00*, pages 80–86, 2000.

[25] S. Kopparty, S. Saraf, and S. Yekhanin. High-rate codes with sublinear-time decoding. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 167–176. ACM, 2011.

[26] Richard J. Lipton. Efficient checking of computations. In *Proceedings of the seventh annual symposium on Theoretical aspects of computer science*, STACS 90, pages 207–215, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[27] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[28] G.A. Margulis. Explicit group theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problems of Information Transmission*, 9(1):39–46, 1988.

[29] M. Morgenstern. 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.

[30] Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 194–203, New York, NY, USA, 1994. ACM.

[31] I. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Information Theory, Transactions of the IRE Professional Group on*, 4(4):38–49, September 1954.

[32] M. Sipser and D.A. Spielman. Expander codes. *Information Theory, IEEE Transactions on*, 42(6):1710–1722, 1996.

[33] D. A. Spielman. Highly fault-tolerant parallel computation. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 154–163. IEEE, October 1996.

[34] D. A. Spielman. Linear-time encodable and decodable error-correcting codes. *Information Theory, IEEE Transactions on*, 42(6):1723–1731, November 1996.

[35] R. Tanner. A recursive approach to low complexity codes. *Information Theory, IEEE Transactions on*, 27(5):533–547, 1981.

[36] Sergey Yekhanin. Towards 3-Query Locally Decodable Codes of Subexponential Length. In *STOC '07*, pages 266–274. ACM, 2007.

[37] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1), 2008.

[38] Sergey Yekhanin. Locally Decodable Codes. *Foundations and Trends in Theoretical Computer Science*, 2010.

[39] G. Zemor. On expander codes. *Information Theory, IEEE Transactions on*, 47(2):835–837, 2001.

A Proof of Lemma 7

In this appendix, we provide a proof of Lemma 7. The lemma follows with only a few tweaks from standard results. The only differences between this and a standard analysis of random walks on expander graphs are that (a) we are walking on the edges of the bipartite graph H, rather than on the vertices of G, and (b) our starting distribution is not uniform but instead close to uniform. Dealing with this differences is straightforward, but we document it below for completeness.

First, we need the relationship between a walk on the edges of a bipartite graph H and the corresponding walk on the vertices of G. For ease of analysis, we will treat H as directed, with one copy of each edge in each direction.

Lemma 9. *Let G be a degree d undirected graph on d vertices with normalized adjacency matrix A, and let H be the double cover of G. For each vertex v of G, label the edges incident to v arbitrarily, and let $v(i)$ denote the i^{th} edge of v. Let H' be the graph with vertices $V(G) \times [d] \times \{0,1\}$ and edges*

$$E(H') = \{((u,i,b),(v,j,b')) \,:\, (u,v) \in E(G), b \neq b', u(i) = v\}.$$

Then H' is a directed graph with $2dn$ edges, and in-degree and out-degree both equal to d. Further, the normalized adjacency matrix A' is given by

$$A' = R \otimes S$$

where $S : \mathbb{R}^2 \to \mathbb{R}^2$ is $S = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $R : \mathbb{R}^{nd} \to \mathbb{R}^{nd}$ is an operator with the same rank and spectrum as A.

Proof. We will write down A' in terms of A. Index $[n]$ by vertices of V, so that $e_v \in \mathbb{R}^n$ refers to the standard basis vector with support on v. Let $\otimes$ denote the Kronecker product. We will need some linear operators. Let $B : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ so that

$$B(e_u \otimes e_v) = e_v \otimes e_v$$

and $P : \mathbb{R}^{n^2} \to \mathbb{R}^{nd}$ so that

$$P(e_u \otimes e_v) = \begin{cases} e_u \otimes e_i & v = u(i) \\ 0 & (u,v) \notin E(G) \end{cases}.$$

Finally, let $S : \mathbb{R}^2 \to \mathbb{R}^2$ be the cyclic shift operator. Then a computation shows that the adjacency matrix A' of H' is given by

$$(P(I \otimes A)BP^T) \otimes S.$$

Let $R = P(I \otimes A)BP^T$. To see that the rank of R is at most n, note that for any $i \in [d]$ and any $u \in V(G)$,

$$R(e_u \otimes e_j) = e_{u(j)} \otimes \frac{1}{d}\mathbf{1}_d.$$

In particular, it does not depend on the choice of j. Since $\{e_u \otimes e_j \,:\, u \in V(G), j \in [d]\}$ is a basis for $\mathbb{R}^{nd}$, the image of R has dimension at most n. Finally, a similar computation shows that if p is an eigenvector of A with eigenvalue λ, then $p \otimes \frac{1}{d}\mathbf{1}_d$ is a right eigenvector of R, also with eigenvalue λ. (The left eigenvectors are $P(\frac{1}{n}\mathbf{1}_n \otimes p)$). This proves the claim. ☐

With a characterization of A' in hand, we now wish to apply an expander Chernoff bound. Existing bounds require slight modification for this case (since the graph H' is directed and also not itself an expander), so for completeness we sketch the changes required. The proof below follows the strategies in [1] and [22]. We begin with the following lemma, following from the analysis of [1].

Lemma 10. *Let G and H be as in Lemma 9, and let $v_0, v_1, \ldots, v_T$ be a random walk on the vertices of H, beginning at a vertex of H, chosen as follows: the side of H is chosen according to a distribution $\sigma_0 = (s, 1-s)$, and the vertex within that side is chosen independently according to a distribution ν with $\|\nu - \frac{1}{n}\mathbf{1}_n\|_2 \leq \frac{1}{\sqrt{n}}$. Let W be any set of edges in H, with $|W| \leq \rho nd$. Suppose that $\rho > 6\lambda$. Then for any set $S \subset \{0, 1, \ldots, T-1\}$,*

$$\mathbb{P}\{(v_t, v_{t+1}) \in W, \forall t \in S\} \leq (\rho + 2\lambda)^{|S|}.$$

Proof. As in Lemma 9, we will consider H as directed, with one edge in each direction. As before, we will index these edges by triples $(u, i, \ell) \in V(G) \times [d] \times \{0,1\}$, so that (u, i, ℓ) refers to the i^{th} edge leaving vertex u on the ℓ^{th} side of H. Let μ be the distribution on the first step (v_0, v_1) of the walk, so

$$\mu = \nu \otimes \frac{1}{d}\mathbf{1}_d \otimes \sigma_0.$$

Let $M \in \mathbb{R}^{2nd}$ be the projector onto the edges in W. Let $M^{(0)}$ be the restriction to edges emanating from the left side of H, and $M^{(1)}$ from the right side, so that both $M^{(0)}$ and $M^{(1)}$ are $nd \times nd$ binary diagonal

matrices with at most ρnd nonzero entries. Let $A' = R \otimes S$ be as in the conclusion of Lemma 9. After running the random walk for T steps, consider the distribution on directed edges of H, conditional on the bad event that $(v_t, v_{t+1}) \in W$ for all $t \in S$. As in the analysis in [1], this distribution is given by

$$\mu_T = \frac{(M_{T_1}A')(M_{T-2}A')\cdots(M_1A')(M_0\mu)}{\mathbb{P}\left\{(v_t, v_{t+1}) \in W, \forall t \in S\right\}},$$

where

$$M_t = \begin{cases} M & t \in S \\ I & t \notin S \end{cases}.$$

Since the ℓ_1 norm of any distribution is 1, we have

$$\mathbb{P}\left\{(v_t, v_{t+1}) \in W, \forall t \in S\right\} = \left\|(M_{T-1}A')(M_{T-2}A')\cdots(M_1A')(M_0\mu)\right\|_1 \tag{9}$$

Let

$$\mu_0 := M_0\mu,$$

and

$$\mu_t := M_tA'\mu_{t-1},$$

so we seek an estimate on $\|\mu_T\|_1$.

The following claim will be sufficient to prove the theorem.

Claim 11. *If $\rho \geq 6\lambda$, and $t \in S$,*

$$(\mu - 2\lambda)\left\|\mu_t\right\|_1 \leq \left\|\mu_{t+1}\right\|_1 \leq (\mu + 2\lambda)\left\|\mu_t\right\|_1.$$

On the other hand, if $t \notin S$,

$$\left\|\mu_t\right\|_1 = \left\|\mu_{t+1}\right\|_1.$$

The second half of the claim follows immediately from the definition of μ_t. To prove the first half, suppose that $t \in S$. We will proceed by induction. Again, we follow the analysis of [1].

Write $\mu_0 = v_0 \otimes \sigma_0$, and write $\sigma_0 = (s, 1-s)$ Part of our inductive hypothesis will be that for all t,

$$\mu_t = v_t^{(0)} \otimes s_te_0 + v_t^{(1)} \otimes (1-s_t)e_1,$$

where $s_t = s$ if t is even and $1-s$ if t is odd, and where $v_t^{(i)} \in \mathbb{R}^{nd}$. For $i \in \{0,1\}$, write

$$v_t^{(i)} = x_t^{(i)} + y_t^{(i)},$$

where $x_t^{(i)} \| \mathbf{1}$ and $y_t^{(i)} \perp \mathbf{1}$. The second part of the inductive hypothesis will be

$$\|y_t^{(i)}\|_2 \leq q\|x_t^{(i)}\|_2, \tag{10}$$

for a parameter q to be chosen later, and for $i \in \{0,1\}$.

Because

$$\begin{aligned}\|\mu_t\|_1 &= s_t\|v_t^{(0)}\|_1 + (1-s_t)\|v_t^{(1)}\|_1 \\ &= s_t\|x_t^{(0)}\|_1 + (1-s_t)\|x_t^{(1)}\|_1 \\ &= \sqrt{nd}\left(s_t\|x_t^{(0)}\|_2 + (1-s_t)\|x_t^{(1)}\|_2\right),\end{aligned}$$

it suffices to show that

$$(\mu - 2\lambda)\left\|x_t^{(0)}\right\|_2 \leq \left\|x_{t+1}^{(1)}\right\|_2 \leq (\mu + 2\lambda)\left\|x_t^{(0)}\right\|_2 \tag{11}$$

and similarly with the 0 and 1 switched. The analysis is the same for the two cases, so we just establish (11). Using the decomposition $A' = R \otimes S$ from Lemma 9,

$$\begin{aligned}\mu_{t+1} &= M_t(R \otimes S)(v_t^{(0)} \otimes s_t e_0 + v_t^{(1)} \otimes (1-s_t)e_1) \\ &= M_t\left(Rv_t^{(0)} \otimes (1-s_{t+1})e_1 + Rv_t^{(1)} \otimes s_{t+1}e_0\right) \\ &= \left(M_t^{(1)}Rv_t^{(0)}\right) \otimes (1-s_{t+1})e_1 + \left(M_t^{(0)}Rv_t^{(1)}\right) \otimes s_{t+1}e_0\end{aligned}$$

This establishes the first inductive claim about the structure of μ_{t+1}, and

$$v_{t+1}^{(0)} = M_t^{(0)}Rv_t^{(1)} \qquad \text{and} \qquad v_{t+1}^{(1)} = M_t^{(1)}Rv_t^{(0)}.$$

Consider just $v_{t+1}^{(1)}$. We have

$$v_{t+1}^{(1)} = M_t^{(1)}R(x_t^{(0)} + y_t^{(0)}).$$

Because $t \in S$, we know that $M_t^{(1)}$ is diagonal with at most ρnd nonzeros, and further we know that R has second normalized eigenvalue at most λ, by Lemma 9. The analysis in [1] now shows that, using the inductive hypothesis (10),

$$\rho\|x_t^{(0)}\|_2 - q\lambda\sqrt{\rho(1-\rho)}\|x_t^{(0)}\|_2 \le \|x_{t+1}^{(1)}\|_2 \le \rho\|x_t^{(0)}\|_2 + q\lambda\sqrt{\rho(1-\rho)}\|x_t^{(0)}\|_2, \tag{12}$$

and that

$$\|y_{t+1}^{(1)}\|_2 \le q\lambda\|x_t^{(0)}\|_2 + \sqrt{\rho(1-\rho)}\|x_t^{(0)}\|_2.$$

We must ensure that (10) is satisfied for the next round. As long as $\lambda < \rho/6$, this follows from the above when

$$q = 2\sqrt{\frac{1-\rho}{\rho}}.$$

With this choice of q, the (11) follows from (12). Further, the hypotheses on ν show that the (10) is satisfied in the initial step. □

Finally, we invoke the following theorem, from [22].

Theorem 12 (Theorem 3.1 in [22])**.** *Let $X_1, \ldots, X_L$ be binary random variables so that for all $S \subset [L]$,*

$$\mathbb{P}\left\{\bigwedge_{i \in S} X_i = 1\right\} \le \delta^{|S|}.$$

Then for all $\gamma > \delta$,

$$\mathbb{P}\left\{\sum_{i=1}^{L} X_i \ge \gamma L\right\} \le e^{-LD(\gamma||\delta)}.$$

Lemma 7 follows immediately.