Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 162 (2020)

High-Probability List-Recovery, and Applications to Heavy Hitters

Dean Doron[*]
Department of Computer Science
Stanford University
ddoron@stanford.edu

Mary Wootters[†]
Departments of Computer Science
and Electrical Engineering
Stanford University
marykw@stanford.edu

Abstract

An error correcting code $\mathcal{C}\colon \Sigma^k \to \Sigma^n$ is efficiently list-recoverable from input list size ℓ if for any sets $\mathcal{L}_1, \dots, \mathcal{L}_n \subseteq \Sigma$ of size at most ℓ, one can efficiently recover the list $\mathcal{L} = \{x \in \Sigma^k : \forall j \in [n], \mathcal{C}(x)_j \in \mathcal{L}_j\}$. While list-recovery has been well-studied in error correcting codes, all known constructions with "efficient" algorithms are not efficient in the parameter ℓ. In this work, motivated by applications in algorithm design and pseudorandomness, we study list-recovery with the goal of obtaining a good dependence on ℓ. We make a step towards this goal by obtaining it in the weaker case where we allow a *randomized* encoding map and a small failure probability, and where the input lists are derived from unions of codewords. As an application of our construction, we give a data structure for the heavy hitters problem in the strict turnstile model that, for some parameter regimes, obtains stronger guarantees than known constructions.

[*]Supported by NSF award CCF-1763311.
[†]Supported by NSF award CCF-1844628 and NSF-BSF award CCF-1814629.

ISSN 1433-8092

1 Introduction

Let $\mathcal{C}\colon \Sigma^k \to \Sigma^n$ be an *error correcting code*. We say that $\mathcal{C}$ is (efficiently) *list-recoverable*[1] from list-size ℓ with output list-size L if, for any lists $\mathcal{L}_1, \dots, \mathcal{L}_n \subseteq \Sigma$ with $|\mathcal{L}_i| \leq \ell$ for all i, there is an (efficient) algorithm to recover the list

$$\mathcal{L} = \{x \in \Sigma^k : \forall i \in [n], \mathcal{C}(x)_i \in \mathcal{L}_i\},$$

and $|\mathcal{L}| \leq L$. List recovery has historically been studied in the context of *list-decodable* codes, where it has been used as a tool to obtain efficient list-decoding algorithms (see, e.g., [GS98, GR08, GW13, Kop15, HRZW19]). However, even though efficient list-recovery algorithms have been developed, all of them have a poor dependence on the parameter ℓ. For example, [HRZW19] presents near-linear-time (in n) list-recovery algorithms, but the output list $\mathcal{L}$ has size doubly-exponential in ℓ.

In this work, we are motivated by the following goal (which we do not fully achieve):

Goal 1.1. *For $\ell \geq 2$, design a family of codes $\mathcal{C}\colon \Sigma^k \to \Sigma^n$ so that:*

1. *$\mathcal{C}$ can be encoded in time $O(n)$;*
2. *The* rate *k/n of the code is a constant (independent of n **and** ℓ);*
3. *The* alphabet size *$|\Sigma|$ is polynomial in ℓ (and independent of n);*
4. *The code $\mathcal{C}$ can be list-recovered in time $O(n \cdot \ell)$ (linear in both n **and** ℓ), with output list size $|\mathcal{L}| = O(\ell)$.*

To the best of our knowledge, this goal is open even if we allow the output list size $|\mathcal{L}|$ and the running time to depend polynomially on ℓ, rather than linearly.

Goal 1.1 is desirable for several reasons. First, it represents a bottleneck in our understanding of algorithmic coding theory, and it seems likely that achieving it would involve developing new techniques that would be useful elsewhere. Second, list-recovery with reasonable dependence on ℓ is related to questions in pseudorandomness, where the the parameter ℓ is often very large (see our discussion in Section 1.2). Third, as we explore in this paper, obtaining Goal 1.1 has applications in algorithm design, in particular to algorithms for heavy hitters.

Probabilistic list-recovery with good dependence on ℓ. In this work, we make progress on Goal 1.1 by achieving a relaxed version where the encoding map $\mathcal{C}\colon \Sigma^k \to \Sigma^n$ is allowed to be *randomized*, and where the input lists are generated from unions of codewords; we must succeed with high probability over the randomness in $\mathcal{C}$. In particular, our main result implies the following theorem.

Theorem 1.2 (informal; weaker than main result)**.** *For all $\ell > 0$, there is a randomized encoding map $\mathcal{C}\colon \Sigma^k \to \Sigma^n$ so that*

1. *$\mathcal{C}$ can be encoded in time $O(n)$;*
2. *The rate of $\mathcal{C}$, k/n, is a constant independent of ℓ and n;*

[1]In this paper we focus on *zero-error* list-recovery, which is the definition given here. Other works focus on the more general problem of list-recovery from errors, in which $\mathcal{C}(x)_i$ needs to be in $\mathcal{L}_i$ only for some fraction of the i-s.

3. *The alphabet size $|\Sigma|$ is polynomial in ℓ (and independent of n);*

4. *For any list $x^{(1)}, x^{(2)}, \ldots, x^{(\ell)} \in \Sigma^k$, there is an algorithm that runs in time $O(n\ell \operatorname{polylog}(\ell))$ that has the following guarantee. With probability at least $1 - o(1)$ over the randomness of $\mathcal{C}$, given the lists $\mathcal{L}_i = \{\mathcal{C}(x^{(j)})_i : j \in [\ell]\}$, the algorithm returns a list $\mathcal{L}$ so that $x^{(i)} \in \mathcal{L}$ for all i, and so that $|\mathcal{L}| = O(\ell)$.*

This statement is weaker than our main result because in fact our result still holds even if a random subset of the lists $\mathcal{L}_i$ in Item 4 are erased, and moreover the result still holds when some of the lists $\mathcal{L}_i$ in Item 4 contain some extra "distractor" symbols that occur according to any sufficiently "nice" distribution. We defer the formal statement of our list-recovery guarantee to Section 6.

Our code is essentially an *expander code* with *aggregated symbols*. That is, we begin with an expander code $\mathcal{C}_0 \colon \Sigma_0^k \to \Sigma_0^n$, as in [SS96], and we aggregate together the symbols as in [ABN+92]. (We discuss this construction in more detail below.) Our recovery algorithm uses ideas from previous algorithms, propagating information around the underlying expander graph given some advice. What makes our work different are the facts that (a) we leverage the randomness of $\mathcal{C}$ and a small failure probability, and (b) our underlying expander graph comes from a *high-dimensional expander.*[2] In particular, using the randomness in $\mathcal{C}$ we are able to obtain an algorithm with running time near-linear in ℓ, and using a high-dimensional expander we are able to boost our success probability to a level appropriate for an application to heavy hitters, which we discuss next.

Motivation from Heavy Hitters. One of the reasons we are interested in Goal 1.1 is because of the potential algorithmic applications of such a code. To illustrate this potential, we work out an application of our construction to the *heavy hitters* problem.

The set-up is as follows. We are given a stream of updates $(x^{(i)}, \Delta^{(i)})$, for $x^{(i)}$ in some universe $\mathcal{U}$ of size N, and $\Delta^{(i)} \in \mathbb{R}$. For all m, x, we assume that $f(x) \triangleq \sum_{j \in [m]} \Delta^{(j)} \cdot \mathbf{1}_{x^{(j)} = x} > 0$.[3] This is called the *strict turnstile model.* The goal is to maintain a small data structure (a "sketch") so that, after m (efficient) updates $(x^{(i)}, \Delta^{(i)})$, we can (efficiently) query the data structure to return a list of *ε-heavy hitters.* That is, we would like to recover a list $\mathcal{L}$ of size at most $O(1/\varepsilon)$ that contains all $x \in \mathcal{U}$ so that $f(x) \geq \varepsilon \cdot \|f\|_1 \triangleq \sum_{x \in \mathcal{U}} f(x)$.

The beautiful *Count-Min Sketch* (CMS) data structure of Cormode and Muthukrishnan [CM05] gives a solution to this problem. It uses (optimal) space $O(\varepsilon^{-1} \log N)$ and has update time $O(\log N)$. However, the query time to return all $O(1/\varepsilon)$ heavy hitters is large, $O(N \log N)$ (essentially, one performs a point query for each $x \in \mathcal{U}$ to see if it is a heavy hitter). The work [CM05] showed how to alleviate this with a so-called "dyadic trick," bringing the query time to $O(\log^2 N)$ at the cost of an extra $\log(N)$ factor in both the space and update time. (See Table 1 for a summary of the parameters in these and other works).

The starting point for our work is the work of Larsen, Nelson, Nguyễn and Thorup [LNNT16a]. That work studied a much more general problem—heavy hitters for all ℓ_p norms in the general turnstile model—but for the special case of the ℓ_1 norm and the strict turnstile model, they were

[2]We note that the construction of [DHK+19] is similar to ours, also using [ABN+92]-style symbol aggregation with a high-dimensional expander. However, in that work they have a more ambitious goal—list-decoding with no randomness in the encoder—but in return the parameters are not close to those in Goal 1.1.

[3]Here, we think of $f(x)$ as the "frequency" of item x. The Δ-s are updates: we may add or remove some quantity of each item x, provided that $f(x)$ never becomes negative.

Reference	Space	Update	Query	Failure probability
[CM05]	$O\left(\frac{\log N}{\varepsilon}\right)$	$O(\log N)$	$O(N \log N)$	N^{-c}
[CM05] ("dyadic trick")	$O\left(\frac{\log^2 N}{\varepsilon}\right)$	$O(\log^2 N)$	$O\left(\frac{\log^2 N}{\varepsilon}\right)$	N^{-c}
[LNNT16a]	$O\left(\frac{\log N}{\varepsilon}\right)$	$O(\log N)$	$O\left(\frac{\log^{1+\gamma} N}{\varepsilon}\right)$	N^{-c}
[LNW18]*	$\widetilde{O}\left(\log^2 N\right)$	$\widetilde{O}\left(\varepsilon \log^2 N\right)$	$\frac{1}{\varepsilon}\operatorname{poly}(\log N)$	N^{-c}
[CN20]*	$O\left(\frac{\log N}{\varepsilon}\right)$	$\widetilde{O}(\log N)$	$\frac{1}{\varepsilon}\operatorname{poly}(\log N)$	0
This work	$O\left(\frac{\log N}{\varepsilon}\right)$	$O(\log N)$	$O\left(\frac{\log N}{\varepsilon}\right)$	$N^{-\operatorname{poly}(\varepsilon)}$
This work	$O\left(\frac{\log N}{\varepsilon^c}\right)$	$O\left(\frac{\log N}{\varepsilon^c}\right)$	$O\left(\frac{\log N}{\varepsilon^c}\right)$	N^{-c}

Table 1: Relevant results on ε-heavy hitters in the strict turnstile model where the universe has size N, for $\log N \gg \operatorname{poly}(1/\varepsilon)$. We consider schemes with failure probability $\delta \geq 1/\operatorname{poly}(N)$; see the discussion in Section 1.3 for smaller failure probability where the works marked with $\star$ shine. The $\tilde{O}$ notation hides $\log\log(N)$ factors and $\log(1/\varepsilon)$ factors. Above, c is a constant independent of N and ε, and γ is any constant larger than 0. Unfortunately, the failure probability for our algorithm is only $N^{-\operatorname{poly}(\varepsilon)}$, rather that N^{-c} for some constant c. By repeating our algorithm $\operatorname{poly}(1/\varepsilon)$ times we can boost the success probability to N^{-c}. We note that each of Space, Update, Query time for [CM05] (with the dyadic trick) and [LNW18] can be multiplied by ε^c if one replaces the failure probability with $N^{-\varepsilon^c}$ and the results from [LNNT16a, Theorem 9] remain the same for that larger failure probability.

able to get a nearly optimal algorithm, with the same space and update time complexity as the original CMS, but with query time $O(\varepsilon^{-1}\log^{1+\gamma} N)$ for any constant $\gamma > 0$. That work highlighted a connection to list-recovery (see [LNNT16a, Section C]; a similar connection is also present in earlier works on group testing and compressed sensing, for example [INR10, NPR11, NPR12, GNP$^+$13, GLPS17]), which is one of our motivations to study Goal 1.1.

The approach of [LNNT16a] was the following (we have modified the description to be more explicitly coding-theoretic). To perform an update on an item $x \in \mathcal{U}$, encode it as $\mathcal{C}(x) \in \Sigma^n$ with our (randomized) encoding function. Then insert each symbol $\mathcal{C}(x)_j$ into n different ε-heavy hitters data structures that work on universe Σ (this could be a small CMS sketch, or something else). To query all of the heavy hitters, we first query each smaller data structure to find a list $\mathcal{L}_j$. Notice that since $|\Sigma| \ll |\mathcal{U}|$, it does not matter that the query algorithm for the small data structures is slow. Now, we do list-recovery on the lists $\mathcal{L}_j$ to recover a list $\mathcal{L}$ that contains all of the heavy hitters.[4]

However, as Goal 1.1 remains open, [LNNT16a] did not use a list-recoverable code to obtain their results. Instead, they (like us) took advantage of the fact that the lists $\mathcal{L}_j$ can be viewed as random variables over the randomness in the encoding map $\mathcal{C}$, and then use a construction based on "cluster-preserving clustering" to solve the problem. While in some sense this construction must be a list-recoverable code for randomized input lists, it is not clear (to us) how to extract a natural code out of it: the work [LNNT16a] took the perspective of graph clustering, rather than coding theory. In contrast, our code is very natural in the context of coding theory, as it is simply an expander code with aggregated symbols (albeit using a high-dimensional expander for the underlying graph).

[4]Provided that the output $\mathcal{L}$ of the list-recovery algorithm is not too large, we can use an additional large CMS data structure to efficiently do point queries on each item $x \in \mathcal{L}$, pruning it down to $O(1/\varepsilon)$.

As an example of the utility of our construction, we plug our randomized list-recoverable code (as in Theorem 1.2) into the framework of [LNNT16a]. This gives us an algorithm for heavy hitters that, in some parameter regimes, even slightly outperforms that of [LNNT16a]. When ε is constant and N is growing, we are able to improve the query time from $O(\log^{1+\gamma} N)$ to $O(\log N)$. In particular, we prove the following theorem. (See Table 1 for a comparison to other work when $\log N \gg \text{poly}(1/\varepsilon)$).

Theorem 1.3 (informal; see Theorem 5.11)**.** *There is a data structure that solves the heavy hitters problem in the strict turnstile model, that uses space $O(\varepsilon^{-1} \log N)$, update time $O(\log N)$, and query time $O(\varepsilon^{-1} \log N \, \text{polylog}(1/\varepsilon))$, with failure probability $\delta = N^{-\Theta(\varepsilon^3)}$, as long as $\varepsilon \geq (\log N)^{-\Omega(1)}$.*

By repeating this data structure $O(\varepsilon^{-3})$ times, we obtain a data structure that takes space $O(\varepsilon^{-4} \log N)$, update time $O(\varepsilon^{-3} \log N)$ and query time $O(\varepsilon^{-4} \log N \, \text{polylog}(1/\varepsilon))$, with failure probability $\delta = N^{-c}$.

Our algorithm has the added property that a successful $\mathcal{L}$ of size $O(1/\varepsilon)$ not only contains all the true heavy hitters, but also does not contain "false-positives", in the sense that each $x \in \mathcal{L}$ satisfies, say, $f(x) \geq \frac{\varepsilon}{4} \|f\|_1$. This property also applies to most previous heavy hitters algorithms.

Contributions. To summarize, our main contributions are the following.

1. **A code with probabilistic list-recovery.** We give a natural code construction that achieves a probabilistic version of Goal 1.1, as per Theorem 1.2. Our code construction leverages recent progress in high-dimensional expanders in order to succeed with high probability. We hope that our construction and techniques may be used in the the future to make further progress on Goal 1.1.

2. **Proof of concept: application to heavy hitters.** As an illustration of the utility of our construction—and as an proof-of-concept meant to encourage study of Goal 1.1—we obtain a new data structure for ε-heavy hitters in the strict turnstile model. Our data structure has slightly stronger guarantees than existing constructions for failure probability $1/\text{poly}(N)$ when ε is constant and the universe size N is growing.

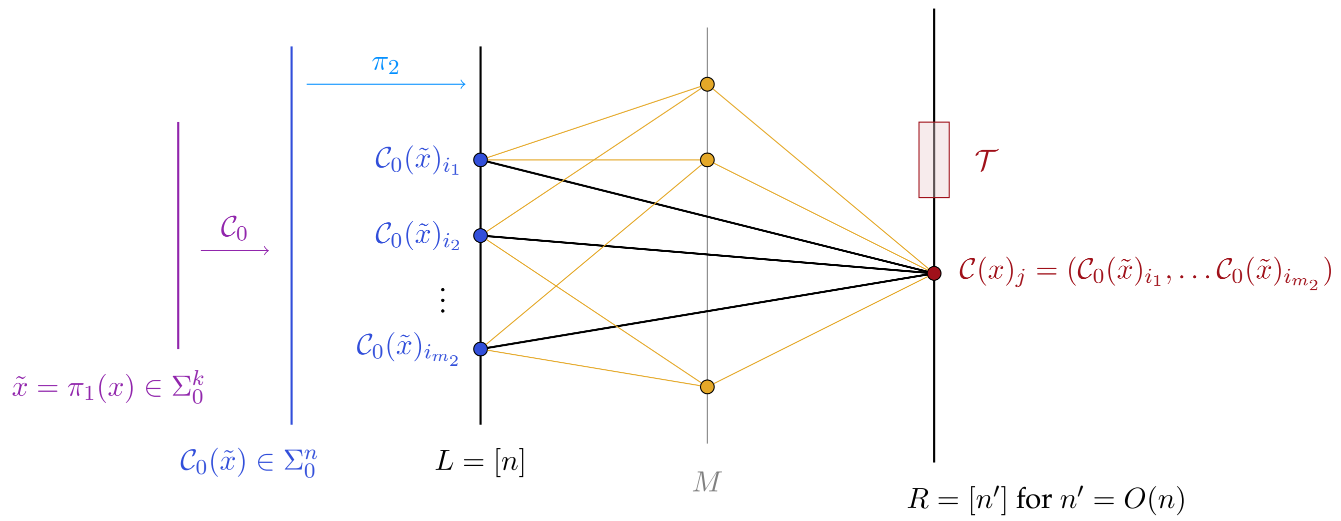
1.1 Construction Overview

In this section, we give a brief overview of our probabilistically list-recoverable code. We use this code to solve the ε-heavy-hitters problem following the paradigm described above, by using small heavy-hitters sketches for each symbol of the (randomized) encoding $\mathcal{C}(x)$ of $x \in \mathcal{U}$.

At a high level, we construct our code $\mathcal{C} \colon \Sigma_0^k \to \Sigma^{n'}$ as follows. We start with some base code $\mathcal{C}_0 \colon \Sigma_0^k \to \Sigma_0^n$, as well as a bipartite expander graph $G = (R, L, E)$, where $L = [n]$ and $R = [n']$, for some $n' = O(n)$.[5] We will need $\mathcal{C}_0$ and G to have specific properties, which we will come to below. For $x \in \Sigma_0^k$, we generate the encoding $\mathcal{C}(x)$ as follows. For $j \in [n']$, the encoded symbol $\mathcal{C}(x)_j$ will be gotten as the concatenation of the symbols $\mathcal{C}_0(x)_i$ for $i \in \Gamma_G(j)$, where $\Gamma_G(j)$ denotes the neighbors of j in the graph G. This sort of "aggregation along an expander" technique, introduced in [ABN+92], has become a standard distance amplification technique in error correcting codes. Because of the concatenation, our final alphabet Σ will be $\Sigma = \Sigma_0^{m_2}$.

To perform list recovery, we will start with a small piece of "advice," and then recover the (hopefully unique) message x consistent with that advice. We will generate our final list $\mathcal{L}$ by

[5]Using the notation of the technical sections below, $\Sigma_0 = \mathbb{F}_q$, $\Sigma = \mathbb{F}_q^{m_2}$ for some constant m_2, and $n' = |V_2| = O(n)$.



π_2
$\mathcal{C}_0$
$\mathcal{C}_0(\tilde{x})_{i_1}$
$\mathcal{C}_0(\tilde{x})_{i_2}$
$\vdots$
$\mathcal{C}_0(\tilde{x})_{i_{m_2}}$
$\mathcal{T}$
$\mathcal{C}(x)_j = (\mathcal{C}_0(\tilde{x})_{i_1}, \ldots \mathcal{C}_0(\tilde{x})_{i_{m_2}})$
$\tilde{x} = \pi_1(x) \in \Sigma_0^k$
$\mathcal{C}_0(\tilde{x}) \in \Sigma_0^n$
$L = [n]$
M
$R = [n']$ for $n' = O(n)$

Figure 1: Illustration of our construction. The coordinates of the inner code $\mathcal{C}_0$ live on the vertices of L. The final code $\mathcal{C}$ consists of symbols aggregated by vertices in R. The randomness in the encoding comes from the permuations π_1 and π_2, which scramble the messages in Σ^k and the coordinates in $[n]$, respectively. We use the vertices in $\mathcal{T} \subseteq R$ to define parity checks that partially define the code $\mathcal{C}_0$. The "middle layer" M is not used in the definition of the code, but is a necessary auxiliary structure for our recovery algorithm.

iterating over all possible values of the advice. Towards this end, we will choose some coordinate $j \in [n']$ for which $\mathcal{L}_j$ is not erased, and some $\sigma^\star \in \mathcal{L}_j$ as our guess for $\mathcal{C}_0(x)|_{\Gamma_G(j)}$ to act as our advice. Given this advice $\sigma^\star$, we wish to keep propagating information until we obtain enough coordinates of $\mathcal{C}_0$ that would allow us to uniquely determine x; this amounts to decoding the code $\mathcal{C}_0$ from erasures.

In the exposition below, we start with a naive attempt to do this propagation, and build up the properties that we will need $\mathcal{C}_0$ and G to satisfy as we refine it. Our construction is depicted in Figure 1.

A naive attempt. Our first attempt (which will not work) is the following. Let $j \in [n']$ be as above, so we assume that we are given as advice the m_2 symbols $\mathcal{C}_0(x)|_{\Gamma_G(j)}$; our goal is to recover (a hopefully unique) x given this advice and given the input lists $\mathcal{L}_{j'}$ for $j' \in [n']$. Choose some coordinate $j' \in [n']$ such that $\Gamma_G(j) \cap \Gamma_G(j') \neq \emptyset$. As we already know the symbols in the coordinates indexed by $\Gamma_G(j)$, this gives us partial information about $\mathcal{C}(x)_{j'}$ in the form of $|\Gamma_G(j) \cap \Gamma_G(j')|$ elements of Σ_0 in known locations. One can hope that this information would be enough to pinpoint a specific entry in the list $\mathcal{L}_{j'}$, allowing us to recover all symbols of $\mathcal{C}_0(x)$ in the coordinates indexed by $\Gamma_G(j')$, and keep going in the same manner until enough information is propagated.

Clearly, when we have no guarantee on the input lists $\mathcal{L}_i$, this approach fails miserably, as it may be the case that $\mathcal{L}_{j'}$ contains numerous elements in $\Sigma_0^{m_2}$ that agree in some of the m_2 locations, and the information coming from our advice for j will not uniquely pin down an element of $\mathcal{L}_{j'}$. However, note that for a *completely random* input list $\mathcal{L}_{j'}$, such an attempt would be successful with probability at least $1 - |\mathcal{L}_{j'}| / |\Sigma_0|$, and we could set the parameters in such a way that $|\mathcal{L}_{j'}| \ll |\Sigma_0|$. That is, in this case it would become reasonably likely that the choice of $\sigma^\star \in \mathcal{L}_j$ would uniquely pin down an element $\sigma \in \mathcal{L}_{j'}$, allowing us to propagate information to another vertex in the graph.

The hope is that we could propagate this information throughout the graph, using the fact that G is an expander to guarantee that most vertices will be determined. Of course, the problem with this is that we do not want to assume that the input lists are completely random, but this leads us to our next attempt, where we inject randomness into the encoding procedure.

Injecting randomness. While we won't get completely random lists $\mathcal{L}_j$ as we might have wanted for the naive attempt, we can make the input lists randomized via a randomized encoding. More specifically, our base code $\mathcal{C}_0$ will be deterministic, and to apply $\mathcal{C}$ we will make use of two permutations: a permutation π_1 acting on the universe $\mathcal{U}$ and a permutation π_2 acting on $[n]$. More formally, given $x \in \Sigma_0^k$, we first apply $\pi_1(x)$ and apply the encoding $\mathcal{C}_0$ to $\pi_1(x)$. Next, we permute the coordinates of the outcome according to π_2. Finally, we aggregate symbols according to G, yielding $\mathcal{C}(x) \in \Sigma^{n'}$. Roughly speaking, the first permutation—which will be pairwise independent—will make $\mathcal{C}_0(x)$ uniformly distributed over the code's image, even conditioned the value of $\mathcal{C}_0(x')$ for some $x' \neq x$. The second permutation will make sure that querying any particular symbol $\mathcal{C}_0(x)_j$ symbol will behave like sampling a uniformly random symbol in $\mathcal{C}_0(x)$, and even more strongly, combined with π_1 it will behave like a random sampling from a nearly uniform distribution over Σ_0.

Analyzing the permutation-aided construction carefully, we are able to show that indeed, with probability roughly $1 - \eta$ for $\eta \approx \left|\mathcal{L}_{j'}\right| / \sqrt{|\Sigma_0|}$, we can pinpoint a single list element of $\mathcal{L}_{j'}$. One conceptual observation that will help us establish that result is the fact that the distribution of symbols in most codewords of a high-rate code is close to uniform, and indeed we will need the rate of $\mathcal{C}_0$ to be very high (see Section 3.3). We leave the more technical details to Section 5.

Although promising, this approach is still problematic. We start with $m_2 = O(1)$ symbols that we know, and at each iteration the set of revealed coordinates grows by a small constant factor, using the expansion properties of G. As initially our sets are of constant size, we cannot hope for success probability much greater than $1 - \eta$ for the initial propagation steps. A failure probability of η, even if we disregard the need for a problematic union bound over all propagation steps, is far too large for us, and in particular for our application to heavy hitters. The problem described here is common to various expander-based techniques, and in this work we resolve it by choosing G to be a special expander graph that comes from a high-dimensional expander, and by choosing $\mathcal{C}_0$ to be a suitable *Tanner code*. We discuss these modifications next.

Using high-dimensional expanders to get a good head start. We resolve the issue described above—that we cannot possibly get a good failure probability if we start with only a few known symbols—by using techniques from high-dimensional expanders. Suppose that, starting with only the advice $\sigma^\star$ for m_2 symbols of $\mathcal{C}_0(x)$, we could deterministically identify find a large subset $\mathcal{T} \subseteq [n']$ for which we know all symbols of $\mathcal{C}_0(x)$ indexed by $\Gamma_G(\mathcal{T})$. This way, concentration bounds can kick in, and hopefully each propagation step would be successful with probability roughly $\eta^{|\mathcal{T}|}$, provided we can get a enough independence between query attempts at the same propagation step. We defer the independence issue to the technical sections (this ends up following from the amount of independence we have in our permutations π_1 and π_2), and concentrate on obtaining such a $\mathcal{T}$.

Recall that we work over the bipartite expander graph $G = (R = [n'], L = [n], E)$. We will construct G', a tripartite extension of G, with an added middle layer M, $|M| = O(n)$, having the following property. Identify each vertex j of R with a subset $\Gamma_G(j) \subset [n]$ of cardinally m_2

in the natural way. Each vertex in M is identified with a subset $S \subset [n]$ of cardinality m_1, for $1 < m_1 < m_2$, such that S is connected to all its m_1 elements on the left, and to all its supersets on the right. More specifically, each vertex j in R will be connected to all $\binom{m_2}{m_1}$ subsets of $\Gamma_G(j)$ in M. (See Figure 1 for an illustration.)

We will choose the code $\mathcal{C}_0$ to be a *Tanner code* with respect to the structure of the graph G. That is, as before, we associate the n symbols of a codeword $\mathcal{C}_0(x)$ with the left hand vertices L of G, and we define $\mathcal{C}_0$ so that a codeword $\mathcal{C}_0(x)$ is a labeling of L so that to following property holds: For every j in an appropriate subset $\mathcal{T} \subset R$, the labels on the vertices of $\Gamma_G(j)$ form a codeword in some error correcting code $\mathcal{C}_{00}$ of length m_2 with good distance; in particular, given any m_1 symbols of $\mathcal{C}_{00}(x')$ for some x', we can recover all of $\mathcal{C}_{00}(x')$.

The reason to choose $\mathcal{C}_0$ like this is the following. Say we know that j and j' are in the set $\mathcal{T}$, and that they have a common neighbor in M. This implies that $|\Gamma_G(j) \cap \Gamma_G(j')| \geq m_1$, since there is some set of size m_1 that both of those sets contain. In particular, by our choice of $\mathcal{C}_{00}$, once we know the symbols of $\Gamma_G(j)$, we can *deterministically* reveal all symbols of $\Gamma_G(j')$ by decoding $\mathcal{C}_{00}$. Then we can continue this process until we recover the symbols in $\Gamma_G(j)$ for all $j \in \mathcal{T}$. By counting constraints, it turns out that we can choose $\mathcal{T}$ to be large and still have a high-rate code $\mathcal{C}_0$. This gives us our set $\mathcal{T}$ so that we can deterministically fill in the symbols of $\Gamma_G(\mathcal{T})$ to use as a head start and increase our success probability.

How do we construct such a tripartite graph, that on the one hand has not too many vertices in R and M (i.e., $R = O(n)$ and $M = O(n)$), but on the other hand has favorable intersection and expansion properties? This is where *high-dimensional expanders* enter the picture, and indeed the tripartite graph comes from an $(m_2 - 1)$-dimensional simplicial complex (see Section 3.1 for the formal definitions). A similar object was used in [DHK+19] as a *double sampler*, and in [DDHRZ20] as a *multilayer agreement sampler*. We note that the construction of [DHK+19] is quite similar to ours, as they also use the symbol-aggregation technique of [ABN+92]; the main difference in the construction is that we use a very specific inner code $\mathcal{C}_0$ that uses the structure of G as part of its parity checks, while the work of [DHK+19] chooses $\mathcal{C}_0$ to be an arbitrary code with good distance.

In our actual construction, the code $\mathcal{C}_0$ is a bit more involved, and its constraints arise both from the special subset $\mathcal{T}$ of R and from an *additional* bipartite expander. Each of the two types of constraints is helpful for a different aspect of our algorithm. Roughly speaking, the constraints that come from $\mathcal{T} \subseteq R$ help us as described above (filling in the set $\Gamma_G(\mathcal{T})$ to get a head start). The other constraints are there to ensure that the final code $\mathcal{C}_0$ has good enough distance to allow for the final unique decoding. All in all, we are able to achieve a set $\mathcal{T}$ that has size about $|\mathcal{T}| \approx \mathrm{poly}(\varepsilon) \cdot n$. We remark that this is the point where we don't quite get the failure probability that we want, resulting in a sub-optimal dependence on ε for our application to heavy hitters: we want failure probability $\exp(-n)$ (we will choose n logarithmic in N, so this would be $\mathrm{poly}(1/N)$), and we end up with failure probability $\exp(-|\mathcal{T}|) = \exp(-\mathrm{poly}(\varepsilon)n)$.

There are plenty of details that are swept under the rug in the description above, including implementation details needed to keep the recovery algorithm linear-time. We give the recovery algorithm in detail in Section 4. We present our list-recovery algorithm in the context of a query algorithm for heavy hitters, since for our analysis we want to focus on the distribution of input lists that arises from the heavy hitters example, and it is easiest to present everything together. In particular, the input lists do not arise simply from the union of ℓ codewords $\mathcal{C}(x)$, but (a) may be erased if the corresponding small data structure failed, and (b) may contain extraneous symbols that arise from items $x^{(i)}$ that appear in the stream that are not heavy hitters.

1.2 Motivating Goal 1.1 from Pseudorandomness

In this section, we briefly explain why Goal 1.1—and in particular, getting a good dependence on the parameter ℓ—is of interest in pseudorandomness. There is a tight connection between error correcting codes and fundamental constructions in pseudorandomness, notably the equivalence between (strong) seeded extractors and list-decodable codes [Tre01, TSZ04]. It turns out that list recovery can also play a prominent role in the study of related objects from extractor theory. In *seeded condensers*, first studied in [RR99], the goal is to "improve" the quality of a random source $\boldsymbol{X}$ using few additional random bits. A bit more formally, given a random variable $\boldsymbol{X} \sim \{0,1\}^n$ with min-entropy k, a condenser $\mathsf{Cond}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is such that $\mathsf{Cond}(\boldsymbol{X}, U_d)$ has min-entropy k', where we want the entropy rate to improve, namely, $\frac{k'}{m} \gg \frac{k}{n}$, and to maintain a small entropy gap $m - k'$. (For the formal definition, see, e.g., [GUV09].) List recoverable codes in the *errors* model[6] give seeded condensers, and vice versa. More specifically, the input and output entropies k and k' are almost in one-to-one correspondence with the (logarithm of the) output and input list sizes, $\log|\mathcal{L}|$ and $\log \ell$ (for the precise statement, see [DMOZ20]). Thus, to get meaningful condensers from list-recoverable codes, the dependence between L and ℓ needs to be good, in all regime of parameters, and in particular handle ℓ that grows arbitrarily with the message length. In fact, the best list-recoverable code in this regime is the (folded) Parvaresh-Vardy code [GUV09], giving $|\mathcal{L}| \approx \ell$.[7] The connection between condensers and list-recoverable codes was recently utilized in the *computational* setting to construct nearly-optimal pseudorandom generators for polynomial-sized circuits [DMOZ20].

The model of *zero-error list recovery*, described in Goal 1.1 (when $|\mathcal{L}|$ depends nicely on ℓ and ℓ can be arbitrary), has applications to pseudorandomness too. A (strong) *disperser* is a function $\mathsf{Disp}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ such that for any random variable $\boldsymbol{X} \sim \{0,1\}^n$ with sufficient min-entropy, the support of $\mathsf{Disp}(\boldsymbol{X}, U_d)$ is large. Such dispersers have found several applications, and are tightly connected to open problems in expander graphs. It is not hard to show, and we do so in Appendix B, that dispersers, in some parameter regime, are equivalent to zero-error list-recoverable codes. We are not aware of this equivalence being stated elsewhere. For completeness, we note that dispersers in another parameter regime give rise to erasure list-decodable codes [BADTS20].

Finally, observe that in order to get good pseudorandomness primitives from list-recoverable codes, efficient recovery is not an issue, and all that is needed is an efficient *encoding*.

Even though a probabilistic guarantee as in Theorem 1.2 does not immediately yield improved pseudorandom objects, it is our hope that our progress on Goal 1.1 is a first step towards achieving that goal, which would imply improved dispersers.

1.3 Related Work

Algorithmic List-Recovery. List-recovery was originally introduced as an avenue towards *list-decoding*, where the goal is, given a vector $z \in \Sigma^n$, to recover the list $\mathcal{L}$ of all messages $x \in \Sigma^k$ so that $\mathcal{C}(x)$ is sufficiently close to z in Hamming distance. For example, the celebrated list-decoding algorithm of Guruswami and Sudan for Reed-Solomon codes [GS98] is in fact a list-recovery algorithm. However, the Guruswami-Sudan algorithm stops working at the so-called *Johnson bound,*

[6] In the errors model, we are given $\mathcal{L}_1, \ldots, \mathcal{L}_n \subseteq \Sigma$ with $|\mathcal{L}_i| \le \ell$ for all i, and we require the list $\mathcal{L} = \{x \in \Sigma^k : \Pr_{i\in[n]}[\mathcal{C}_i(x) \in \mathcal{L}_i] \ge 1-\gamma\}$ to be small, for some error parameter γ.

[7] Note, however, that the rate of the code in [GUV09] is only $k^{-\Omega(1)}$ for k being the message length.

which in the context of list-recovery means that the rate k/n of the code can be at most $1/\ell$. Since the Guruswami-Sudan algorithm, there has been a great deal of work, mostly based on algebraic constructions, aimed at surpassing the Johnson bound for list-decoding and list-recovery. In particular, the works [GR08, GW13, Kop15, KRZSW18] show variations of Reed-Solomon codes, like folded RS codes and multiplicity codes, can be efficiently list-decoded and list-recovered beyond the Johnson bound. For list-recovery, these constructions are able to obtain rate $k/n = \Omega(1)$, but unfortunately the size of the lists $\mathcal{L}$ returned (and in particular the running time of the algorithm that returns that list) is at least quasipolynomial in ℓ [GR08, KRZSW18], and sometimes exponential in ℓ. Moreover, those constructions naturally have large alphabet sizes, polynomial in n. In order to reduce the alphabet size, constructions using algebraic geometry codes have been used (e.g. [GX12, GX13, GK16]), although these works still have parameters with an exponential dependence on ℓ. Moreover, all of the works mentioned above have polynomial—and not linear—time recovery algorithms. Using expander-based techniques (e.g. that of [AEL95]), these algorithms can be improved to near-linear time in n (e.g., [HRZW19]), but at the cost of increasing the dependence on ℓ to doubly-exponential.

In addition to algebraic constructions, there have also been a few constructions of purely graph-based codes, which are more similar to our constructions. The work of [GI04] gives a linear-time algorithm for list-recovery of graph-based codes, which does even better in the setting of *mixture-recovery* (similar to the setting that we study here) where the input lists are generated from unions of codewords. That work achieves output list size $|\mathcal{L}|$ exactly equal to ℓ, but has rate $O(1/\ell)$ and the alphabet size is exponential in ℓ. The work of [HW18] gives an $O(n)$-time algorithm for list-recovering graph-based codes (the expander codes of [SS96, Zém01], with an appropriate inner code); these can have high rate (close to 1), but unfortunately the dependence on ℓ in other parameters is *quadruply*-exponential.

The work of Dinur, Harsha, Kaufman, Livni-Navon and Ta-Shma [DHK$^+$19], which directly inspired our work, used *double-samplers* derived from high-dimensional expanders, combined with an expander-based symbol aggregation technique of [ABN$^+$92] that we also use. The goal of that work was to give an efficient list-decoding algorithm for *any* code that follows the [ABN$^+$92] construction. This is much more general that what we are aiming to do (since we get to carefully design our code before applying the [ABN$^+$92] construction), and also the goal is different (list-decoding in the worst case, rather than randomized list-recovery). That work is able to get efficient (polynomial-time) algorithms, but when one tries to turn their algorithm into a list-recovery algorithm in the most direct way, the parameters are not close to those in Goal 1.1; in particular, the algorithm is only $\mathrm{poly}(n)$-time, and the dependence on ℓ is again exponential. It is not clear (to us) how to use the approach of [DHK$^+$19] to achieve Goal 1.1.

We also mention a recent work of [DDHRZ20] that suggests an approach for constructing locally testable codes. In particular, as in our construction they also use the underlying graph (an *agreement expander* coming from a high-dimensional expander) both for symbol manipulation and for defining the parity checks. However, their goal is quite different than ours: they obtain locally testable codes via lifting a set of "smaller" locally testable codes, extending the natural Tanner tests.

Heavy Hitters. The first work with provable guarantees for the heavy hitters problem was [MG82], which applied to the *cash register* model where each of the updates $\Delta^{(i)}$ are equal to 1. We work in the more general strict turnstile model described above. For the strict turnstile model, the Count-

Min Sketch data structure of [CM05] above already gets good results, and the best current results for the parameter regime we are motivated by (in particular, with failure probability $1/\operatorname{poly}(N)$, and where $\log N \gg \operatorname{poly}(1/\varepsilon)$) are those of [LNNT16a] described above. It is known [JST11] that $\Omega(\varepsilon^{-1}\log N)$ words of memory are required for this setting, and thus the space used by these works are optimal.

Two recent works studied heavy hitters in the strict turnstile model when the failure probability is extremely small (or zero). In [LNW18], the authors modify the Count-Min Sketch by looking at different hash functions, and they present a data structure with failure probability δ with space $\widetilde{O}\left(\log(\varepsilon N)\left(\varepsilon^{-1} + \log(1/\delta)\right)\right)$, update time $\widetilde{O}(\log^2(1/\varepsilon)\log(\varepsilon N)\,(1+\varepsilon\log(1/\delta))$, and query time $\widetilde{O}(\varepsilon^{-1}\log^2(1/\varepsilon)\log(\varepsilon N)\log(1/\delta))$. For $\delta = N^{-c}$ and $\log N \gg \operatorname{poly}(1/\varepsilon)$, this gives the parameters stated in Table 1. However, when δ is much smaller—for example, $\delta = N^{\Omega(1/\varepsilon)}$—this gives better results that the works previously discussed, and in particular implies a result that is *uniform* over all sets of heavy hitters by union bounding over the $N^{O(1/\varepsilon)}$ choices for such sets. In [CN20], the authors give a randomized construction of a data structure that also solves the heavy hitters problem uniformly over all streams $x^{(1)}, x^{(2)}, \ldots$ (that is, with error probability *zero* assuming that the data structure was constructed correctly). This scheme uses space $O(\varepsilon^{-1}\log(N\varepsilon))$, has update time $\tilde{O}(\log^2(1/\varepsilon)\log(\varepsilon N))$, and query time $\varepsilon^{-1}\operatorname{polylog}(N)$.[8] That work actually provides solutions to several problems, not just heavy hitters, via a construction of *list-disjunct matrices.*

We note that there are algorithms that achieve $O(\log N)$ update and query time for constant ε, but with only a constant failure probability. For example, such an algorithm is given in the full version of [LNNT16a] (see [LNNT16b, Theorem 10]).

One can generalize to the *general turnstile* model, where there is no guarantee that $f(x)$ is positive at each point in the stream, and one can generalize to ℓ_p-heavy hitters, where the goal is to return all x so that $|f(x)| \geq \varepsilon\|f\|_p$. There has been a great deal of work along both of these lines; see [LNNT16a] and the references therein. In particular, for ℓ_p heavy hitters in the general turnstile model, the work [LNNT16a] gives a data structure with space $O(\varepsilon^{-p}\log N)$, update time $O(\log N)$, and query complexity $\varepsilon^{-p}\operatorname{polylog}(n)$.

We briefly discuss the approach of [LNNT16a], in order to illustrate the differences between their approach and ours. While that work inspired the list-recovery approach we take, and they also use error correcting codes and expander graphs, the construction itself is quite different. That work takes the perspective of *graph clustering.* In more detail, their sketch can output a graph in which each heavy hitter is represented by a well-connected cluster in the graph. They then develop a clustering algorithm that can recover the clusters, and hence the heavy hitters. In order to make the connection to graph clustering, they first encode x with an error correcting code $\mathcal{C}_0$ as we do; but they only need this code to have good distance, as they do not go down the list-recovery route. Then they break $\mathcal{C}_0(x)$ up into n' chunks. Before putting the j-th of these chunks into the j-th smaller data structure, they append it with tags $h_j(x)$ and $\{h_{\Gamma(j)_i}(x)\}$, where the h_j are hash functions and Γ is the adjacency function for an expander graph G. Thus, the j-th chunk of $\mathcal{C}_0(x)$ is essentially connected by edges in G to the other chunks of $\mathcal{C}_0(x)$, and in particular the chunks of $\mathcal{C}_0(x)$ form a cluster that can be recovered by a clustering algorithm.

We note that [BNS19] was also inspired by [LNNT16a], and builds on their approach to develop differently private heavy-hitters algorithms. In fact, that work even casts the scheme of

[8]We note that here the guarantee is to return a list $\mathcal{L}$ of size $O(1/\varepsilon)$ containing all the true heavy hitters, although in both [LNW18] and [CN20], the list is allowed to contain elements with frequencies $f(x) \ll \varepsilon\|f\|_1$, while most of the heavy-hitters work surveyed above, including ours, does not have such false positives.

[LNNT16a] as a list-recovery scheme, in a relaxed definition of list-recovery that is different from our relaxed version in Theorem 1.2. In particular, their notion of list-recovery will not handle input lists $\mathcal{L}_1, \dots, \mathcal{L}_n$ that are generated by any ℓ distinct messages, as we handle in Theorem 1.2.[9]

Algorithmic applications of list-recovery. Our work is inspired by the use of list-recovery in [LNNT16a], but there is a rich history of using list-recoverable codes in similar algorithmic applications. One example is *group testing*, where the goal is to identify d "positive" items out of a universe of size N, given tests of the form $\bigvee_{i \in I} \mathbf{1}[i \text{ is positive}]$ for subsets $I \subset [N]$. A classic construction of Kautz and Singleton [KS64] reduces this question to the question of list-recovery. This connection, and elaborations on it, has been exploited in several works, which aim to both minimize the number of tests and to develop sublinear-time algorithms to recover the set of positive items [INR10, NPR11].

A second example, even closer to our work, is in *compressed sensing*. In compressed sensing, the goal is to approximately recover an approximately sparse vector $v \in \mathbb{R}^N$ given linear measurements Av for some $A \in \mathbb{R}^{t \times N}$. The heavy-hitters problem is closely related, as a (linear) solution to the heavy hitters problem can approximately recover the support of v. List-recoverable codes have been used in the context of compressed sensing in a similar way as it was used in [LNNT16a]: associate each $i \in [N]$ with a message, and encode it with a list-recoverable code to get a codeword $\mathcal{C}(i) = (c_1, \dots, c_n) \in \Sigma^n$. Then reduce the compressed sensing problem to n smaller instances of the same problem for vectors of length $|\Sigma|$: for each $j \in [n]$, we have a vector $w^{(j)}$ indexed by Σ so that the entry $w^{(j)}_\sigma$ is obtained by aggregating all of the coordinates v_i of v so that $\mathcal{C}(i)_j = \sigma$. Now we can either recurse or solve these smaller problems in another way. Previous works [NPR12, GNP$^+$13, GLPS17] have observed that a good list-recoverable code (e.g., satisfying Goal 1.1) would solve this problem. However, they ran into the same issue that we did, namely that we do not know of any such codes. Instead, they either used sub-optimal codes or developed work-arounds, as we describe below.

The work of [NPR12] was, to the best of our knowledge, the first to apply list-recovery in compressed sensing. We mention two results in that work that use a framework quite similar to that of [LNNT16a] (and thus to ours), making explicit use of (sub-optimal) list-recoverable codes. The first result is based on the list-recoverability of Reed-Solomon codes. As RS codes do not achieve Goal 1.1, this results in a sub-optimal number of measurements, but is nice and simple. The second is based on Parvaresh-Vardy (PV) codes. PV codes have good rate and output list-size, but unfortunately the alphabet size is very large. To get around this, [NPR12] (inspired by the work [NPR11] on group testing mentioned above) considered a code constructed by repeatedly concatenating PV codes with themselves. This does not lead to a code that achieves Goal 1.1—the rate depends on ℓ, and either the alphabet size or the rate must depend on n—but they are able to make these dependencies not too bad. This leads to schemes with near-optimal number of measurements t, although the schemes only work for non-negative signals. Further, since PV

[9]In a bit more detail, the notion of list-recovery in [BNS19] allows for $\mathcal{L}_1, \dots, \mathcal{L}_n$ to be generated by ℓ messages $x^{(1)}, \dots, x^{(\ell)}$, *provided* that the messages lie in distinct "buckets," according to any fixed bucketing of the message space. The choice of the code may depend on the bucketing. In this language, the result of [LNNT16a] (see [BNS19, Theorem 3.6]) says that it is possible to obtain a code with constant rate, output list size $L = O(\ell)$, and alphabet size that is polynomial in the number of buckets, and a polynomial-time decoding algorithm. If the number of buckets is $\mathrm{poly}(\ell)$, the alphabet size is also $\mathrm{poly}(\ell)$, as we would hope, but as the number of buckets grows (to approach the general case with $|\Sigma|^k$ buckets of size one, where there is no "bucketing" restriction) the alphabet size grows accordingly.

codes do not have near-linear-time algorithms, the recovery algorithm runs in time $\mathrm{poly}(t)$ rather than near-linear in t.

The work of [GNP$^+$13] follows a similar outline, using the Loomis-Whitney-based codes of [NPRR18]. These are codes $\mathcal{C}\colon [N] \to [N^{1/d}]^{d-1}$ are $(\ell, \ell^{d/(d-1)})$-list-recoverable in time $O(\ell^{d/(d-1)} \log N)$, where $d > 0$ is some integer parameter. In terms of the desiderata of Goal 1.1, this does give near-linear-time recovery with good dependence on ℓ; however the alphabet size is huge, growing exponentially in the message length. In [GNP$^+$13], they deal with this by applying the scheme mentioned above recursively until the alphabet size becomes manageable. As a result, they are able to get a nearly optimal number of measurements, with a recovery time that depends polynomially (but not linearly) on $\log N$, and with an extremely small error probability, smaller than $1/\mathrm{poly}(N)$.

We also mention [GLPS17], which uses list-recoverable codes (PV codes) in a more complicated way to achieve a near-optimal compressed sensing algorithm in the uniform ("forall") model. They also treat the indices i as messages and encode them with a list-recoverable code, but they develop more machinery—using an expander to add linking information between the symbols for example—in order to reduce to the list-recovery problem.

1.4 Open Questions and Future Work

In this work we have made progress towards Goal 1.1 by constructing a *randomized* code that supports, with high probability, linear-time list-recovery from certain lists. This was enough for our application to heavy hitters, but many open questions still remain.

1. The most obvious open question is to fully attain Goal 1.1. In addition to furthering our knowledge in algorithmic coding theory, it seems likely that attaining Goal 1.1 (or the techniques used to do it), would have other applications in algorithm design, as well as in pseudorandomness (as per Section 1.2).

2. While we are able to use techniques from high-dimensional expansion to obtain a failure probability of $N^{-\Omega(\varepsilon^3)}$ (in the setting of ε-heavy hitters), we would like a failure probability of $N^{-\Omega(1)}$.

3. In this paper we studied only *zero-error* list-recovery (or, more accurately, list-recovery from a small fraction of erasures). While this question is interesting and challenging on its own, one can ask about extending our results to list-recovery from errors. In particular, this might lead to improved heavy-hitters schemes in the general turnstile model.

4. We motivated our "probabilistic list-recovery" model by an application to heavy hitters. However, we hope that there are many other algorithmic applications for such a model and for our construction. Indeed, there are several algorithmic applications of list-recovery mentioned in Section 1.3 (e.g., [INR10, NPR11, NPR12, GNP$^+$13]) that explicitly use list-recoverable codes and would be improved by codes that achieve Goal 1.1. It is our hope that some of these applications could also be improved by better constructions of the probabilistically list-recoverable codes that we study here. As one example, if one could obtain Theorem 1.2 with $|\Sigma| = \widetilde{O}(\ell)$ (rather than polynomial in ℓ), then by the construction of Kautz

and Singleton mentioned above [KS64] this would yield optimal constructions of probabilistic group testing matrices with sublinear-time decoding, matching a recent result of [PS20] in a black-box way.

1.5 Organization

In Section 2, we set notation, introduce necessary definitions from error correcting codes and formally introduce the heavy hitters problem, and we state a few claims we will need about the pairwise independent permutations, expander graphs, and concentration bounds.

In Section 3 we describe the code $\mathcal{C}$ that we will use. As discussed above, we construct $\mathcal{C}$ by aggregating symbols of an inner code $\mathcal{C}_0$, according to a high-dimensional expander. We introduce the tools from high-dimensional expanders that we need in Section 3.1, and then we discuss the code $\mathcal{C}_0$ in Section 3.2. In Section 3.3, we show that $\mathcal{C}_0$ (and indeed, any code with high enough rate) will have the property that for a random x, $\mathcal{C}_0(x)$ will have a roughly uniform distribution of symbols, an important property that we will use in the analysis. In Section 3.4, we finally define our code $\mathcal{C}$ by aggregating symbols of $\mathcal{C}_0$ according to a high-dimensional expander.

We work out the application to heavy hitters in Section 4. In particular, we describe the data structure and update procedure sketched above in the beginning of Section 4 and in Section 4.2. We describe the query procedure—which implicitly describes the efficient list-recovery algorithm of the informal Theorem 1.2—in Section 4.3.

In Section 5, we show that our query procedure/list-recovery algorithm is actually correct, given input lists that are generated by a heavy hitters instance. In particular, the informal Theorem 1.2 on list-recovery follows from this analysis when the frequency vector f is exactly ℓ-sparse with $f(x) \in \{0,1\}$, and Theorem 1.3 about the more general heavy hitters problem follows from Theorem 5.11, our main result in Section 5. In Section 6, we abstract out our formal theorem about probabilistic list-recovery that generalizes Theorem 1.2.

2 Preliminaries

For a positive integer n, we denote by $[n]$ the set $\{1,\dots,n\}$. The density of a set $B \subseteq A$ is $\rho(B) = \frac{|B|}{|A|}$. We use base 2 for logarithms unless stated otherwise. The *statistical distance* between two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ over the same domain Ω is defined as

$$|\boldsymbol{X} - \boldsymbol{Y}| = \max_{A \subseteq \Omega} \left(\Pr[\boldsymbol{X} \in A] - \Pr[\boldsymbol{Y} \in A]\right).$$

If $|\boldsymbol{X} - \boldsymbol{Y}| \le \varepsilon$ we say that $\boldsymbol{X}$ is ε-close to $\boldsymbol{Y}$ and denote it by $\boldsymbol{X} \approx_\varepsilon Y$. We denote by U_n the random variable distributed uniformly over $\{0,1\}^n$. For a set A, we denote by U_A the random variable distributed uniformly over the elements of A.

For a vector $f \in \mathbb{R}^n$, we let $\|f\|_p$ to be its induced ℓ_p norm, i.e., $\|f\|_p = (\sum_{i\in[n]} |f[i]|^p)^{1/p}$, and $\|f\|_\infty = \max_{i\in[n]} |f[i]|$. We let $\|f\|_0$ be the support size of f, i.e., $|\{i \in [n] : f[i] \neq 0\}|$. We will often identify f with the function $f\colon [n] \to \mathbb{R}$ in the natural way (where $[n]$ can be replaced by an arbitrary domain).

For a bipartite graph $G = (R, L, E)$ and $v \in R$, we denote by $\Gamma_G(v)$ the set of vertices in L that are adjacent to v. For a set $A \subseteq R$, we denote $\Gamma_G(A) = \bigcup_{v\in A} \Gamma_G(v)$. We sometimes treat $\Gamma_G(v)$ as a vector, and then we require some fixed ordering of E. As it will be clear from context, we also treat

$\Gamma_G(\cdot)$ as the right-neighborhood function, namely mapping $\Gamma_G(u)$ for $u \in L$ to the its neighbors in R.

2.1 Error Correcting Codes

A linear code $\mathcal{C}$ of message length k and block length n over $\mathbb{F}_q$ is a linear subspace of $\mathbb{F}_q^n$ of dimension k. We will often identify a linear code with its encoding function, $\mathcal{C}\colon \mathbb{F}_q^k \to \mathbb{F}_q^n$, and often identify a codeword $c \in \mathcal{C}$ with a function $c\colon [n] \to \mathbb{F}_q$ in the natural way. The *rate* of $\mathcal{C}$ is $r = \frac{k}{n}$, and its distance is

$$d = \min_{x,y\in\mathcal{C}, x\neq y} \Delta(x,y) = \min_{x\in\mathcal{C}, x\neq \mathbf{0}} \Delta(x,\mathbf{0}),$$

for Δ being the Hamming distance (the latter equality only holds when $\mathcal{C}$ is linear). The code's *relative distance* is $\delta = \frac{d}{n}$.

A code is *uniquely decodable* from e erasures (or, $\frac{e}{n}$-fraction of erasures) if given $x \in (\mathbb{F}_q \cup \{\perp\})^n$ with at most e coordinates $i \in [n]$ in which $x[i] = \perp$, there is at most a single $c \in \mathcal{C}$ such that $x[i] = c[i]$ whenever $x[i] \neq \perp$. The *Reed-Solomon* code over $\mathbb{F}_q$ of block length $n \le q$ and dimension k is the subspace of all degree-$(k-1)$ univariate polynomials over $\mathbb{F}_q$. The Reed-Solomon code has distance $d_{\mathsf{RS}} = n-k+1$, and as any code, is uniquely decodable from $d_{\mathsf{RS}} - 1$ erasures.

We will make use of *Tanner codes* [Tan81]. Given a biregular bipartite graph $G = (R, L = [n], E)$ with right-degree D, and an *inner code* $\mathcal{C}_0 \subseteq \Sigma^D$, we define the corresponding tanner code by

$$\mathcal{C} = \left\{ c \in \Sigma^n : \forall r \in R, c|_{\Gamma_G(r)} \in \mathcal{C}_0 \right\},$$

where we choose an ordering of the edges at each vertex of R. Note that $\mathcal{C}$ is linear if $\mathcal{C}_0$ is a linear code.

As discussed above, Our work is inspired by the problem of list recovery from erasures. Formally, a code $\mathcal{C} \subseteq \Sigma^n$ is (γ, ℓ, L) *list recoverable from erasures* if for every $S_1, \ldots, S_n \subseteq \Sigma$ such that $|S_i| \le \ell$ for at least $(1-\gamma)n$ of the i-s and $S_i = \Sigma$ for the remaining, there are at most L codewords $c \in \mathcal{C}$ so that $c \in S_1 \times \ldots \times S_n$.

2.2 The Heavy Hitters Problem

In this work we consider the ε-heavy hitters problem, in the ℓ_1 norm. Set some threshold parameter $\varepsilon > 0$, and a universe $\mathcal{U}$ of size N. We see a stream of updates $(x^{(i)}, \Delta^{(i)}) \in \mathcal{U} \times \mathbb{R}$ for $i = 1, \ldots, m$, and we want to maintain $f \in \mathbb{R}^N$ for $f(x) = \sum_{i\in[m]} \Delta^{(i)} \mathbf{1}_{x^{(i)}=x}$ in a way that supports updates and one allowed query, as follows.

Update Given $x \in \mathcal{U}$ and $\Delta \in \mathbb{R}$ with finite precision, update $f(x) \leftarrow f(x) + \Delta$.

ε-HH Query Return a list of all *ε-heavy hitters*: All $x \in \mathcal{U}$ that satisfy $|f(x)| \ge \varepsilon \|f\|_1$. Note that there are at most $1/\varepsilon$ such elements.

We may also require our data structure to support a **1-query**. I.e., returning an estimation of $f(x)$ given any $x \in \mathcal{U}$. We work in the *strict turnstile* model, meaning that each update Δ may be an arbitrary number, but we are promised that $f(x) \ge 0$ for all $x \in \mathcal{U}$ at every step of the stream.

As in previous works, we work in the RAM model. Each machine word can store integers up to $\max\{N, \|f\|_1\}$. Standard word operations take constant time. We count *space* in terms of the number of words stored, and *time* in terms of the number of word operations. Moreover, we assume standard field operations take constant time. In particular, we will perform arithmetic in $\mathbb{F}_N$ and $\mathbb{F}_q$, for $q = q(\varepsilon)$ to be set later on.

We will rely on the following sketches-based ε-heavy hitters data structures.

Theorem 2.1 (Count Min Sketch, [CM05]). *For any $\varepsilon, \delta > 0$ and positive integer N, there exists an algorithm that maintains $f \in \mathbb{R}^N$ in the strict turnstile model and supports the following procedures using space $O(\frac{1}{\varepsilon} \log \frac{1}{\delta})$.*

- *An update, which is done in time $O(\log \frac{1}{\delta})$.*
- *A 1-query, which is done in time $O(\log \frac{1}{\delta})$ with accuracy ε and failure probability δ. More specifically, given $x \in [N]$, the 1-query procedure outputs $\hat{f}(x)$ such that $\hat{f}(x) \geq f(x)$ (with probability 1), and with probability at least $1 - \delta$ it holds that $\hat{f}(x) \leq f(x) + \frac{\varepsilon}{4}\|f\|_1$.*

Outputting the ε-heavy hitters using Theorem 2.1 would take $O(N \log \frac{1}{\delta})$ time, and one should think of $\delta = \frac{1}{\mathrm{poly}(N)}$. A significantly better running time was achieved by Cormode and Muthukrishnan using the "dyadic trick".

Theorem 2.2 (Count Min Sketch with the dyadic trick, [CM05]). *For any $\varepsilon, \delta > 0$ where $\varepsilon \geq \delta$ and $\delta \leq \frac{1}{\log N}$, and positive integer N, there exists an algorithm that maintains $f \in \mathbb{R}^N$ in the strict turnstile model and supports the following procedures using space $O(\frac{1}{\varepsilon} \log \frac{1}{\delta} \cdot \log N)$.*

- *An update, which is done in time $O(\log \frac{1}{\delta} \cdot \log N)$.*
- *An ε-HH query, which is done in time $O(\frac{1}{\varepsilon} \log \frac{1}{\delta} \cdot \log N)$ with failure probability δ. More specifically, the query procedure outputs a list $\mathcal{L} \subseteq [N]$ such that $\{x \in [N] : |f(x)| \geq \varepsilon\|f\|_1\} \subseteq \mathcal{L}$ (with probability 1), and with probability at least $1 - \delta$ it holds that $\mathcal{L} \subseteq \left\{x \in [N] : |f(x)| \geq \frac{\varepsilon}{4}\|f\|_1\right\}$. That is, $\mathcal{L}$ always contains all ε-heavy hitters and with probability at least $1 - \delta$, it does not contain any elements which are not $\frac{\varepsilon}{4}$-heavy hitters.*

For general ε and δ, the $\log \frac{1}{\delta}$ factors should be replaced with $\log\left(\frac{\log(\varepsilon N)}{\varepsilon\delta}\right)$.

Larsen et al. succeeded in getting optimal space requirement and update time, and nearly optimal query time. For simplicity, we state their result for $\delta = \frac{1}{\mathrm{poly}(N)}$.

Theorem 2.3 ([LNNT16a], Theorem 9). *For any $\varepsilon \in (0, 1/2)$, sufficiently large positive integer N, $\delta \in \left(\frac{1}{\mathrm{poly}(N)}, 1/2\right)$, and a constant $\gamma \in (0, 1/2)$, there exists an algorithm that maintains $f \in \mathbb{R}^N$ in the strict turnstile model and supports the following procedures using space $O(\frac{1}{\varepsilon} \log N)$.*

- *An update, which is done in time $O(\log N)$.*
- *An ε-HH query, which is done in time $O\left(\frac{1}{\varepsilon} \log^{1+\gamma} N\right)$ with failure probability δ, with a one-sided error guarantee as in Theorems 2.1 and 2.2.*

Finally, note that whenever we state a space requirement for an algorithm, we account for the space needed to initialize and maintain the workspace, as well as the auxiliary space needed to perform update and query procedures.

2.3 Permutations and Pseudorandom Permutations

For a positive integer n, we denote by S_n the set of all permutations over $[n]$.

Definition 2.4 (pseudorandom permutation). *We say that a random variable $\boldsymbol{\pi} \sim S_n$ is a k-wise permutation if for any distinct $i_1, \ldots, i_k \in [n]$ it holds that $(\boldsymbol{\pi}(i_1), \ldots, \boldsymbol{\pi}(i_k))$ is the uniform distribution over k-tuples of distinct elements from $[n]$.*

For $k = 2$, a simple affine transformations work.

Theorem 2.5. *For every prime number n there exists a strongly explicit pairwise permutation $\boldsymbol{\pi}$ with support size $(n-1)n$. Specifically, given a prime field $\mathbb{F}$, the set of all permutations $\pi(x) = ax + b$ for $a \in \mathbb{F}^\star$ and $b \in \mathbb{F}$ yields a pairwise permutation.*

2.4 Expander Graphs

Given an undirected graph G with a diagonal degree matrix D and an adjacency matrix A_G, its transition matrix is given by $A = D^{-1}A_G$, and we denote its second largest eigenvalue by $\lambda(G)$. In Section 3.1 we generalize this notion to weighted bipartite graphs. We will make use of Tanner's inequality, relating vertex expansion and spectral expansion.

Theorem 2.6 ([Tan84]). *Let $G = (V, E)$ be an undirected regular graph with $\lambda(G) \le \lambda$. Then, for every $A \subseteq V$ we have*

$$|\Gamma_G(A)| \ge \frac{1}{\rho(A) + (1-\rho(A))\lambda^2} \cdot |A|.$$

We will need the following bipartite expanders.

Lemma 2.7. *For any positive integers N and $M \le N$ there exists a biregular bipartite graph $G = ([N], [M], E)$ with left-degree $D = \Theta(\log \frac{N}{M})$ such that the following holds. There exists a universal constant $c > 0$ such that for every set $A \subseteq [N]$ satisfying $K \le c \cdot \frac{M}{\log(N/M)}$, it holds that $|\Gamma_G(A)| \ge \frac{D}{4}|A|$.*

Moreover, a uniformly random biregular bipartite graph with the parameters N, M and D as above satisfy that property with probability at least $1 - 2^{-\Omega(M)}$. Such a random graph can be sampled in time $O(ND)$.

Proof (sketch): A vertex expansion of $\frac{D}{4}$ readily follows from good spectral expansion (e.g., by Tanner's inequality for bipartite graphs we will soon state), and a random bipartite expander is a good spectral expander with overwhelming probability (see, e.g., [BDH18]). A direct, vertex expansion result, can be found in [SS96]. We note that an even better vertex expansion is possible using lossless expanders, but it will not be crucial for our application. Lastly, sampling a uniformly random biregular bipartite graph can be done via the configuration model in linear time. ■

2.5 Auxiliary Claims

A series of random variables $X_1, \ldots, X_n \sim \{0,1\}$ are *k-wise independent* if for any distinct $i_1, \ldots, i_k \in [n]$ it holds that $\mathbb{E}[X_{i_1} \cdot \ldots \cdot X_{i_k}] = \mathbb{E}[X_{i_1}] \cdot \ldots \cdot \mathbb{E}[X_{i_k}]$. Good tail bounds for bounded-independence are known.

Theorem 2.8 ([SSS95]). *Let $X_1, \dots, X_n \sim \{0,1\}$ with $\mu = \frac{1}{n}\sum_{i\in[n]} \mathbb{E}[X_i]$. Let $\delta \geq 1$, and let k be a positive integer satisfying $k \leq \delta\mu e^{-1/3}n$. Then, if the random variables are k-wise independent, it holds that*

$$\Pr\left[\left|\frac{1}{n}\sum_{i\in[n]} X_i - \mu\right| \geq \delta\mu\right] \leq e^{-k/2}.$$

The above theorem can be extended to the case in which the random variables are negatively correlated. We will need the following extension of negative dependence (studied, e.g., in [PS97, IK10] for the completely independent case).[10]

Theorem 2.9. *Let $X_1, \dots, X_n \sim \{0,1\}$ be such that for every $i \in [n]$, $\mathbb{E}[X_i] \leq \mu$, and moreover, k is an even integer such that for any distinct $i_1, \dots, i_{k'} \in [n]$ where $k' \leq k$ it holds that*

$$\mathbb{E}\left[\prod_{j\in[k']} X_{i_j}\right] \leq \mu^{k'}.$$

Then, for any $\delta \geq 1$,

$$\Pr\left[\left|\frac{1}{n}\sum_{i\in[n]} X_i - \mu\right| \geq \delta\mu\right] \leq e^{-k/2},$$

assuming $k \leq \delta\mu e^{-1/3}n$.

Claim 2.10. *Let A and E be any two events such that $\Pr[E] \geq 1 - \varepsilon$ for some $0 < \varepsilon < 1$. Then, $|\Pr[A|E] - \Pr[A]| \leq \varepsilon$.*

We give the easy proof in Appendix A.1.

3 The Randomized Encoding Procedure

The construction of $\mathcal{C}$, outlined in Section 1, will require quite a few primitives. The next subsection discusses bipartite graphs coming from high-dimensional expanders, culminating in Theorem 3.6 that described the graphs G and G_{mid} we will use. For the reader's convenience, we include in Table 2 a table of parameters that we will set throughout the construction.

3.1 HDXs-Based Bipartite Expanders

Before giving Theorem 3.6, we need additional preliminaries. A *d-dimensional simplicial complex* X is a family of sets of some ground set $[n]$ that is downwards closed to containment, each of cardinality at most $d+1$. For each integer $i \geq 0$ we denote by $X(i)$ the set of i-dimensional faces, which are the sets of cardinality $i+1$ in X. A complex is *pure* if every i-dimensional face is a subset of some d-dimensional face.

[10] To see that one can extend Theorem 2.8 to our setting, we first note that we can assume without loss of generality that $\mathbb{E}[X_i] = \mu$ for every $i \in [n]$. In such a case, denoting $Z = \frac{1}{n}\sum_{i\in[n]} X_i - \mu$, one can show that for an even k, $\mathbb{E}[Z^k] \leq \mathbb{E}[\widetilde{Z}^k]$, where $\widetilde{Z}$ being the same as Z, but we replace $\{X_1, \dots, X_n\}$ with $\left\{\widetilde{X}_1, \dots, \widetilde{X}_n\right\}$ – independent random variables with $\mathbb{E}[\widetilde{X}_i] = \mathbb{E}[X_i]$. From here we can use standard bounds on $\mathbb{E}[\widetilde{Z}^k]$ to get the desired result.

Parameter	Role	Setting/notes
ε	The parameter for ε-heavy hitters	
δ	The failure probability for list-recovery/heavy-hitters	
N	The size of the universe $\mathcal{U}$ in heavy hitters	
n	The length of the code $\mathcal{C}_0$	$n = \Theta(\log_{1/\varepsilon} N)$
k	The length of the message encoded by $\mathcal{C}_0$	$k = \Theta(n)$
q	The size of the field we work over	$q^k = N$, and we will set $q = \mathrm{poly}(1/\varepsilon)$
V_1	The left-hand side of the bipartite graph G_{mid}, used for the initial propagation step.	$\lvert V_1 \rvert = \Theta(n)$
V_2	The right-hand side of the bipartite graph used to define $\mathcal{C}_0'$, corresponding to sets $T \subseteq [n]$.	$\lvert V_2 \rvert = \Theta(n)$
m_2	The size of the sets $T \in V_2$	$m_2 = 8$
Σ	The alphabet for the code $\mathcal{C}$	$\Sigma = \mathbb{F}_q^{m_2}$
α, α', α''	The rate of the codes $\mathcal{C}_0, \mathcal{C}_0'$, and $\mathcal{C}_0''$ are at least $1-\alpha, 1-\alpha'$, and $1-\alpha''$, respectively.	$\alpha = \alpha' + \alpha''$, and all three are $\Theta\left(\frac{\log q}{q}\right)$
τ	The distance of $\mathcal{C}_0''$ (and hence a bound on the distance of $\mathcal{C}_0$)	$\tau = \Theta(1/q)$
β	We have $\lvert V_2 \rvert \beta = \lvert \mathcal{T} \rvert$.	$\beta = \Theta(\alpha) = \Theta\left(\frac{\log q}{q}\right)$

Table 2: A summary of the notation used in the construction

Let X be a pure d-dimensional complex. Given a probability distribution $\mathcal{D}_d$ on $X(d)$, we extend it to a distribution $\mathcal{D}$ over sequences $s_0 \subset s_1 \subset \ldots \subset s_d$ for $s_i \in X(i)$ in the natural way: Choose $s_d \sim \mathcal{D}_d$, and repeatedly choose $s_{i-1} \subset s_i$ by removing a uniformly random element from s_i. We let $\mathcal{D}_i$ be the distribution induced this way on $X(i)$. For each i we can consider the space of functions $f\colon X(i) \to \mathbb{R}$, which together with the inner product $\langle f, g \rangle_i = \mathbb{E}_{s \sim \mathcal{D}_i}[f(s)g(s)]$ form the inner product space $L^2(X(i))$.

Fix $b < a \leq d$. Let $G_{a,b} = (R, L, E)$ be the weighted bipartite graph with $R = X(a)$, $L = X(b)$,

$$E = \{(s,t) \in R \times L : t \subset s\},$$

and the weight on an edge (s,t) is $\Pr[\mathcal{D}_a = s \wedge \mathcal{D}_b = t]$. Let $M_{a,b}\colon L^2(X(a)) \to L^2(X(b))$ be the *bipartite adjacency operator* defined by

$$(M_{a,b} f)(t) = \sum_{s \in X(a)} \Pr[\mathcal{D}_a = s \mid \mathcal{D}_b = t] \cdot f(s)$$

for every $t \in X(b)$. We denote by $\lambda(G_{a,b})$ the spectral norm of $M_{a,b}$ when restricted to $\{\mathbf{1}\}^{\perp}$, the orthogonal complement of the constant functions. Namely,

$$\lambda(G_{a,b}) = \sup_{f,g \perp \mathbf{1}} \frac{\langle M_{a,b} f, g \rangle_b}{\lVert f \rVert \cdot \lVert g \rVert}.$$

We can further define the *two-step lower walk* as follows. Given $s \in X(a)$, we choose $s' \in X(a)$ by first choosing $t \in X(b)$ given that $t \subset s$ and then choose s' given that $t \subset s'$. We denote the corresponding operator by $D_{a,b}$, and one can see that $D_{a,b} = M^{\dagger}_{a,b} M_{a,b}$, where $M^{\dagger}_{a,b}\colon L^2(X(b)) \to L^2(X(a))$ is the disjoint operator with respect to the inner products defined by $\mathcal{D}_a$ and $\mathcal{D}_b$ on $X(a)$ and $X(b)$, i.e., the unique operator for which

$$\langle f, M_{a,b} g \rangle_a = \langle M^{\dagger}_{a,b} f, g \rangle_b$$

for all $f\colon X(a) \to \mathbb{R}$ and $g\colon X(b) \to \mathbb{R}$. Denoting $\lambda(D_{a,b})$ as the second largest eigenvalue of the self-adjoint operator $D_{a,b}$, this implies that $\lambda(D_{a,b}) = \lambda(G_{a,b})^2$.

If X is a good enough *high-dimensional expander* then $G_{a,b}$ has favorable spectral properties. This connection has been studied, e.g., in [KM17, KO20, DK17, DDFH18, DD19] and we now make it formal.

Definition 3.1 (link). *Given a d-dimensional simplicial complex X equipped with a distribution $\mathcal{D}$, the* link *of $s \in X(i)$ is a $d-(i+1)$-dimensional simplicial complex defined by $X_s = \{t \setminus s : s \subseteq t \in X\}$. The associated probability measure for the link of s is defined by $\Pr_{\mathcal{D}_s}[t] = \Pr_{\mathcal{D}}[t \cup s \mid s]$.*

Definition 3.2 (underlying graph). *Given a d-dimensional simplicial complex X equipped with a distribution $\mathcal{D}$, the* underlying graph *of X is the graph whose vertices are $X(0)$ and edges are $X(1)$, with the restriction of $\mathcal{D}$ to the vertices and edges.*

Definition 3.3 (spectral expander). *A d-dimensional simplicial complex is a λ-HDX if for every $i < d-1$ and every $s \in X(i)$, the underlying graph of the link X_s is a λ-spectral expander, namely its second normalized eigenvalue in magnitude is bounded by λ.*[11]

Theorem 3.4 ([DK17]). *Let X be a d-dimensional λ-HDX. Then, for any integers $b < a \le d$ it holds that*

$$\lambda(D_{a,b}) \le \frac{b+1}{a+1} + O(b(a-b)\lambda).$$

We will use a family of λ-HDXs due to Lubotzky, Samuels and Vishne.

Theorem 3.5 ([LSV05a, LSV05b]). *For infinitely many values of n, for every $\lambda > 0$ and every positive integer d there exists an explicit infinite family of d-dimensional λ-HDXs. Furthermore:*

1. *The simplicial complex can be computed in time $\mathrm{poly}(n)$.*
2. *The distributions $\mathcal{D}_d$ and $\mathcal{D}_0$ are uniform.*
3. *For every $i < d$, each $s \in X(i)$ is contained in at most $D = D(d, \lambda)$ d-dimensional faces. Note that for constants λ and d, D is constant as well.*

More specifically, given λ, letting q be the smallest prime larger than $\frac{4}{\lambda^2}$, there exists such a λ-HDX for each n satisfying $n = q^{cm}$ for some $c = c(d)$ and any large enough $m \in \mathbb{N}$. Also, $D = q^{d^2}$.

[11]Here we give the definition of a *two-sided* HDX, also known as a two-sided link expander or a two-sided local spectral expander. If instead we only require the second eigenvalue to be bounded, this gives rise to the weaker, one-sided, notion. Our results also hold using the weaker variant, due to the result in [KO20].

Hence, there exists a universal constant $c > 0$ such that for infinitely many values of n and any two integer constants m_2 and $m_1 = m_2 - w$ for some constant $w > 0$ we have an $(m_2 - 1)$-dimensional λ-HDX X with $X(0) = [n]$ for $\lambda = \lambda(m_1 - 1, m_2 - 1) \in (0, 1)$ being the largest constant for which both $\lambda(D_{m_2-1,m_1-1}) \leq 1 - \frac{w}{2m_2}$ and $\lambda(D_{m_2-1,0}) \leq \frac{2}{m_2}$. Equipped with X, we are now ready to give the bipartite graphs we work with.

Theorem 3.6. *For any positive integers m_2 and $m_1 = m_2 - w$ (for some positive integer w), and for infinitely many values of n,*[12] *there exist sets $V_1 \subseteq \binom{[n]}{m_1}$ and $V_2 \subseteq \binom{[n]}{m_2}$, a bipartite biregular graph $G = (V_2, [n], E)$ and a weighted bipartite graph $G_{\mathsf{mid}} = (V_2, V_1, E_{\mathsf{mid}}, W)$ such that the following holds.*

1. *The graphs G and G_{mid} are inclusion graphs. Namely, in G, each $s \in V_2$ is connected to all its m_2 elements, and in G_{mid}, each $s \in V_2$ is connected to all its subsets of cardinality m_1. Thus, G has right-degree m_2 and G_{mid} has right-degree $\binom{m_2}{m_1}$.*
2. *There exists a constant $C = C(m_2) > 1$ such that $|V_2| = C \cdot n$ and $|V_1| \leq C \cdot n$.*
3. *It hold that $\lambda(G)^2 \leq \frac{2}{m_2}$.*
4. *Let $G_2 = (V_2, E_2)$ be the two-step random walk graph of G. Then, $\lambda(G_2)$, the second largest eigenvalue, in magnitude, of G_2, is bounded by $\frac{2}{m_2}$.*
5. *It holds that $\lambda(G_{\mathsf{mid}})^2 \leq 1 - \frac{w}{2m_2}$. In particular, G_{mid} is connected.*
6. *Both G and G_{mid} can be computed in time $\mathrm{poly}(n)$.*

Proof: We set $V_2 = X(m_2 - 1)$, $V_1 = X(m_1 - 1)$, $G = G_{m_2-1,0}$ and $G_{\mathsf{mid}} = G_{m_2-1,m_1-1}$. The fact that G_{mid} is unweighted (or, all its edge weights are the same), and is biregular, follows from the uniformity of $\mathcal{D}_0$ and $\mathcal{D}_{m_2-1}$. Also, note that the random walk operator of G_2 is self-adjoint, so all its eigenvalues are nonnegative. All the items then follow from the above discussion. ■

Note that we do not use the full power of HDXs-based containment graphs (say, as in [DHK+19]). In particular, we will not use the spectral properties of G_{mid}, but only that it is connected.

We conclude this section by giving two lemmas regarding the expansion properties of bipartite graphs. The first is a straightforward extension of the expander mixing lemma for bipartite graphs.

Lemma 3.7. *Let $G = (R, L, E)$ be a (possible weighted) bipartite graph with $\lambda(G) \leq \lambda$. Then, for any $A \subseteq L$ and $B \subseteq R$ it holds that*

$$|\Pr[E(A, B)] - \alpha \cdot \beta| \leq \lambda\sqrt{\alpha\beta(1-\alpha)(1-\beta)},$$

where we denote $\alpha = \Pr_{v\sim L}[v \in A]$ and $\beta = \Pr_{v\sim R}[v \in B]$.

The second is an extension of Tanner's inequality, given in Theorem 2.6.

Lemma 3.8. *Let $G = (R = [N], L = [n])$ be a (possible weighted) bipartite graph with $\lambda(G) \leq \lambda$, and denote $C = \frac{N}{n} \geq 1$. Assume that the induced probability distributions on L and R are uniform. Then, for every $S \subseteq R$ it holds that*

$$|\Gamma_G(S)| \geq \frac{1}{\rho(S) + \lambda(1 - \rho(S))} \cdot \frac{|S|}{C}.$$

We defer the proof of Lemma 3.8 to Appendix A.2.

[12]More specifically, for any given n_0 there is such an n satisfying $n_0 \leq n \leq \overline{C} \cdot n_0$ where $\overline{C} = C(m_2)$.

3.2 The Base Code $\mathcal{C}_0$

We set $m_2 = 8$ and $m_1 = w = 4$. From here onward, set n to be some integer for which the graphs G, G_{mid} and G_2 from Theorem 3.6 exist. Our code $\mathcal{C}_0$ will be

$$\mathcal{C}_0 = \mathcal{C}_0' \cap \mathcal{C}_0'',$$

where:

- $\mathcal{C}_0'$ is a linear high rate Tanner code constructed from a subgraph of G.
- $\mathcal{C}_0''$ is a linear high rate Tanner code with noticeable distance.

In what follows, we give the construction of both codes. To this end, we set some parameters. Fix some $\alpha', \alpha'' > 0$ (to be chosen later), and define

$$\beta = \frac{\alpha'}{(m_2 - m_1)C}, \tag{1}$$

where C is the constant, depending on m_2, that is guaranteed to us by Theorem 3.6. Also, set $\alpha = \alpha' + \alpha''$.

3.2.1 The code $\mathcal{C}_0'$

For an inner code for $\mathcal{C}_0'$, we use the following code.

Claim 3.9. *There exists a constant $q_{00} \in \mathbb{N}$ such that q_{00} is the smallest prime power larger than m_2 and so that for every $q \geq q_{00}$ there exists a linear code $\mathcal{C}_{00} \subseteq \mathbb{F}_q^{m_2}$ with the following properties.*

1. *$\mathcal{C}_{00}$ has dimension m_1.*
2. *$\mathcal{C}_{00}$ has distance $\delta_{00} m_2 \geq m_2 - m_1 + 1$. Thus, it is uniquely decodable from $m_2 - m_1$ erasures.*

Proof: For a prime power $q_{00} \geq m_2 + 1$, the standard Reed-Solomon code satisfies these requirements. ■

Fix a prime power $q \geq m_2 + 1$ to be determined later. We define the code $\mathcal{C}_0' \subseteq \mathbb{F}_q^n$ as follows. Let $\mathcal{T} \subseteq V_2$ be some set of $\beta|V_2|$ vertices of V_2. Then,

$$\mathcal{C}_0'(\mathcal{T}) = \left\{ c \in \mathbb{F}_q^n : \forall T \in \mathcal{T}, c|_T \in \mathcal{C}_{00} \right\}.$$

The rate of $\mathcal{C}_0'(\mathcal{T})$ can be lower bounded in a standard way.

Claim 3.10. *$\mathcal{C}_0'(\mathcal{T})$ has rate at least $1 - \alpha'$.*

Proof: The code $\mathcal{C}_0$ is defined by

$$|\mathcal{T}| \left(1 - \frac{m_1}{m_2}\right) m_2 \leq (m_2 - m_1)\beta |V_2|$$

linear equations, so its dimension is at least $n - (m_2 - m_1)\beta C n \geq (1 - \alpha')n$. ■

We will not establish distance for $\mathcal{C}_0'$, but we will require a certain connectivity property from our set $\mathcal{T}$.

Claim 3.11. *There exists a subset $\mathcal{T} \subseteq V_2$ of size $\beta|V_2|$ and an ordering $\mathcal{T} = \{T_1, \ldots, T_{\beta|V_2|}\}$ such that for every integer $1 < i \le \beta|V_2|$, there is a path of length two from T_i to T_j in G_{mid} for some $j < i$. The subset and its ordering can be found deterministically in time $O(n)$.*

Proof: Consider the two-step walk graph G_2. The graph G_2 is connected, and in particular contains an induced tree $\mathcal{T} \subseteq V_2$ of size $\beta|V_2|$. Such a tree can be found deterministically in time $O(|V_2|) = O(n)$, say by performing depth-first search from an arbitrary starting vertex. The ordering can be determined according to the inorder traversal on the (partial) DFS tree. ■

Fixing $\mathcal{T}$ to be the set guaranteed by the above claim, from here onward we denote $\mathcal{C}_0' = \mathcal{C}_0'(\mathcal{T})$. The following easy claim will clarify our choice of $\mathcal{T}$.

Claim 3.12. *For $i > j$, let $T_i, T_j \in \mathcal{T}$ such that there exists a path of length two from T_i to T_j in G_{mid}. Let $c_j \colon T_j \to \mathbb{F}_q$ and $c_i \colon T_i \to \mathbb{F}_q \cup \{\perp\}$ be such that $c_i|_{T_i \cap T_j} = c_j|_{T_i \cap T_j}$. Then, there is at most one possibility to complete c_i to a codeword of $\mathcal{C}_{00}$, and such a completion can be done in constant time.*

The claim follows using Claim 3.9 and the fact that $|T_i \cap T_j| \ge m_1$. Applying the above two claims iteratively, we can conclude the following.

Corollary 3.13. *Denote $U = \Gamma_G(\mathcal{T}) \subseteq [n]$. Then, for any $c_1 \colon T_1 \to \mathbb{F}_q \in \mathcal{C}_{00}$ and $v \in \mathbb{F}_q^{[n]\setminus U}$ there exists at most one $c \in \mathbb{F}_q^n$ such that $c|_{T_1} = c_1$, $c|_{[n]\setminus U} = v$ and $c \in \mathcal{C}_0'$.*

Moreover, given any $c_1 \colon T_1 \to \mathbb{F}_q \in \mathcal{C}_{00}$, the unique $c|_U$ can be found, if exists, deterministically, in time $O(n)$.

3.2.2 The Code $\mathcal{C}_0''$

For $\mathcal{C}_0''$, we will use the following code.

Theorem 3.14. *There exists a prime power q_{00}' such that the following holds for any integer $q \ge q_{00}'$, a positive integer n and $\alpha'' > 0$. There exists an explicit linear code $\mathcal{C}_0'' \subseteq \mathbb{F}_q^n$ of rate $1 - \alpha''$ and relative distance $\delta'' = \frac{c}{\log(1/\alpha'')}\alpha''$ for some universal constant $c > 0$ that is uniquely decodable from up to $\tau \triangleq \delta''$ fraction of erasures in time $\mathrm{poly}(\log(1/\alpha'')) \cdot n$.*

Moreover, $\mathcal{C}_0''$ is a Tanner code with a Reed-Solomon code as as inner code. Namely, there exists a bipartite graph $G'' = (L = [n], R, E'')$ with right-degree D' such that

$$\mathcal{C}_0'' = \left\{ c \in \mathbb{F}_q^n : \forall v \in R, c|_{\Gamma_{G''}(v)} \in \mathcal{C}_{\mathsf{RS}} \right\}$$

for some suitable Reed-Solomon code $\mathcal{C}_{\mathsf{RS}} \subseteq \mathbb{F}_q^{D'}$. In particular, we may take q_{00}' to be the smallest prime power larger than D'.

Proof: Let $G'' = ([n], R, E'')$ be the regular bipartite graph given in Lemma 2.7, for $|R| = \frac{\alpha''}{4}n$. Thus, G'' has left-degree $D = \Theta(\log(1/\alpha''))$ and right-degree $D' = \frac{4D}{\alpha''}$. Let $\mathcal{C}_{\mathsf{RS}} \subseteq \mathbb{F}_q^{D'}$ be a Reed-Solomon code of distance $d_{\mathsf{RS}} = 5$, and define the Tanner code $\mathcal{C}_0''$ accordingly. In this setting, $\mathcal{C}_0''$ indeed has dimension at least

$$n - |R|\left(D' - \dim\left(\mathcal{C}_{\mathsf{RS}}\right)\right) = n - 4|R| = (1 - \alpha'')n.$$

To bound δ'', assume towards a contradiction that there exists a codeword $z \in \mathcal{C}_0''$ of Hamming weight less than $\delta'' n \triangleq \frac{c''}{4} \frac{\alpha'' n}{\log(1/\alpha'')}$, and let $I \subseteq [n]$ be the set of its nonzero coordinates. Let $U \subseteq R$ be the set of constraints that see at least 1 and less than d_{RS} neighbors in I, and let c'' be the constant guaranteed by Lemma 2.7. Thus,

$$D \cdot |I| \geq |U| + d_{\mathsf{RS}} \left|\Gamma_G(I) \setminus U\right| \geq d_{\mathsf{RS}} \cdot \frac{D}{4}|I| - (d_{\mathsf{RS}} - 1)\,|U|,$$

where we have used the expansion property of G''. As

$$(d_{\mathsf{RS}} - 1)\,|U| \geq (d_{\mathsf{RS}} - 1)\, D \cdot |I| - d_{\mathsf{RS}} \cdot \frac{3D}{4}|I|,$$

we get

$$|U| \geq D\left(1 - \frac{3}{4} \cdot \frac{d_{\mathsf{RS}}}{d_{\mathsf{RS}} - 1}\right) \cdot |I| \geq 1,$$

in contradiction of the fact that z was a codeword, and thus should have no violated constraints. We can thus set $c = \frac{c''}{4}$ and q_{00}' to be the smallest prime power which greater than D'.

To establish efficient decoding, we follow a very similar reasoning. Given $w \in (\mathbb{F}_q \cup \{\perp\})^n$, let $E_w \subseteq [n]$ be the set of erased coordinates in w. We perform the following.

1. Initialize $w_0 \leftarrow w$ and $E_0 \leftarrow E_w$.
2. For $i = 0, 1, 2, \ldots,$,
   (a) If $E_i = \emptyset$, break and return $\hat{w} = w_i$.
   (b) Let U_i be the set of constraints in R that see at least 1 and less than d_{RS} neighbors in E_i. As $|E_i| \leq \delta'' n$, it holds that $|U_i| \geq \frac{D}{16}|E_i|$, following our calculations above.
   (c) As the Reed-Solomon code is uniquely decodable from $d_{\mathsf{RS}} - 1$ erasures, we can decode each constraint in U, and therefore can replace each $w_i[j] = \perp$ with the correct symbol in $\mathbb{F}_q$ whenever $j \in \Gamma_G(U_i) \cap E_i$. Update w_i to w_{i+1} accordingly.
   (d) Set $E_{i+1} = E_i \setminus \Gamma_G(U_i)$.

To complete the correctness of the decoding procedure, we need to argue that $\Gamma_G(U_i) \cap E_i$ is never empty. This is clear, as $\frac{D}{16}|E_i| \geq 1$ whenever E_i itself is nonempty.

Finally, we want to argue that the above procedure can be performed efficiently. Toward this end, first let us bound the number of times Item 2 above is performed. Each constraint in U_i is adjacent to at least one vertex of E_i. It holds that

$$|\Gamma_G(U_i) \cap E_i| \geq \frac{1}{16}|E_i|,$$

as otherwise we would have a vertex in E_i with more than D adjacent vertices. Thus, $|E_{i+1}| \leq \frac{15}{16}|E_i|$, and so $O(\log n)$ iterations suffice. At each iteration, we need to compute U_i and $\Gamma_G(U_i) \cap E_i$, which takes $O(D|E_i|)$ time, and perform unique decoding of Reed-Solomon from erasures for $|\Gamma_G(U_i) \cap E_i|$ times. Unique decoding of Reed-Solomon from erasures amounts to performing a univariate polynomial interpolation over $\mathbb{F}_q$. Fast, FFT-based, interpolation can be done in

$O(D' \log^2 D' \log\log D')$ field operations (see, e.g., [VZGG13, Section 10.2]). Overall, the decoding running time is bounded by

$$\sum_i \left(O(D|E_i|) + |\Gamma_G(U_i) \cap E_i| \cdot O(D' \log^2 D' \log\log D') \right) = O(D) \cdot \sum_i |E_i| + \\ O(D' \log^2 D' \log\log D') \cdot |E_0| = O(D) \cdot |E_0| + O(D' \log^2 D' \log\log D') \cdot |E_0|,$$

and since $D'|E_0|$ is bounded from above by a constant, the above is at most

$$O(\log^2 D' \log\log D') \cdot n = \mathrm{poly}(\log(1/\alpha'')) \cdot n.$$

■

3.2.3 Putting $\mathcal{C}_0'$ and $\mathcal{C}_0''$ Together to Obtain $\mathcal{C}_0$

Fix q to be the a prime power larger than q_{00}, q_{00}', i.e., larger than

$$\max\left\{m_2 + 1, D' + 1\right\} = \Theta\left(\frac{\log(1/\alpha'')}{\alpha''}\right),$$

and instantiate $\mathcal{C}_0'$ and $\mathcal{C}_0''$ accordingly (we will determine the precise value of q later on). Thus,

$$\mathcal{C}_0 = \mathcal{C}_0' \cap \mathcal{C}_0''$$

is a linear code over $\mathbb{F}_q^n$ with rate at least

$$r_0 \geq 1 - \alpha' - \alpha'' = 1 - \alpha.$$

We can lower-bound its distance by the distance of $\mathcal{C}_0''$, but we will not use this property directly.

For the choice of parameters, we set

$$\alpha' = \alpha'' = \frac{c_\alpha \log q}{q},$$

where $c_\alpha > 1$ is a universal constant chosen to satisfy $q \geq D' + 1$. Thus,

$$\alpha = \frac{2c_\alpha \log q}{q} \qquad \text{and} \qquad \tau \geq \frac{c_\tau}{q} \tag{2}$$

for some universal constant c_τ, where we recall that τ is the fraction of erasures that $\mathcal{C}_0''$ can handle in Theorem 3.14.

To conclude this section, we argue that $\mathcal{C}_0$ admits nearly-linear time encoding procedure.

Lemma 3.15. *There exists a deterministic algorithm that on input $x \in \mathbb{F}_q^{k=(1-r_0)n}$, runs in time $O(\log q \cdot n)$ and outputs $\mathcal{C}_0(x) \in \mathbb{F}_q^n$, given access to $\mathcal{T}$, G and G''. The algorithm uses $\mathrm{poly}(n)$ preprocessing time and $O(\log q \cdot n)$ auxiliary space.*

Proof: The encoding amounts to encoding a Tanner code, imposing the constraints of both $\mathcal{C}_0'$ and $\mathcal{C}_0''$. We can think of $\mathcal{C}_0$ as a low density LDPC code with coefficients from $\mathbb{F}_q$ over its edges, by identifying each parity check of Reed-Solomon with a separate vertex. Namely, there exists $G_{\mathsf{LDPC}} = (L = [n], R_{\mathsf{LDPC}}, E_{\mathsf{LDPC}})$ such that G_{LDPC} is right-regular with degree

$$d_{\mathsf{LDPC}} = \max\left\{D', m_2\right\} = D',$$

$|R| = \alpha \cdot n$, and each $e \in E_{\mathsf{LDPC}}$ is assigned with $c(e) \in \mathbb{F}_q$, such that

$$\mathcal{C}_0 = \left\{ x \in \mathbb{F}_q^n : \forall u \in R, \sum_{e=(u,v)} c(e) = 0 \right\}.$$

In our case, computing G_{LDPC} given access to the graphs used in the Tanner codes can be done in time $O(D' \cdot n)$, as the parity check matrix of the Reed-Solomon code can be easily computed.

Fortunately, Kobayashi and Shibuya gave a linear time encoding algorithm for q-ary LDPC codes. Formally, their result goes as follows.

Theorem 3.16 ([KS12])**.** *Let H be an $m \times n$ parity check matrix of some linear code $\mathcal{C} \colon \mathbb{F}_q^k \to \mathbb{F}_q^n$, let $\mathrm{nz}(H)$ be the number of its nonzero entries and let $\mathrm{rw}(H)$ be the maximal number of nonzero entries in each row. Then, there exists an algorithm that runs in time $O(\mathrm{nz}(H) + m \cdot \mathrm{rw}(H))$ and computes $\mathcal{C}(x)$ given $x \in \mathbb{F}_q^k$. The algorithm uses $\mathrm{poly}(n)$ preprocessing time and $O(\mathrm{nz}(H) + m \cdot \mathrm{rw}(H))$ auxiliary space.*

We note that the preprocessing step in [KS12] involves inversion of square matrices of dimension roughly αn, however it suffices to *store* matrices of dimension roughly $D' \times \alpha n$ in the preprocessing step.

In our case, both $\mathrm{nz}(H)$ and $m \cdot \mathrm{rw}(H)$ amounts to

$$|E_{\mathsf{LDPC}}| = D' \cdot |R| = \alpha D' \cdot n = O\left(\log q \cdot n\right),$$

and so the lemma follows from Theorem 3.16. ■

3.3 High-Rate Codes Have Many Balanced Codewords

We argue that $\mathcal{C}_0$, and in fact any high rate code, has many codewords whose empirical distribution is close to uniform (we will prove a slightly stronger claim). Towards this end, we first extend the standard Shannon entropy, as well as the Kullback–Leibler divergence, to arbitrary logarithm bases.

Definition 3.17 (Shannon entropy, KL divergence)**.** *Let X be a random variable distributed over some domain Ω, and let $q \geq 2$ be any integer. The q-ary Shannon entropy of X is given by*

$$H_q(X) = \sum_{z \in \Omega} \Pr[X = z] \log_q \frac{1}{\Pr[X = z]}.$$

The q-ary KL divergence between two random variables $X, Y \sim \Omega$ is given by

$$\mathcal{D}_q(X \| Y) = \sum_{z \in \Omega} \Pr[X = z] \log_q \frac{\Pr[X = z]}{\Pr[Y = z]}.$$

In particular, if $\Omega = \mathbb{F}_q^\ell$ then $\mathcal{D}_q(X \| U_\Omega) = \ell - H_q(X)$.

Note that $H_q(X) \in [0, \log_q |\Omega|]$, and $H_q(X) = \log_q |\Omega|$ if and only if X is the uniform distribution over Ω.

Given a codeword $c \in \mathbb{F}_q^n$, we write $H_q(c)$ to denote the q-ary entropy of the corresponding *empirical distribution* $\boldsymbol{c}$ over $\mathbb{F}_q$ for which $\Pr[\boldsymbol{c} = \sigma] = \Pr_{i\in[n]}[c_i = \sigma]$. Given two codewords $c, c' \in \mathbb{F}_q^n$, we write $H_q(c, c')$ to denote the q-ary entropy of the empirical distribution $(\boldsymbol{c}, \boldsymbol{c'})$ over $\mathbb{F}_q^2$ for which $\Pr[(\boldsymbol{c}, \boldsymbol{c'}) = (\sigma, \sigma')] = \Pr_{i\in[n]}[c_i = \sigma \wedge c'_i = \sigma']$.

The following claim is a direct corollary of Pinsker's inequality (see, e.g., [Gra11, Section 6.3]).

Claim 3.18. *Let $c, c' \in \mathbb{F}_q^n$ be such that $H_q(c, c') \geq 2(1-\gamma)$ for some $\gamma \geq 0$. Then,*

$$\left|(\boldsymbol{c}, \boldsymbol{c'}) - U_{\mathbb{F}_q\times\mathbb{F}_q}\right| \leq \sqrt{\ln q \cdot \gamma}.$$

The same holds for the empirical distribution of a single copy. Namely, if $c \in \mathbb{F}_q^n$ is such that $H_q(c) \geq 1-\gamma$ then $\left|\boldsymbol{c} - U_{\mathbb{F}_q}\right| \leq \sqrt{\frac{1}{2}\ln q \cdot \gamma}$.

Next, we show that there are not many (c, c')-s with small $H_q(c, c')$. First, we exhibit a nice property of $H_q(\cdot)$.

Claim 3.19. *Let $X \sim \mathbb{F}_q$ be any empirical distribution of a vector in $\mathbb{F}_q^n$. Then, the number of $c \in \mathbb{F}_q^n$ whose empirical distribution is identical to X is at most $q^{nH_q(X)}$.*

Similarly, if $X \sim \mathbb{F}_q^2$ is any empirical distribution of a pair of vectors, each in $\mathbb{F}_q^n$, then the number of $(c, c') \in \mathbb{F}_q^n \times \mathbb{F}_q^n$ whose empirical distribution is identical to X is at most $q^{nH_q(X)}$.

For the proof, see [CS04, Section 2].

Claim 3.20. *Fix some $0 < \gamma < 1$. The number of vectors $(c, c') \in \mathbb{F}_q^n \times \mathbb{F}_q^n$ satisfying $H_q(c) < 2(1-\gamma)$ is at most $n^{2q} \cdot q^{(1-\gamma)2n}$.*

Proof: Fix any empirical distribution $X \sim \mathbb{F}_q^2$ satisfying $H_q(X) < 2(1-\gamma)$. By Claim 3.19, there are at most $q^{(1-\gamma)2n}$ vectors $c \in \mathbb{F}_q^n$ whose symbols distribute according to X. As there are at most $\binom{n+q-1}{q}^2 \leq n^{2q}$ such distributes overall, we get our desired bound. ■

We are now ready to prove our main lemma for this section.

Lemma 3.21. *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be an error correcting code of rate $1-\alpha$ such that $\frac{n}{\log n} \geq \frac{2q}{\alpha \log q}$. Then,*

$$\Pr_{(c,c')\in\mathcal{C}\times\mathcal{C}}\left[\left|(\boldsymbol{c}, \boldsymbol{c'}) - U_{\mathbb{F}_q\times\mathbb{F}_q}\right| \leq \sqrt{2\ln q \cdot \alpha}\right] \geq 1 - q^{-\alpha n}.$$

Proof: Let $B \subseteq \mathbb{F}_q^n \times \mathbb{F}_q^n$ be the set of codewords (c, c') for which $\left|(\boldsymbol{c}, \boldsymbol{c'}) - U_{\mathbb{F}_q\times\mathbb{F}_q}\right| > \sqrt{2\ln q \cdot \alpha} \triangleq \varepsilon$. By Claim 3.18, each $(c, c') \in B$ satisfies $H_q(c, c') < 2(1-\gamma)$ for $\gamma = \frac{\varepsilon^2}{\ln q}$. Thus, by Claim 3.20, $|B| \leq n^{2q} \cdot q^{(1-\gamma)2n}$. As $|\mathcal{C} \times \mathcal{C}| = q^{(1-\alpha)2n}$, we get that

$$\Pr_{(c,c')\in\mathcal{C}\times\mathcal{C}}\left[\left|(\boldsymbol{c}, \boldsymbol{c'}) - U_{\mathbb{F}_q\times\mathbb{F}_q}\right| > \varepsilon\right] \leq \frac{n^{2q} q^{(1-\gamma)2n}}{q^{(1-\alpha)2n}} = q^{-\left(\gamma-\alpha-\frac{q\log_q n}{n}\right)2n}.$$

As $\gamma - \alpha - \frac{q \log_q n}{n} \geq \frac{\alpha}{2}$, the lemma follows. ■

3.4 The Randomized Encoding $\mathcal{C}$

We fix some universe $\mathcal{U}$ of cardinality N, a heavy hitters threshold $\varepsilon > 0$ and a designated failure probability $\delta > 0$, and write $N = q^k$, where $q = q(\varepsilon)$ is the parameter guaranteed to us from Section 3.2 whose exact value we will soon determine.

We also set some small $\zeta > 0$, and fix an independence parameter

$$t = n^{1-\zeta}.$$

One can think of ζ as an arbitrary small constant, but it will help us setting $\zeta = o(1)$, and will be possible as long as ε is not too small.

Let

$$\mathcal{C}_0 \colon \mathbb{F}_q^k \to \mathbb{F}_q^n$$

be the error correcting code from Section 3.2, so $n = \frac{k}{r_0} = O(\log_q N)$. Let $G = (V_2, [n], E)$ be the biregular bipartite graph guaranteed to us by Theorem 3.6, with right-degree m_2, and $V_2 = C \cdot n$. We use the following families of permutations.

- Let $\boldsymbol{\pi}_1 \sim S_N$ be the pairwise permutation family guaranteed to us by Theorem 2.5. We enumerate the support of $\boldsymbol{\pi}_1$ as $\pi_1^{(1)}, \ldots, \pi_1^{(\ell_1)}$, where $\log \ell_1 = O(\log N)$.
- Let $\boldsymbol{\pi}_2 \sim S_n$ be a truly uniform permutation. Each $\pi_2 \sim \boldsymbol{\pi}_2$ can be explicitly described using n words, and it takes $O(n)$ time to sample from $\boldsymbol{\pi}_2$ (say by an efficient Fisher-Yates shuffle).

Denote $\Sigma = \mathbb{F}_q^{m_2}$. Given $\pi_1 \sim \boldsymbol{\pi}_1$ and $\pi_2 \sim \boldsymbol{\pi}_2$, the encoding

$$\mathcal{C}^{\pi_1,\pi_2} \colon \mathbb{F}_q^k \to \Sigma^{|V_2|}$$

goes as follows. Given $x \in [N]$,

1. Compute $x_0 = \pi_1(x)$.
2. Compute $y_0 = \mathcal{C}_0(x_0) \in \mathbb{F}_q^n$.
3. Permute the coordinates of y_0 according to π_2. Namely, let $y \in \mathbb{F}_q^n$ be such that $y[i] = y_0[\pi_2(i)]$ for every $i \in [n]$.
4. Aggregate the symbols of y according to G. Namely, let $c \in \Sigma^{V_2}$ be the word in which for every $T \in V_2$, $c[T] = y|_{\Gamma_G(T)} \in \mathbb{F}_q^{m_2}$.
5. Output c.

First, note that $\mathcal{C}$ has constant rate: $\frac{r_0}{C \cdot m_2}$. We next argue for the efficiency of the encoding.

Lemma 3.22. *The encoding of $\mathcal{C}$ can be computed in time $O(\log N)$ and space $O(\log N)$ with a preprocessing step which takes $\mathrm{poly}(\log N)$ time. More specifically, given $\mathcal{T}$, G, and G'', $x \in \mathbb{F}_q^k$, $i_1 \in \{0,1\}^{\log \ell_1}$ and $\pi_2 \sim \boldsymbol{\pi}_2$, then*

$$\mathcal{C}^{\pi_1^{(i_1)},\pi_2}(x)$$

can be computed in time $O(\log N)$.

Proof: The preprocessing step includes the following.

- Computing the representation of G as an adjacency list in time $\mathrm{poly}(n)$ (see Theorem 3.6).
- Computing the set $\mathcal{T} \subseteq V_2$ (see Section 3.2). This takes $O(n)$ time.
- Computing the representation of G''. We draw G'' uniformly at random, and aggregate the error in the end. By Lemma 2.7, this takes $O(n)$ time.
- Performing the preprocessing step for the encoding of $\mathcal{C}_0$. By Lemma 3.15 we can do this in time $\mathrm{poly}(n)$ too.
- Drawing $i_1 \in \{0,1\}^{\ell_1}$ uniformly at random, as well as $\pi_2 \sim \boldsymbol{\pi}_2$ uniformly at random. By the discussion above, this takes $O(n)$ time.

Observe, furthermore, that the preprocessing step uses $O(\log q \cdot n)$ space. Once everything is in place,

- Applying π_1 can be done in constant time,
- Computing $\mathcal{C}_0(x)$ can be done in $O(\log q \cdot n)$ time (see Lemma 3.15),
- Permuting according to π_2 can be done in linear time, and,
- Folding according to G ca be done in time $O(|V_2| \cdot m_2) = O(n)$.

The lemma is thus concluded. ■

As stated, our combinatorial objects (graph families, permutation families) exist for infinite value of n. However, with negligible loss in parameters, we can assume from here onward that we can handle any positive integer n by performing standard modifications (and in particular, we will perform field arithmetic even when we do not explicitly say the field's cardinality is a prime power).

4 The Heavy Hitters Algorithm

Recall that we fixed some universe $\mathcal{U}$ of cardinality $N = q^k$, and were given heavy hitters parameters $\varepsilon, \delta > 0$. Also, recall that we set $n = O(\log_q N)$ following the parameters of our code $\mathcal{C}$, and the parameter q is yet to be set, and will be set later on as a function of ε. We will use the primitives defined in Section 3, and the following ingredients from Section 2.2.

- Let CMS be the data structure from Theorem 2.1, instantiated with failure probability $\delta_1 = \frac{\delta}{3|\Sigma|}$, threshold parameter ε, and universe size N. For simplicity, we assume that $\delta \leq |\Sigma|^{-1}$. Thus, by Theorem 2.1, the space requirement is $S_1 = O(\frac{1}{\varepsilon}\log\frac{1}{\delta})$, the update time is $U_1 = O(\log\frac{1}{\delta})$ and the 1-query time is $Q_1 = O(\log\frac{1}{\delta})$. It is instructive to think of $\delta = \frac{1}{\mathrm{poly}(N)}$.
- Let InnerHH be the data structure from Theorem 2.3, instantiated with $\gamma = \frac{1}{4}$, failure probability $\delta_2 = \frac{c_\varepsilon \log q}{2\varepsilon \cdot \sqrt{q}}$ for some constant c_ε later to be determined, threshold parameter ε, and universe size $|\Sigma|$, recalling that $|\Sigma| = q^{m_2}$. Note that $\delta_2 \leq \frac{1}{|\Sigma|}$ for a large enough q. By Theorem 2.2, the space requirement is $S_2 = (\frac{1}{\varepsilon}\log q)$, the update time is $U_2 = O(\log q)$ and the query time is $Q_2 = O(\frac{1}{\varepsilon}\log^2 q)$.

Remark 1 (Choice of InnerHH)**.** *We remark that if we wanted to instantiate* InnerHH *with, say, a Count-Min Sketch (rather than the construction of [LNNT16a] from Theorem 2.3), our construction would still work, with a slightly worse dependence on ε.*

4.1 Setting Up the Workspace

We allocate the space needed for a single instance of CMS, CMS, and n instances of InnerHH, HH_T for every $T \in V_2$. This takes space

$$S_1 + |V_2| \cdot S_2 = O\left(\frac{n}{\varepsilon} \cdot \log q + \frac{\log(1/\delta)}{\varepsilon}\right).$$

Next, draw the randomness $r \sim \boldsymbol{r}$ needed for the auxiliary data structure instances of CMS and InnerHH, where the randomness is independent among the instances. We use $\boldsymbol{r}_T$ when we want to explicitly refer to the randomness used by the individual auxiliary sketch HH_T.

Finally, we allocate the space needed to compute $\mathcal{C}$, including the preprecessing step. This includes computing the required graphs and drawing the randomness for the permutations. By Lemma 3.15, this takes $O(\log N)$ space. We will also need another $O(\log N)$ words for auxiliary computations. We record the overall space requirement in the following lemma.

Lemma 4.1 (space requirement)**.** *The overall space used is $O\left(\frac{\log(N/\delta)}{\varepsilon}\right)$.*

4.2 The Update Procedure

We are given $x \in \mathcal{U}$ and $f(x) = \Delta \in \mathbb{R}$.

1. Perform an update to CMS on the input (x, Δ).
2. For every $T \in V_2$, compute $c_T = \mathcal{C}^{\pi_1,\pi_2}(x)[T] \in \Sigma$.
3. Upon computing each c_T, perform an update to HH_T on the input (c_T, Δ).

The next lemma readily follows from our discussions above.

Lemma 4.2 (update time)**.** *The above update procedure can be preformed in time $O\left(\log \frac{N}{\delta}\right)$ and can be computed within the workspace's auxiliary space.*

Before we continue, we set a new helpful notation.

The function f_T. For $T \in V_2$ and $\sigma \in \Sigma$, let $f_T(\sigma)$ denote the frequency of σ at HH_T, namely

$$f_T(\sigma) = \sum_{x \in \mathcal{U}} f(x) \cdot \mathbf{1}_{\mathcal{C}(x)[T]=\sigma}.$$

Note that f_T is a function of π_1 and π_2. We would often want to specify it explicitly, so we denote it by $f_T^{\pi_1,\pi_2}$.

4.3 The Query Procedure

Our query procedure goes as follows. First, we apply the ε-heavy hitters algorithm on each inner ε-HH data structure. Treating the output of each application as an *input list*, we attempt to list-recover the code $\mathcal{C}$. The list-recovery will start by establishing, for every initial "advice", a large enough fraction of correct symbols induced by our set $\mathcal{T}$ given in Section 3. Then, we will attempt to propagate the information to additional symbols, relying on expansion properties and favorable properties of the input lists induced by the randomness used in the encoding process. Finally, we use the unique decoding algorithm of the code $\mathcal{C}_0$. See Section 1 for a more elaborate high-level discussion of our list-recovery approach to the query procedure.

1. Run the heavy hitters algorithm with threshold parameter ε and error parameter δ_2 on each HH_T^r to obtain a list $\mathcal{L}_T^{\pi_1,\pi_2,r}$ for every $T \in V_2$.
2. Choose an arbitrary $T^\star \in \mathcal{T}$ that satisfy $\left|\mathcal{L}_{T^\star}^{\pi_1,\pi_2,r}\right| \le \frac{4}{\varepsilon}$, where $\mathcal{T}$ is the set given in Section 3. Initialize $\mathcal{L}^{\pi_1,\pi_2,r} \leftarrow \emptyset$.
3. For each $\sigma^\star \in \mathcal{L}_{T^\star}^{\pi_1,\pi_2,r}$,
   (a) Set $\mathcal{T}_0 \leftarrow \mathcal{T}$ and $\hat{y} \in (\mathbb{F}_q \cup \{\perp\})^n$. We start with $\hat{y} = \perp^n$.
   (b) Denoting $U = \Gamma_G(\mathcal{T}) \subseteq [n]$, use Corollary 3.13 and $\sigma^\star \colon T^\star \to \mathbb{F}_q$ to find the unique $c|_U$ that agrees with $\mathcal{C}_0''$. If none exists, move on to the next $\sigma^\star$.
   (c) Set $\hat{y}|_U = c|_U$.
   (d) For $i = 0, 1, 2, \ldots$,
      - Let $\mathcal{S}_i = \Gamma_G(\mathcal{T}_i)$, and note that we already know $\hat{y}[j]$ for every $j \in \mathcal{S}_i$.
      - Let $\mathcal{F}_i = \Gamma_G(\mathcal{S}_i) \setminus \mathcal{T}_i$, and set $\mathcal{T}_{i+1} \leftarrow \mathcal{T}_i$.
      - For every $T \in \mathcal{F}_i$ such that $\left|\mathcal{L}_T^{\pi_1,\pi_2,r}\right| \le \frac{4}{\varepsilon}$,
        - By definition, there is some $j \in \mathcal{S}_i$ such that $j \in \Gamma_G(T)$.
        - If there is a unique $\sigma \in \mathcal{L}_T^{\pi_1,\pi_2,r}$ such that $\sigma_j = \hat{y}[j]$,[13] set $\hat{y}|_{\Gamma_G(T)} \leftarrow \sigma$ and add T to $\mathcal{T}_{i+1}$.
      - If $|\Gamma_G(\mathcal{T}_{i+1})| \ge (1-\tau)n$, for the τ given in Theorem 3.14, break.
   (e) Apply π_2^{-1} to the coordinates of $\hat{y}$ to get $\hat{y}_0$.
   (f) Run the unique decoding algorithm of $\mathcal{C}_0''$ on $\hat{y}_0$ to recover all coordinates of $\hat{y}_0$. If the unique decoding failed, move on to the next $\sigma^\star$.
   (g) Check that $\hat{y}_0 \in \mathcal{C}_0'$. If not, move on to the next $\sigma^\star$.
   (h) We know that $\hat{y}_0 \in \mathcal{C}_0$. Add it to $\mathcal{L}^{\pi_1,\pi_2,r}$.
4. For every $\hat{y}_0 \in \mathcal{L}^{\pi_1,\pi_2,r}$,
   (a) Retrieve the $\hat{x}_0$ satisfying $\mathcal{C}_0(\hat{x}_0) = \hat{y}_0$.
   (b) Compute $\hat{x} = \pi_1^{-1}(\hat{x}_0)$.
   (c) Use CMS to obtain $\hat{f}$, an estimate of $f(\hat{x})$. If $\hat{f} \ge \varepsilon\|f\|_1$, add $\hat{x}$ to the final heavy hitters list.

[13] Abusing notation, by σ_j we actually refer to $\sigma[j']$ for $j' \in [D]$, where j is the j'-th element of $\Gamma_G(T)$.

Lemma 4.3 (query time)**.** *The above query procedure takes* $\frac{\log N \cdot \mathrm{polylog}(q)}{\varepsilon}$ *time and can be computed within the workspace's auxiliary space.*

Proof: In the preprocessing step G and $\mathcal{T}$ were already computed, as well as the graph G'' needed to decode $\mathcal{C}_0''$. Running the heavy hitters algorithm on all HH_T^r-s in Item 1 takes $O\left(\frac{1}{\varepsilon}\log^2 q \cdot n\right)$ time. During Item 1, we will store not only $\mathcal{L}_T$, but also m_2 Red-Black trees, $H_{T,j}$ for $j \in [m_2]$. These data structures will store key-value pairs with keys in $\mathbb{F}_q$ and support `search` and `insert`, each in time $O(\log q)$. These will have the property that for $a \in \mathbb{F}_q$ $H_{T,j}.\texttt{search}(a)$ will return a pointer to $\sigma \in \mathcal{L}_T$ if σ is the unique element of $\mathcal{L}_T$ with $\sigma_j = a$, and otherwise it will return -1. Note that we can initialize such data structures in time $O(m_2|\mathcal{L}_T|\log q)$, so the total amount of time required is $O(nm_2|\mathcal{L}_T|\log q) = O\left(\frac{n\log q}{\varepsilon}\right)$. Moreover, the space required to store both $\mathcal{L}_T$ and the m_2 search trees is $O(q \cdot m_2 + |\mathcal{L}_T|)$ for each T, for a total of $O(n/\varepsilon)$ space.

Moving onto the next step of the algorithm, computing $c|_U$, per $\sigma^\star$, takes $O(n)$ time, for a total of $O(n/\varepsilon)$ time, recalling that there are at most $O(1/\varepsilon)$ values of $\sigma^\star$ to iterate through.

We now consider each iteration of Item 3d. For each $\sigma^\star$, we essentially do a breadth-first search on the graph G, continuing along a path only if we add T to $\mathcal{T}_{i+1}$. At each step in this breadth-first search, we must decide whether or not to add T to $\mathcal{T}_{i+1}$, and whether to update $\hat{y}|_{\Gamma_G(T)}$. We do this by querying the data structures $H_{T,j}$ described above. In time $O(\log q)$, given a value of j and $\hat{y}[j]$, we query $H_{T,j}.\texttt{search}(\hat{y}[j])$. If it returns $\sigma \in \mathcal{L}_T$, we add T to $\mathcal{T}_{i+1}$ and update $\hat{y}$ in time $O(1)$. Otherwise we do nothing. Therefore, the total time at each step of the breadth-first search is $O(\log q)$, and so the total running time, per $\sigma^\star$, is $O(n\log q)$, recalling that there are $O(n)$ vertices and edges in G. Thus, the total running time of Item 3d over all $\sigma^\star$ is $O\left(\frac{n\log q}{\varepsilon}\right)$.

Step Item 3e takes $O(n)$ time, and Item 3f take $\mathrm{poly}(\log q) \cdot n$ time, following Theorem 3.14. To perform Item 3g, we need to check all the constraints of $\mathcal{C}_0'$, which can be done in linear time. Overall, all iterations of Item 3 take $(\mathrm{poly}(\log q) + O(1/\varepsilon)) \cdot n$ time. Retrieving $\hat{x}_0$ for each $\hat{y}_0$ takes $O(n\log q)$ time, given the preprocessing step of Lemma 3.15. Applying π_1^{-1} amounts to basic field arithmetic. As there are at most $\frac{4}{\varepsilon}$ elements in $\mathcal{L}^{\pi_1,\pi_2,r}$ (otherwise we can just abort and declare failure), we can maintain an overall running time of $\frac{n \cdot \mathrm{poly}(\log q)}{\varepsilon}$.

For the auxiliary space needed to perform the query procedure, note that we can only store the $\hat{y}_0$-s that end up being in the final heavy hitters list.[14] Thus, we only need to store $O(n/\varepsilon)$ words, on top of the auxiliary space needed for the unique decoding algorithm and the auxiliary space needed for the HH_T-s and CMS, which is already allocated. ∎

5 Correctness of the Query Procedure

In this section we will establish the correctness of the above query procedure, or, looking at it differently, we will show that our lists are "random enough" to admit very good list recovery parameters via our expander-based algorithm.

Set

$$\eta = \frac{c_\eta \log q}{\varepsilon \cdot \sqrt{q}}$$

for some constant c_η soon to be determined.

[14]As CMS may fail and return $\hat{x}$ which is not a true heavy hitter, the final list might be larger than $\frac{1}{\varepsilon}$. However, such a case is considered a failure, and we will later see that it happens with probability at most δ.

Definition 5.1. *We say $T \in V_2$ is* good *for (π_1, π_2, r) if*

$$\left\{\sigma \in \Sigma : f_T^{\pi_1,\pi_2}(\sigma) \geq \varepsilon\|f\|_1\right\} \subseteq \mathcal{L}_T^{\pi_1,\pi_2,r} \subseteq \left\{\sigma \in \Sigma : f_T^{\pi_1,\pi_2}(\sigma) \geq \frac{\varepsilon}{4}\|f\|_1\right\}.$$

Claim 5.2. *For every $T \in V_2$, any fixing of the randomness $\pi_1 \sim \boldsymbol{\pi}_1$ and $\pi_2 \sim \boldsymbol{\pi}_2$, and with probability at least $1 - \delta_2$ over $r \sim \boldsymbol{r}$, it holds that T is good for (π_1, π_2, r).*

Proof: The fixing of $\pi_1 \sim \boldsymbol{\pi}_1$ and $\pi_2 \sim \boldsymbol{\pi}_1$ makes f_T a deterministic function of its input. For any $T \in V_2$, it holds that

$$\mathcal{L}_T^{\pi_1,\pi_2,r} \subseteq \left\{\sigma \in \Sigma : f_T^{\pi_1,\pi_2}(\sigma) \geq \frac{\varepsilon}{4}\|f\|_1\right\}$$

with probability at least $1 - \delta_2$ over $r \sim \boldsymbol{r}$, and

$$\left\{\sigma \in \Sigma : f_T^{\pi_1,\pi_2}(\sigma) \geq \varepsilon\|f\|_1\right\} \subseteq \mathcal{L}_T^{\pi_1,\pi_2,r}$$

with probability 1. ■

For $\sigma \in \Sigma$, we define

$$S_\sigma = \{\sigma' \in \Sigma \setminus \{\sigma\} \;:\; \exists j \in [m_2], \sigma'_j = \sigma_j\}. \tag{3}$$

Lemma 5.3. *Fix $T \in V_2$, $x \in \mathcal{U}$. Fix $\sigma \in \Sigma$. Then, there exists a constant $c_\varepsilon > 1$ such that*

$$\Pr\left[\exists \sigma' \in S_\sigma, f_T^{\pi_1,\pi_2}(\sigma') \geq \frac{\varepsilon}{4}\|f\|_1 \;\middle|\; \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[T] = \sigma\right] \leq \frac{c_\varepsilon \log q}{\varepsilon \cdot \sqrt{q}}.$$

Moreover, for any pairwise disjoint $T, T_2, \dots, T_t \in V_2$ and an event $F_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,T_2,\dots,T_t}(x)$ whose randomness is over $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2^{-1}(T_2), \dots, \boldsymbol{\pi}_2^{-1}(T_t)$,

$$\Pr\left[\exists \sigma' \in S_\sigma, f_T^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(\sigma') \geq \frac{\varepsilon}{4}\|f\|_1 \;\middle|\; \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[T] = \sigma \wedge F_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,T_2,\dots,T_t}(x)\right] \leq \frac{c_\varepsilon \log q}{\varepsilon \cdot \sqrt{q}}.$$

Proof: For the ease of presentation, we begin with the first statement, although it follows from the second. Then we explain how to prove the second statement.

We first observe that for any $\pi_1 \sim \boldsymbol{\pi}_1$ and $\pi_2 \sim \boldsymbol{\pi}_2$,

$$\max_{\sigma' \in S_\sigma} f_T^{\pi_1,\pi_2}(\sigma') \leq \sum_{z \in \mathcal{U} \setminus \{x\}} f^{\pi_1,\pi_2}(z) \cdot \mathbf{1}\left\{\exists j \in T, \mathcal{C}^{\pi_1,\pi_2}(x)[j] = \mathcal{C}^{\pi_1,\pi_2}(x)[j]\right\} \tag{4}$$

and we will bound the latter using Markov's inequality.

Since $\boldsymbol{\pi}_1$ is a pairwise pseudorandom permutation, $(\boldsymbol{\pi}_1(x), \boldsymbol{\pi}_1(z)) = U_{\mathcal{U}} \times U_{\mathcal{U}}$. By Lemma 3.21, and using the fact that $\mathcal{C}_0$ is surjective, with probability at least $1 - q^{-\alpha n}$ over $\pi_1 \sim \boldsymbol{\pi}_1$,

$$(\mathcal{C}_0(\pi_1(x)), \mathcal{C}_0(\pi_1(z))) \triangleq (c, c')$$

satisfies

$$\left|(\boldsymbol{c}, \boldsymbol{c'}) - U_{\mathbb{F}_q \times \mathbb{F}_q}\right| \leq \sqrt{2 \ln q \cdot \alpha} \triangleq \xi_1. \tag{5}$$

We will set q in such a way that

$$\frac{n}{\log n} \geq \frac{2q}{\alpha \log q} = \frac{q^2}{c_\alpha \log^2 q},$$

so the premise of Lemma 3.21 indeed holds. Denote this good event (5) by E_{π_1}.

Fix some $i \in [m_2]$ and any $a \in \mathbb{F}_q$. Recall that x is fixed and fix some $z \neq x$. Fixing some $\pi_1 \sim \boldsymbol{\pi}_1$ for which $E_{\pi_1} = 1$, we have that

$$\Pr\left[\mathcal{C}^{\pi_1,\boldsymbol{\pi}_2}(z)[T]_i = a \mid \mathcal{C}^{\pi_1,\boldsymbol{\pi}_2}(x)[T]_i = a\right] = \\ \Pr_{\pi_2\sim\boldsymbol{\pi}_2}\left[\mathcal{C}_0(\pi_1(z))_{\pi_2^{-1}(\Gamma_G(T)_i)} = a \,\middle|\, \mathcal{C}_0(\pi_1(x))_{\pi_2^{-1}(\Gamma_G(T)_i)} = a\right] \le \frac{1}{q} + \xi_1,$$

as $\pi_2^{-1}(\Gamma_G(T)_i)$ is simply a random element of $[n]$.

Since $\Pr[E_{\boldsymbol{\pi}_1} = 1] \ge 1 - q^{-\alpha n}$, we get that, for any $a \in \mathbb{F}_q$,

$$\Pr\left[\mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(z)[T]_i = a \mid \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[T]_i = a\right] \le \frac{1}{q} + \xi_1 + q^{-\alpha n} \triangleq \xi.$$

By a union bound over all m_2 values $j \in T$, it follows that

$$\Pr\left[\exists j \in T, \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[j] = \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[j] \mid \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[T] = \sigma\right] \le m_2 \cdot \xi.$$

Thus, returning to the quantity Equation (4), using Markov's inequality and linearity of expectation,

$$\Pr\left[\sum_{z\in\mathcal{U}\setminus\{x\}} f^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(z) \cdot \mathbf{1}\left\{\exists j \in T, \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[j] = \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[j]\right\} > \frac{\varepsilon}{4}\|f\|_1\right] \\ \le \frac{4}{\varepsilon\|f\|_1} \cdot \sum_{z\in\mathcal{U}\setminus\{x\}} f^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(z) \cdot \Pr\left[\exists j \in T, \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[j] = \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[j] \mid \mathcal{C}^{\boldsymbol{\pi_1},\boldsymbol{\pi_2}}(x)[T] = \sigma\right] \\ \le \frac{4m_2\xi}{\varepsilon}.$$

We conclude from (4) that

$$\Pr\left[\max_{\sigma'\in S_\sigma} f_T^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(\sigma') \ge \frac{\varepsilon}{4}\|f\|_1\right] \le \frac{4m_2\xi}{\varepsilon}.$$

Observing that ξ is dominated by $\xi_1 = \sqrt{\alpha \cdot 2\ln q}$ and recalling that $\alpha = \frac{2c_\alpha \log q}{q}$, we conclude that

$$\Pr\left[\max_{\sigma'\in S_\sigma} f_T^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(\sigma') \ge \frac{\varepsilon}{4}\|f\|_1\right] \le \frac{c_\varepsilon \log q}{\varepsilon\sqrt{q}}$$

for some constant $c_\varepsilon(m_2) > 0$, and this proves the first statement.

Finally, we move on to discuss the "Moreover" part of the lemma. Once $\pi_1 \sim \boldsymbol{\pi}_1$ is fixed, the randomness is only over $\boldsymbol{\pi}_2$. Note that for every $\pi_2 \sim \boldsymbol{\pi}_2$, $\pi_2^{-1}(T)$ and $\pi_2^{-1}(T_i)$ are disjoint, for every $2 \le i \le t$.

Denote

$$\mathcal{I}_{\pi_2} = \pi_2^{-1}(T_2) \cup \ldots \cup \pi_2^{-1}(T_t).$$

We already established the fact that for most $\pi_1 \sim \boldsymbol{\pi}_1$, the empirical distribution $(\boldsymbol{c}, \boldsymbol{c'})$ is close to $U_{\mathbb{F}_q} \times U_{\mathbb{F}_q}$. Fix such a π_1. Now, given *any* $\pi_2 \sim \boldsymbol{\pi}_2$ and an assignment I to $\mathcal{C}^{\pi_1,\pi_2}(x)$ in $\mathcal{I}_{\pi_2}$, it holds that

$$\left|(\boldsymbol{c}, \boldsymbol{c'}) - [(\boldsymbol{c}, \boldsymbol{c'}) | \{\mathcal{C}^{\pi_1,\pi_2}(x) \text{ in } \mathcal{I}_{\pi_2} \text{ is } I\}]\right| \le \frac{tm_2}{n},$$

so altogether

$$\left|\left(\boldsymbol{c}, \boldsymbol{c'}, \mathcal{C}^{\pi_1,\boldsymbol{\pi}_2}(x)[\mathcal{I}_{\boldsymbol{\pi}_2}]\right) - \left(U_{\mathbb{F}_q}, U_{\mathbb{F}_q}, \mathcal{C}^{\pi_1,\boldsymbol{\pi}_2}(x)[\mathcal{I}_{\boldsymbol{\pi}_2}]\right)\right| \leq \xi_1 + \frac{tm_2}{n},$$

where the two copies of $U_{\mathbb{F}_q}$ are independent. Following the same outline as before, we get that, for any $i \in T$,

$$\Pr\left[\mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2'}(z)[T]_i = a \,\middle|\, \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2'}(x)[T]_i = a \wedge F_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,T_2,\ldots,T_t}(x)\right] \leq m_2\left(\frac{1}{q} + \xi_1 + \frac{tm_2}{n} + q^{-\alpha n}\right).$$

We will set q and ζ such that

$$\frac{tm_2}{n} \leq \frac{1}{q},$$

and in particular, the ξ_1 term still dominates the bound above. Thus we may continue with the proof as before, and the same bound holds. ■

Given Lemma 5.3, we can now say something about the probability that a "heavy hitter" $x \in \mathcal{U}$ that yields the symbol $\sigma \in \Sigma$ at T is confounded, in the sense that $\mathcal{L}_T^{\pi_1,\pi_2,r}$ contains some $\sigma' \in S_\sigma$. (Recall from Equation (3) that S_σ is the set of all $\sigma' \neq \sigma$ so that $\sigma_j = \sigma'_j$ for some $j \in [m_2]$.)

Lemma 5.4. *Fix $x \in \mathcal{U}$ such that $f(x) \geq \varepsilon\|f\|_1$ and fix $T \in V_2$. Let $\sigma = \mathcal{C}^{\pi_1,\pi_2}(x)[T]$. Then, with probability at least $1-\eta$ over $\pi_1 \sim \boldsymbol{\pi}_1$, $\pi_2 \sim \boldsymbol{\pi}_2$, and $r \sim \boldsymbol{r}$, we have $S_\sigma \cap \mathcal{L}_T^{\pi_1,\pi_2,r} = \emptyset$.*

Moreover, the above holds even conditioning on any event $F_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,\boldsymbol{r},T_2,\ldots,T_t}(x)$ whose randomness is over $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_2^{-1}(T_2), \ldots, \boldsymbol{\pi}_2^{-1}(T_t)$, and $\boldsymbol{r}_{T_2}, \ldots, \boldsymbol{r}_{T_t}$, where $T, T_2, \ldots, T_t \in V_2$ are pairwise disjoint.

Proof: Fix $\sigma \in \Sigma$, and in the rest of the analysis we condition on $\mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[T] = \sigma$ and $F = F_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,\boldsymbol{r},T_2,\ldots,T_t}(x)$, recalling that $\boldsymbol{r}_T$ is independent of $\boldsymbol{r}_{T_2}, \ldots, \boldsymbol{r}_{T_t}$. We can write

$$\Pr\left[\exists\sigma' \in S_\sigma \cap \mathcal{L}_T^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,\boldsymbol{r}} \,\middle|\, \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[T] = \sigma \wedge F\right] = \mathop{\mathbb{E}}_{(\pi_1,\pi_2)\sim\boldsymbol{\pi}_1\times\boldsymbol{\pi}_2,\boldsymbol{r}|\mathcal{C}^{\pi_1,\pi_2}(x)[T]=\sigma,F}\left[\Pr\left[\exists\sigma' \in S_\sigma \cap \mathcal{L}_T^{\pi_1,\pi_2,\boldsymbol{r}_T}\right]\right] \quad (6)$$

which is at most

$$\mathop{\mathbb{E}}_{(\pi_1,\pi_2)\sim\boldsymbol{\pi}_1\times\boldsymbol{\pi}_2,\boldsymbol{r}|\mathcal{C}^{\pi_1,\pi_2}(x)[T]=\sigma,F}\left[\Pr_{r\sim\boldsymbol{r}_T}\left[\exists\sigma' \in S_\sigma \cap \mathcal{L}_T^{\pi_1,\pi_2,r} \,\middle|\, T \text{ is good for } (\pi_1,\pi_2,r)\right] + \Pr_{r\sim\boldsymbol{r}_T}\left[T \text{ is not good for } (\pi_1,\pi_2,r)\right]\right].$$

Now, $\Pr[T \text{ is not good for } (\pi_1,\pi_2,\boldsymbol{r}_T)] \leq \delta_2$. For a good T, the event $\sigma' \in \mathcal{L}_T^{\pi_1,\pi_2,r}$ implies that $f_T(\sigma') \geq \frac{\varepsilon}{4}\|f\|_1$. Thus, Equation (6) is at most

$$\Pr\left[\exists\sigma' \in S_\sigma, f_T^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(\sigma') \geq \frac{\varepsilon}{4}\|f\|_1 \,\middle|\, \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[T] = \sigma, F \text{ holds, and } T \text{ is good for } (\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,\boldsymbol{r})\right] + \delta_2.$$

By Claim 2.10, the above is at most

$$\Pr\left[\exists\sigma' \in S_\sigma, f_T^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(\sigma') \geq \frac{\varepsilon}{4}\|f\|_1 \,\middle|\, \mathcal{C}^{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x)[T] = \sigma \wedge F\right] + 2\delta_2. \quad (7)$$

Fix the randomness $\boldsymbol{r}_{T_2}, \ldots, \boldsymbol{r}_{T_t}$, so now $F = F_{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, T_2, \ldots, T_t}(x)$. By Lemma 5.3, the first term is at most

$$\frac{c_\varepsilon \log q}{\varepsilon \cdot \sqrt{q}}.$$

Overall, as $\delta_2 \leq \frac{c_\varepsilon \log q}{2\varepsilon \cdot \sqrt{q}}$, Equation (7) is at most

$$\frac{c_\varepsilon \log q}{\varepsilon \cdot \sqrt{q}} + 2\delta_2 \leq \frac{2c_\varepsilon \log q}{\varepsilon \cdot \sqrt{q}},$$

and so it is also true averaging over all fixings of $\boldsymbol{r}_{T_2}, \ldots, \boldsymbol{r}_{T_t}$. This proves the lemma. ■

Definition 5.5. *Given $\pi_1 \sim \boldsymbol{\pi}_1$, $\pi_2 \sim \boldsymbol{\pi}_2$ and $r \sim \boldsymbol{r}$, we say $T \in V_2$ is* excellent *for (π_1, π_2, r) w.r.t. some $x \in \mathcal{U}$ if both conditions hold:*

1. *$\left\{\sigma \in \Sigma : f_T^{\pi_1,\pi_2}(\sigma) \geq \varepsilon \|f\|_1\right\} \subseteq \mathcal{L}_T^{\pi_1,\pi_2,r} \subseteq \left\{\sigma \in \Sigma : f_T^{\pi_1,\pi_2}(\sigma) \geq \frac{\varepsilon}{4}\|f\|_1\right\}$ (i.e., it is good).*
2. *Let $\sigma = \mathcal{C}^{\pi_1,\pi_2}(x)[T]$. Then for all $\sigma' \in S_\sigma$, $f_T^{\pi_1,\pi_2}(\sigma) < \frac{\varepsilon}{4}\|f\|_1$.*

We denote by $\mathsf{Exc}(T, x, \pi_1, \pi_2, r) \in \{0, 1\}$ the indicator which is 1 if and only if T is excellent for (π_1, π_2, r) w.r.t. x.

Note that if T is excellent w.r.t. x, then there is no $\sigma' \neq \sigma = \mathcal{C}^{\pi_1,\pi_2}(x)[T]$ in $\mathcal{L}_T^{\pi_1,\pi_2,r}$ such that $\sigma'_j = \sigma_j$ for some $j \in [m_2]$.

Combining Lemma 5.4 and Claim 5.2, we can then conclude the following.

Corollary 5.6. *Fix $x \in \mathcal{U}$ such that $f(x) \geq \varepsilon \|f\|_1$ and fix pairwise disjoint $T, T_2, \ldots, T_t \in V_2$. Then,*

$$\Pr\left[\mathsf{Exc}(T, x, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{r}) = 1\right] \geq 1 - 2\eta,$$

and furthermore, for every $b_2, \ldots, b_t \in \{0, 1\}$,

$$\Pr\left[\mathsf{Exc}(T, x, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{r}) = 1 \mid (\mathsf{Exc}(T_2, x, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{r}), \ldots, \mathsf{Exc}(T_t, x, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{r})) = (b_2, \ldots, b_t)\right] \geq 1 - 2\eta.$$

Proof: The first statement follows from a simple union bound. To see the second one, we observe that indeed

$$(\mathsf{Exc}(T_2, x, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{r}), \ldots, \mathsf{Exc}(T_t, x, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{r})) = (b_2, \ldots, b_t)$$

is an event whose randomness is over $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_2^{-1}(T_2), \ldots, \boldsymbol{\pi}_2^{-1}(T_t)$, and $\boldsymbol{r}_{T_2}, \ldots, \boldsymbol{r}_{T_t}$. ■

5.1 The Propagation Step

We now begin analyzing the propagation over the expander. Observe that as $\mathcal{T}$ is large, $|\mathcal{S}_0|$ is large too, in particular $|\mathcal{S}_0| = \Omega(\beta n)$ (which follows from Lemma 3.7), but we will not use this fact directly.

We proceed to analyzing the propagation iterations.

Lemma 5.7. *Fix any positive integer L. With probability at least $1 - 2^L \cdot 2^{-t}$ over $\pi_1 \sim \boldsymbol{\pi}_1$, $\pi_2 \sim \boldsymbol{\pi}_2$ and $r \sim \boldsymbol{r}$, the following holds. For every integer $0 \leq i \leq L$,*

$$|\mathcal{T}_{i+1}| \geq (1 + \mu_i) \cdot |\mathcal{T}_i|$$

for $\mu_i = \frac{1-\rho(\mathcal{T}_i)}{4Cm_2^2}$.

Proof: At each iteration i we condition on the previous iteration being successful, i.e., $|\mathcal{T}_i| \geq (1+\mu_i)|\mathcal{T}_{i-1}|$. By Claim 2.10, we can analyze each iteration and then aggregate the error. Formally, we show by induction that for every i,

$$\Pr\left[|\mathcal{T}_{i+1}| \geq (1+\mu_i)\,|\mathcal{T}_i| \mid \forall j < i, |\mathcal{T}_{j+1}| \geq (1+\mu_j)\,|\mathcal{T}_j|\right] \geq 1 - 2^i \cdot 2^{-t}.$$

Setting $\mathcal{T}_{-1} = \{T^\star\}$, the inductive claim is immediate for $i = 0$. Fix some iteration $i \in [L]$, for L to be determined later, and assume the claim holds for iteration $i-1$. In what follows, condition on the event that $|\mathcal{T}_j| \geq (1+\mu_{j-1})\,|\mathcal{T}_{j-1}|$ for all $j \leq i$.

For $T \in \mathcal{F}_i = \mathcal{F}_i^{\pi_1,\pi_2,r}$, let X_T be the indicator random variable (depending on $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_2$ and $\boldsymbol{r}$), that is 1 if and only if T does *not* get added to $\mathcal{T}_{i+1} = \mathcal{T}_{i+1}^{\pi_1,\pi_2,r}$. Let $\widetilde{\mathcal{F}}_i \subseteq \mathcal{F}_i$ be a maximal set so that for all $T, T' \in \widetilde{\mathcal{F}}_i$ it holds that $\Gamma_G(T) \cap \Gamma_G(T') = \emptyset$. Such a set exists with

$$\left|\widetilde{\mathcal{F}}_i\right| \geq \frac{1}{Cm_2^2} \cdot |\mathcal{F}_i|$$

by considering the algorithm that greedily takes a set $T \in \mathcal{F}_i$ and excludes the at most Cm_2^2 T'-s that overlap with it (note that Cm_2 is the left-degree of G). By Corollary 5.6, we can use Theorem 2.9 on the indicators $\{X_T\}_{T \in \widetilde{\mathcal{F}}_i}$, and we know that $\mathbb{E}_{\boldsymbol{\pi_1},\boldsymbol{\pi_2},\boldsymbol{r}}[X_T] \leq 2\eta \leq \frac{1}{4}$ since being excellent implies that we add T to $\mathcal{T}_{i+1}$. Applying Theorem 2.9 with $\delta = 1$, we get that

$$\Pr_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2,\boldsymbol{r}}\left[\sum_{T \in \widetilde{\mathcal{F}}_i} X_T > \frac{1}{2}\left|\widetilde{\mathcal{F}}_i\right|\right] \leq e^{-t/2},$$

where we used the fact that $\left|\widetilde{\mathcal{F}}_i\right| \gg t$. To see this, note that $\left|\widetilde{\mathcal{F}}_i\right| \geq |\mathcal{T}_0| = \frac{\alpha'}{m_2 - m_1} n = \Omega(n/q)$, $t = n^{1-\zeta}$, and we will set the parameters such that $q \ll n^{\zeta}$.

Next, we lower bound the size of $|\mathcal{F}_i|$ using the expansion properties of G_2. By Tanner's inequality (Theorem 2.6), we have

$$\begin{aligned}
|\Gamma_G(\mathcal{S}_i)| &\geq |\mathcal{T}_i| \cdot \frac{1}{\rho(\mathcal{T}_i) + (1-\rho(\mathcal{T}_i)) \cdot \frac{4}{m_2^2}} \\
&\geq |\mathcal{T}_i| \cdot \left(1 + (1-\rho(\mathcal{T}_i)) \cdot \left(1 - \frac{4}{m_2}\right)\right) \geq |\mathcal{T}_i| + \frac{1-\rho(\mathcal{T}_i)}{2} \cdot |\mathcal{T}_i|,
\end{aligned}$$

where we used the fact that $m_2 \geq 8$. Thus,

$$|\mathcal{F}_i| \geq \frac{1-\rho(\mathcal{T}_i)}{2} \cdot |\mathcal{T}_i|,$$

and when the favorable case happens (that is, when $\sum_{T \in \tilde{\mathcal{F}}_i} X_T \leq \frac{1}{2}|\tilde{\mathcal{F}}_i|$),

$$|\mathcal{T}_{i+1}| \geq |\mathcal{T}_i| + \frac{1}{2}\left|\widetilde{\mathcal{F}}_i\right| \geq |\mathcal{T}_i| + \frac{1-\rho(\mathcal{T}_i)}{4Cm_2^2} \cdot |\mathcal{T}_i|.$$

By Claim 2.10, we get that the probability that for every $j \leq i+1$, $|\mathcal{T}_j| \geq (1+\mu_{j-1})\,|\mathcal{T}_{j-1}|$ is at least

$$1 - 2^i \cdot 2^{-t} - 2^{-t} = 1 - 2^{i+1} \cdot 2^{-t},$$

as desired. ■

Finally, we wish to bound L, the number of iterations needed (with high probability) until we can unique decode. We remark that the reason to bound L is only for the analysis of the failure probability; the running time of the algorithm is independent of L, because we touch each vertex at most once. For the claim below, we recall that β is defined in Equation (1) and is defined so that $|\mathcal{T}| = \beta|V_2|$; and τ is defined in Equation (2) and is the fraction of erasures that the code $\mathcal{C}_0''$ (and hence $\mathcal{C}_0$) can handle.

Claim 5.8. *With probability at least $1 - 2^{-t+L}$ over $\pi_1 \sim \boldsymbol{\pi}_1$, $\pi_2 \sim \boldsymbol{\pi}_2$ and $r \sim \boldsymbol{r}$, the propagation process of the query procedure stops (i.e., breaks the loop in Item 3d) within $L = O\left(\tau^{-2}\log(1/\beta)\right) = \mathrm{poly}(q)$ iterations.*

Proof: Let us first find the smallest $\rho(\mathcal{T}_{i+1})$ for which $|\Gamma_G(\mathcal{T}_{i+1})| \geq (1-\tau)n$. By Tanner's inequality for bipartite graphs (Lemma 3.8), we know that

$$\rho(\mathcal{S}_{i+1}) \geq \frac{1}{\rho(\mathcal{T}_{i+1}) + \sqrt{\frac{2}{m_2}}\,(1-\rho(\mathcal{T}_{i+1}))} \cdot \rho(\mathcal{T}_{i+1}).$$

Thus, we can keep propagating until the first time that

$$\rho(\mathcal{T}_{i+1}) \geq \frac{\sqrt{\frac{2}{m_2}}(1-\tau)}{\tau + \sqrt{\frac{2}{m_2}}(1-\tau)}.$$

We make use of the following claim:

Claim 5.9. *It holds that*

$$\frac{\sqrt{\frac{2}{m_2}}(1-\tau)}{\tau + \sqrt{\frac{2}{m_2}}(1-\tau)} \leq 1 - \frac{\sqrt{m_2}\cdot\tau^2}{\sqrt{2}}$$

Proof: For brevity, denote $a = \sqrt{\frac{m_2}{2}} > 1$. Rearranging terms, we want to show that

$$\frac{1-\tau}{(a-1)\tau+1} \leq 1 - a\tau^2.$$

The functions $f(\tau^\star) = \frac{1-\tau^\star}{(a-1)\tau^\star+1}$ and $g(\tau^\star) = 1 - a\tau^{\star 2}$ are monotonically decreasing for $\tau \geq 0$, and $f(0) = g(0) = 1$. They then intersect at

$$\tau_0 = \frac{-1+\sqrt{4a-3}}{2(a-1)}$$

and $f(\tau^\star) \leq g(\tau^\star)$ for $\tau^\star \in [0, \tau_0]$. m_2 was chosen such that $a \leq 3$, so $\tau_0 \geq \frac{1}{2}$, and indeed τ will be smaller than $\frac{1}{2}$. ■

By Lemma 5.7 and the above claim, it is sufficient to choose L so that

$$\left(1 + \frac{\frac{\sqrt{m_2}\cdot\tau^2}{\sqrt{2}}}{4Cm_2^2}\right)^L \cdot \beta \geq 1 - \frac{\sqrt{m_2}\cdot\tau^2}{\sqrt{2}},$$

and we get

$$L \leq \frac{4\sqrt{2}Cm_2^2}{\sqrt{m_2}\cdot\tau^2}\log\frac{1}{\beta},$$

as desired. ■

5.2 Determining the Parameters

We now set q, ζ, and establish a lower bound on ε. Collecting all constraints, we need to satisfy the following.

1. There exists a universal constant c_q such that $q \geq c_q$.
2. $\frac{n}{\log n} \geq \frac{q^2}{c_\alpha \log^2 q}$.
3. $\frac{tm_2}{n} \leq \frac{1}{q}$, implying that $q \leq \frac{1}{8} n^{\zeta}$.
4. $\frac{c_\eta \log q}{\varepsilon \cdot \sqrt{q}} \leq \frac{1}{8}$.
5. $q \ll n^{\zeta}$.

Fix $q = \max\left\{c_q, \frac{1}{\varepsilon^{2.25}}\right\}$.[15] Item 4 readily holds, possibly after increasing c_q. For Item 2 to hold, we should require $\varepsilon \geq n^{-1/2.25}$.[16] Setting ζ such that $q \leq n^{0.8\zeta}$, Items 3 and 5 hold. Combining these two requirements, we get $\varepsilon \geq n^{-16\zeta/45}$. Minimizing ζ, we set

$$\zeta = \frac{45 \log(1/\varepsilon)}{16 \log n}.$$

The probability bound of Claim 5.8 then becomes

$$2^{L-t/2} \leq 2^{-t/4} = 2^{-\Theta(\varepsilon^{2.8125} n)} \leq N^{-\Theta(\varepsilon^{2.9})}. \tag{8}$$

Note further that for $P \triangleq \max\left\{1, \Theta\left(\frac{\log(1/\delta)}{\varepsilon^{2.9}\log N}\right)\right\}$,

$$N^{-P\cdot\Theta(\varepsilon^{2.9})} \leq \frac{\delta}{3}, \tag{9}$$

and we will use it soon.

5.3 Finishing the Proof

Everything is in place to establish the correctness of our algorithm, followed by a runtime analysis.

Theorem 5.10. *With probability at least $1 - N^{-\Theta(\varepsilon^3)}$, the output list $\mathcal{L} = \mathcal{L}^{\pi_1,\pi_2,r}$ of our query procedure satisfies*

$$\{x \in \mathcal{U} : f(x) \geq \varepsilon \|f\|_1\} \subseteq \mathcal{L} \subseteq \left\{x \in \mathcal{U} : f(x) \geq \frac{\varepsilon}{4}\|f\|_1\right\}.$$

where the randomness is over $\pi_1 \sim \boldsymbol{\pi}_1$, $\pi_2 \sim \boldsymbol{\pi}_2$, $r \sim \boldsymbol{r}$ and the randomness of the preprocessing step.

Proof: Let $\tilde{\delta} = N^{-c\varepsilon^{2.9}}$ for a small enough constant $c > 0$ implied by Equation (8), and assume without loss of generality that $\delta \leq \tilde{\delta}$. Fix $x \in \mathcal{L}$. First, we observe that with probability at least $1 - \tilde{\delta}$, all of the following events occur.

[15] The choice of $\varepsilon^{2.25}$ is for concreteness. In fact, one can replace it with $\varepsilon^{2+\gamma}$ for some carefully chosen $\gamma = o(1)$.
[16] Here too, one can replace $-\frac{1}{2.25}$ with $-\frac{1}{2} - \gamma$ for a small γ.

- The preprocessing step was successful (that is, the expander G'' is good enough).
- There exists $T^\star \in \mathcal{T}$ for which $|\mathcal{L}_{T^\star}| \le \frac{4}{\varepsilon}$. The latter follows from the fact that $\delta_2^{|\mathcal{T}|} \le \tilde{\delta}$, which holds assuming q is at least a large enough constant.
- All 1-query calls of CMS were successful. The latter follows from a simple union bound, recalling that $\delta_1 = \frac{\delta}{3|\Sigma|}$, where δ_1 is the failure probability of the large CMS instance as in the beginning of Section 4.

We consider two cases.

1. x is such that $f(x) \le \frac{\varepsilon}{4}\|f\|_1$. Notice that we only add x to $\mathcal{L}$ if we applied the update procedure of CMS on x and the 1-query procedure of CMS declares x to be a heavy hitter. As the 1-query procedure of CMS is successful, $\hat{f}(x) < \varepsilon\|f\|_1$ and we do not add it to our output list.
2. x is such that $f(x) \ge \varepsilon\|f\|_1$. Setting $\sigma_x = \mathcal{C}(x)[T^\star]$, we know that $f_{T^\star}(\sigma_x) \ge f(x) \ge \varepsilon\|f\|_1$. As $\|f\|_1 = \|f_{T^\star}\|_1$, by our promise on $HH_{T^\star}$, we get that $\sigma_x \in \mathcal{L}_{T^\star}$. By the analysis above, with probability at least $1-\tilde{\delta}$, our propagation step succeeds and we output $\hat{x} = x$. As CMS succeeds, we add x to our output list.

The theorem follows from a union bound over at most $\frac{4}{\varepsilon}$ elements. ■

To reduce the failure probability to our arbitrary $\delta > 0$, we duplicate our workspace P times, repeat each update P times and perform the query procedure P times as well. Note that we do not need to duplicate the CMS instance CMS, but just to reduce its failure probability by a negligible multiplicative factor of $\frac{1}{P}$. Having P output lists, we can choose the one with at most $\frac{4}{\varepsilon}$ elements (conditioned on all CMS 1-query calls being successful). With probability at least $1-\frac{2\delta}{3}$, both the CMS calls and the preprocessing step is successful, and by Equation (9) we reach our designated success probability of $1-\delta$.

Having established correctness, plugging in q we can collect Lemmas 4.1 to 4.3 to the following theorem.

Theorem 5.11. *There exists a constant $c > 0$ such that the following holds for any positive integer N, $\delta > 0$ and $\varepsilon \ge (\log N)^{-0.4}$. Set $P = \max\left\{1, \frac{c\log(1/\delta)}{\varepsilon^3 \log N}\right\}$. Then, there exists an algorithm that maintains $f \in \mathbb{R}^N$ in the strict turnstile model, after a preprocessing step which takes $\operatorname{poly}(\log N) + \log(1/\delta)$ time, and supports the following procedures using space $O\left(P \cdot \frac{\log(N/\delta)}{\varepsilon}\right)$.*

1. *An update, which is done in time $O\left(P \cdot \log \frac{N}{\delta}\right)$.*
2. *An ε-HH query, which is done in time*

$$O\left(P \cdot \frac{\operatorname{polylog}(1/\varepsilon)}{\varepsilon} \cdot \log N\right)$$

   with failure probability δ. More specifically, with probability at least $1-\delta$, the query procedure outputs a list $\mathcal{L} \subseteq [N]$ such that $|\mathcal{L}| \le O(1/\varepsilon)$, and for all $x \in \mathcal{U}$ so that $f(x) \ge \varepsilon\|f\|_1$, $x \in \mathcal{L}$.

In particular, choosing the parameter P, we have the following statements:

1. *For $\delta = N^{-\Theta(\varepsilon^3)}$, the space requirement is $O\left(\frac{1}{\varepsilon}\log N\right)$, the update time is $O(\log N)$, and the query time is $O\left(\frac{\mathrm{polylog}(1/\varepsilon)}{\varepsilon}\log N\right)$.*

2. *For $\delta = \frac{1}{\mathrm{poly}(N)}$, the space requirement is $O\left(\frac{1}{\varepsilon^4}\log N\right)$, the update time is $O(\frac{1}{\varepsilon^3}\log N)$, and the query time is $O\left(\frac{1}{\varepsilon^4}\log N\right)$.*

6 Randomized List Recovery

Inspecting the proof in Section 5, we can extract a list recovery result for our (randomized) code $\mathcal{C}$ that tolerates a small fraction of erasures. Our randomized encoding can handle input lists that come from a union of codewords $\{\boldsymbol{\mathcal{C}}(x) : x \in \mathcal{L}_0\}$ for some $\mathcal{L}_0 \subseteq \mathbb{F}_q^k$; this is what we stated in Theorem 1.2. Moreover, our algorithm can also handle some extra "distractor symbols," provided that those symbols are randomized and unlikely to collide with the symbols that come from $\mathcal{L}_0$. In order to state this formally, we first give a definition that captures the sort of input lists that our algorithm can handle.

Definition 6.1. *Let $\boldsymbol{\mathcal{C}}\colon \mathbb{F}_q^k \to \Sigma^{n'}$ be a randomized encoding, for $\Sigma = \mathbb{F}_q^b$. Consider a randomized function of $\mathcal{C} \sim \boldsymbol{\mathcal{C}}$ and a set of messages $\mathcal{L}_0 \subseteq \mathbb{F}_q^k$, that outputs lists $\mathcal{L}_1, \ldots, \mathcal{L}_{n'} \subseteq \Sigma$. We say that such a function is (t,η)-*nice *w.r.t. $\boldsymbol{\mathcal{C}}$ if the following holds for all $\mathcal{L}_0 \subseteq \mathbb{F}_q^k$ (note that the lists $\mathcal{L}_i$ can depend on $\mathcal{L}_0$):*

1. *For any $i \in [n']$,*
   - *with probability at least $1-\eta$, $|\mathcal{L}_i| = O(|\mathcal{L}_0|)$, and,*
   - *with probability 1, $\boldsymbol{\mathcal{C}}(x)_i \in \mathcal{L}_i$ for all $x \in \mathcal{L}_0$.*
2. *For any $x \in \mathcal{L}_0$ and $i \in [n']$, with probability at least $1-\eta$ it holds that $(\boldsymbol{\mathcal{C}}(x)_i)_j \neq \sigma_j$ for every $j \in [b]$ and $\sigma \in \mathcal{L}_i \setminus \{\boldsymbol{\mathcal{C}}(y)_i : y \in \mathcal{L}_0\}$.*

Furthermore, we require that the above properties should hold t-wise independently across the lists. Namely, for any $x \in \mathcal{L}_0$, whether (1) and (2) hold for some $i \in [n']$ is independent of whether it holds for any $t-1$ other values of $i' \in [n']$.

To illustrate this definition, we give a few examples.

Example 6.2 (Lists from a union of codewords)**.** *The simplest example of a nice distribution is the function that gives*

$$\mathcal{L}_i = \{\boldsymbol{\mathcal{C}}(x)_i : x \in \mathcal{L}_0\}\,.$$

That is, the lists $\mathcal{L}_i$ are just given by the union of the codewords in $\mathcal{L}_0$. To see that this is $(\eta = 0, t = n')$-nice, observe that both (1) and (2) hold deterministically, with probability 1. Indeed, (1) holds by construction, and (2) holds because there are no $\sigma \in \mathcal{L}_i \setminus \{\boldsymbol{\mathcal{C}}(y)_i : y \in \mathcal{L}_0\}$, so the condition is trivial.

Example 6.3 (Lists with random distractor symbols)**.** *Another natural example of a nice distribution is the example above, with some uniformly random extra "distractor" symbols. That is,*

$$\mathcal{L}_i = \{\boldsymbol{\mathcal{C}}(x)_i : x \in \mathcal{L}_0\} \cup \{\sigma_{i,h} : h \in [r]\}$$

where $r > 0$ is some parameter and where $\sigma_{i,h}$ are i.i.d. and uniform in Σ. Again, this satisfies item (1) deterministically, provided that $r = O(|\mathcal{L}_0|)$. For (2), we can compute the probability of a collision between the distractor symbols $\{\sigma_{i,j} : j \in [r]\}$ and a given codeword $\boldsymbol{\mathcal{C}}(x)$ for $x \in \mathcal{L}_0$:

$$\Pr\left[(\boldsymbol{\mathcal{C}}(x)_i)_j \neq \sigma_j \ \forall j \in [b], \sigma \in \{\sigma_{i,h} : h \in [r]\}\right] = \left(1 - \frac{1}{q}\right)^{br} \leq \exp(-br/q).$$

In particular, when $q \gg br$, this is $1 - O\left(\frac{br}{q}\right)$. Thus, this distribution is (η, t)-nice where $\eta = O(br/q)$ and $t = n$.

Finally, we note that the distribution of distractor symbols that arises in our heavy hitters application is also nice for the code $\boldsymbol{\mathcal{C}}$ that we use. The first point of Item (1) holds because the lists $\mathcal{L}_i$ can only become too large if the inner InnerHH fails and includes items that are not $\varepsilon/4$-heavy hitters. The second point of Item (1) holds because our instantiation of InnerHH has only one-sided error. Item (2) holds even for any $\sigma \in \mathcal{L}_i \setminus \{\boldsymbol{\mathcal{C}}(x)_i\}$, which follows from Lemma 5.4.

With this definition in place, we can now state our main theorem for list-recovery. Theorem 6.4 generalized Theorem 1.2, because it allows for input lists with some extra "distractor" symbols, as per Definition Definition 6.1.

Theorem 6.4. *There exist constants $c > 1$ and $\gamma \in (0, 1)$ such that the following holds for any positive integers k and $\ell \leq k^\gamma$. There exists a randomized encoding $\boldsymbol{\mathcal{C}} : \mathbb{F}_q^k \to \Sigma^{n'}$, for $q = \mathrm{poly}(\ell)$, $\Sigma = \mathbb{F}_q^{O(1)}$ and $n' = \Theta(k)$, and a randomized list recovery algorithm $\mathcal{A}$ running in time $\ell^c \cdot k$, with the following guarantee.*

For some constant $\eta < 1$, and an integer $t = \frac{k}{\mathrm{poly}(\ell)}$, for any list of messages $\mathcal{L}_0 \subseteq \mathbb{F}_q^k$ of size ℓ, and any distribution over input lists $\mathcal{L}_1, \ldots, \mathcal{L}_{n'}$ to $\mathcal{A}$ which are randomized functions of $\boldsymbol{\mathcal{C}}$ and $\mathcal{L}_0$ and are (t, η)-nice w.r.t. $\boldsymbol{\mathcal{C}}$, the list recovery algorithm $\mathcal{A}$, with probability $1 - \ell^{-\Omega(k)}$ (over the randomness of the encoding and the lists), outputs $\mathcal{L} \subseteq \mathbb{F}_q^k$ of size $O(\ell)$ such that $\mathcal{L}_0 \subseteq \mathcal{L}$. Furthermore, the encoding time of $\mathcal{C}$ is $O(k \log \ell)$, with a preprocessing step which takes $\mathrm{poly}(k)$ time.

We stress that unlike in standard state-of-the-art efficient list recovery algorithms, here we have a good dependence on ℓ, namely $q = \mathrm{poly}(\ell)$ and $|\mathcal{L}| = O(\ell)$.

We hope that Theorem 6.4 will find more applications. As discussed in Section 1.3, there are many algorithmic applications of list-recovery in the literature, and several previous applications of list-recovery have ended up with sub-optimal parameters due to the unavailability of codes that achieve Goal 1.1. It seems possible that Theorem 6.4 (or a further improvement on our techniques) could lead to improved results in (non-uniform or "for-each") group testing or compressed sensing.

Acknowledgements

We would like to thank Jelani Nelson and Amnon Ta-Shma for helpful conversations. We thank Mahdi Cheraghchi, Venkat Guruswami, and Badih Ghazi for pointing out relevant related work. We also thank anonymous reviewers for helpful comments and for pointing out related works.

References

[ABN$^+$92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on information theory*, 38(2):509–516, 1992.

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

[BADTS20] Avraham Ben-Aroya, Dean Doron, and Amnon Ta-Shma. Near-optimal erasure list-decodable codes. In *35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[BDH18] Gerandy Brito, Ioana Dumitriu, and Kameron Decker Harris. Spectral gap in random bipartite biregular graphs and applications. *arXiv preprint arXiv:1804.07808*, 2018.

[BNS19] Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. *ACM Transactions on Algorithms (TALG)*, 15(4):1–40, 2019.

[CM05] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[CN20] Mahdi Cheraghchi and Vasileios Nakos. Combinatorial group testing and sparse recovery schemes with near-optimal decoding time. In *61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*. IEEE, 2020. To appear.

[CS04] I. Csiszár and P. C. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528, 2004.

[DD19] Yotam Dikstein and Irit Dinur. Agreement testing theorems on layered set systems. In *60th Annual Symposium on Foundations of Computer Science (FOCS 2019)*, pages 1495–1524. IEEE, 2019.

[DDFH18] Yotam Dikstein, Irit Dinur, Yuval Filmus, and Prahladh Harsha. Boolean function analysis on high-dimensional expanders. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116, pages 38:1–38:20. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018.

[DDHRZ20] Yotam Dikstein, Irit Dinur, Prahladh Harsha, and Noga Ron-Zewi. Locally testable codes via high-dimensional expanders. *arXiv preprint arXiv:2005.01045*, 2020.

[DHK$^+$19] Irit Dinur, Prahladh Harsha, Tali Kaufman, Inbal Livni Navon, and Amnon Ta-Shma. List decoding with double samplers. In *ACM-SIAM 38th Annual Symposium on Discrete Algorithms (SODA 2019)*, pages 2134–2153. SIAM, 2019.

[DK17] Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *58th Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 974–985. IEEE, 2017.

[DMOZ20] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. In *61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 1057–1068. IEEE, 2020.

[GI04] Venkatesan Guruswami and Piotr Indyk. Linear-time list decoding in error-free settings. In *International Colloquium on Automata, Languages, and Programming (ICALP 2004)*, pages 695–707. Springer, 2004.

[GK16] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016.

[GLPS17] Anna C. Gilbert, Yi Li, Ely Porat, and Martin J. Strauss. For-all sparse recovery in near-optimal time. *ACM Transactions on Algorithms (TALG)*, 13(3):1–26, 2017.

[GNP$^+$13] Anna C. Gilbert, Hung Q. Ngo, Ely Porat, Atri Rudra, and Martin J. Strauss. ℓ_2/ℓ_2-foreach sparse recovery with low risk. In *International Colloquium on Automata, Languages, and Programming (ICALP 2013)*, pages 461–472. Springer, 2013.

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

[Gra11] Robert M. Gray. *Entropy and Information Theory*. Springer Publishing Company, Incorporated, 2nd edition, 2011.

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *39th Annual Symposium on Foundations of Computer Science (FOCS 1998)*, pages 28–37. IEEE, 1998.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *Journal of the ACM (JACM)*, 56(4):1–34, 2009.

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

[GX12] Venkatesan Guruswami and Chaoping Xing. Folded codes from function field towers and improved optimal rate list decoding. In *44th Annual Symposium on Theory of Computing (STOC 2012)*, pages 339–350. ACM, 2012.

[GX13] Venkatesan Guruswami and Chaoping Xing. List decoding Reed-Solomon, algebraic-geometric, and Gabidulin subcodes up to the Singleton bound. In *45th Annual Symposium on Theory of Computing (STOC 2012)*, pages 843–852. ACM, 2013.

[HRZW19] Brett Hemenway, Noga Ron-Zewi, and Mary Wootters. Local list recovery of high-rate tensor codes and applications. *SIAM Journal on Computing*, pages FOCS17–157, 2019.

[HW18] Brett Hemenway and Mary Wootters. Linear-time list recovery of high-rate expander codes. *Information and Computation*, 261:202–218, 2018.

[IK10] Russell Impagliazzo and Valentine Kabanets. Constructive proofs of concentration bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2010)*, pages 617–631. Springer, 2010.

[INR10] Piotr Indyk, Hung Q. Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *ACM-SIAM 21st Annual Symposium on Discrete Algorithms (SODA 2010)*, pages 1126–1142. SIAM, 2010.

[JST11] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for ℓ_p samplers, finding duplicates in streams, and related problems. In *ACM SIGMOD-SIGACT-SIGART 30th Annual Symposium on Principles of Database Systems*, pages 49–58, 2011.

[KM17] Tali Kaufman and David Mass. High dimensional random walks and colorful expansion. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[KO20] Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. *Combinatorica*, pages 1–37, 2020.

[Kop15] Swastik Kopparty. List-decoding multiplicity codes. *Theory of Computing*, 11(1):149–182, 2015.

[KRZSW18] Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved decoding of folded Reed-Solomon and multiplicity codes. In *59th Annual Symposium on Foundations of Computer Science (FOCS 2018)*, pages 212–223. IEEE, 2018.

[KS64] William Kautz and Roy Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10(4):363–377, 1964.

[KS12] Kazuki Kobayashi and Tomoharu Shibuya. Generalization of Lu's linear time encoding algorithm for LDPC codes. In *2012 International Symposium on Information Theory and its Applications*, pages 16–20. IEEE, 2012.

[LNNT16a] Kasper Green Larsen, Jelani Nelson, Huy L. Nguyễn, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *57th Annual Symposium on Foundations of Computer Science (FOCS 2016)*, pages 61–70. IEEE, 2016.

[LNNT16b] Kasper Green Larsen, Jelani Nelson, Huy L. Nguyễn, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. arxiv:1604.01357 [cs.DS], 2016.

[LNW18] Yi Li, Vasileios Nakos, and David P. Woodruff. On low-risk heavy hitters and sparse recovery schemes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116, pages 19:1–19:13. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018.

[LSV05a] Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Explicit constructions of ramanujan complexes of type $\tilde{A}_d$. *European Journal of Combinatorics*, 26(6):965–993, 2005.

[LSV05b] Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Ramanujan complexes of type $\tilde{A}_d$. *Israel Journal of Mathematics*, 149(1):267–299, 2005.

[MG82] Jayadev Misra and David Gries. Finding repeated elements. *Science of computer programming*, 2(2):143–152, 1982.

[NPR11] Hung Q. Ngo, Ely Porat, and Atri Rudra. Efficiently decodable error-correcting list disjunct matrices and applications. In *International Colloquium on Automata, Languages, and Programming (ICALP 2011)*, pages 557–568. Springer, 2011.

[NPR12] Hung Q. Ngo, Ely Porat, and Atri Rudra. Efficiently decodable compressed sensing by list-recoverable codes and recursion. In *29th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14, pages 230–241. LIPIcs, 2012.

[NPRR18] Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-case optimal join algorithms. *Journal of the ACM (JACM)*, 65(3):1–40, 2018.

[PS97] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff–Hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

[PS20] Eric Price and Jonathan Scarlett. A fast binary splitting approach to non-adaptive group testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[RR99] Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In *31st Annual Symposium on Theory of Computing (STOC 1999)*, pages 159–168, 1999.

[SS96] Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996.

[SSS95] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff–Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.

[Tan81] R. Michael Tanner. A recursive approach to low complexity codes. *IEEE Transactions on information theory*, 27(5):533–547, 1981.

[Tan84] R. Michael Tanner. Explicit concentrators from generalized n-gons. *SIAM Journal on Algebraic Discrete Methods*, 5(3):287–293, 1984.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM (JACM)*, 48(4):860–879, 2001.

[TSZ04] Amnon Ta-Shma and David Zuckerman. Extractor codes. *IEEE Transactions on Information Theory*, 50(12):3015–3025, 2004.

[VZGG13] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, USA, 3rd edition, 2013.

[Zém01] Gillés Zémor. On expander codes. *IEEE Transactions on Information Theory*, 47(2):835–837, 2001.

A Deferred Proofs

A.1 Proof of Claim 2.10

Proof: It holds that

$$\Pr[A|E] - \Pr[A] = \frac{\Pr[A \wedge E](1-\Pr[E])}{\Pr[E]} - \Pr[A \wedge \neg E].$$

Thus, on the one hand,

$$\frac{\Pr[A \wedge E](1-\Pr[E])}{\Pr[E]} - \Pr[A \wedge \neg E] \le \frac{\Pr[A \wedge E](1-\Pr[E])}{\Pr[E]} \le 1 - \Pr[E] \le \varepsilon,$$

and on the other hand,

$$\frac{\Pr[A \wedge E](1-\Pr[E])}{\Pr[E]} - \Pr[A \wedge \neg E] \ge -\Pr[A \wedge \neg E] \ge -\Pr[\neg E] \ge -\varepsilon.$$

■

A.2 Proof of Lemma 3.8

Proof: For brevity, denote $\rho = \rho(S)$. Let $M \colon \mathbb{R}^L \to \mathbb{R}^R$ denote the bipartite adjacency operator, and let $D \in \mathbb{R}^{N \times N}$ denote the matrix corresponding to the two-step random walk operator $M^\dagger M$. By our assumption, $\lambda_2(D) \le \lambda$. Let $\{(\lambda_i, v_i)\}_{i \in [N]}$ be the orthonormal basis of D with respect to the inner-product defined by $\mathcal{D}_R$. As $\mathcal{D}_R$ is uniform, we will use the standard inner product.

Let χ be the characteristic vector of S. There exist $\alpha_1, \ldots, \alpha_N$ such that $\chi = \sum_{i \in [N]} \alpha_i v_i$, for $\alpha_i = \langle \chi, v_i \rangle$. Thus, $\chi^\dagger M^\dagger M \chi = \sum_{i \in [N]} \alpha_i^2 \lambda_i$. We also know that $\lambda_1 = 1$ and $v_1 = \frac{1}{\sqrt{N}}\mathbf{1}$ where where $\mathbf{1}$ is the all-ones vector in $\mathbb{R}^N$. Thus, $\alpha_1 = \langle \chi, v_1 \rangle = \frac{|S|}{\sqrt{N}} = \sqrt{N}\rho$. We can then write

$$\chi^\dagger M^\dagger M \chi \le \alpha_1^2 + \lambda \sum_{i=2}^{N} \alpha_i^2 = \alpha_1^2 + \lambda\left(\langle \chi, \chi \rangle - \alpha_1^2\right) = N\rho^2 + \lambda \rho N (1-\rho), \tag{10}$$

observing that $\langle \chi, \chi \rangle = |S| = \rho N$.

We now bound $\chi^\dagger M^\dagger M \chi$ from below. The quantity $\chi^\dagger M^\dagger M \chi$ measures the weighted sum of paths of length 2 on G that start and end in S. Namely,

$$\chi^\dagger M^\dagger M \chi = \sum_{x \in S} \sum_{v \in \Gamma(S)} \sum_{y \in S} M[v,x] M^\dagger[y,v].$$

Write $d_S(v) = \sum_{x \in S} \Pr[\mathcal{D}_R = x \mid \mathcal{D}_L = v]$ and $d_S^\dagger(v) = \sum_{x \in S} \Pr[\mathcal{D}_L = v \mid \mathcal{D}_R = x]$. Thus,

$$\chi^\dagger M^\dagger M \chi = \sum_{v \in \Gamma(S)} d_S(v) \cdot d_S^\dagger(v).$$

For sanity check, note that for G with all the edge weights being identical, $d_S(v) = \frac{|S \cap \Gamma(v)|}{D}$ and $d_S^\dagger(v) = \frac{|S \cap \Gamma(v)|}{d}$. By Cauchy-Schwarz,

$$\chi^\dagger M^\dagger M \chi \ge \frac{1}{|\Gamma(S)|} \left(\sum_{v \in \Gamma(S)} \sqrt{d_S(v) \cdot d_S^\dagger(v)} \right)^2.$$

As both $\mathcal{D}_R$ and $\mathcal{D}_L$ are uniform, we can write

$$d_S(v) \cdot d_S^\dagger(v) = \sum_{x \in S} n \cdot W((x,v)) \sum_{y \in S} N \cdot W((y,v)) = nN \left(\sum_{x \in S \cap \Gamma(v)} W((x,v)) \right)^2,$$

so

$$\chi^\dagger M^\dagger M \chi \geq \frac{1}{|\Gamma(S)|} \left(\sum_{v \in \Gamma(S)} \sqrt{nN} \sum_{x \in S \cap \Gamma(v)} W((x,v)) \right)^2 = \frac{nN}{|\Gamma(S)|} \left(\sum_{x \in S} \sum_{i \in [d]} W((x, \Gamma(x,i))) \right)^2.$$

As $\mathcal{D}_R$ is uniform, for each $x \in S$ we have $\sum_{i \in [d]} W((x, \Gamma(x,i))) = \frac{1}{N}$. Thus,

$$\chi^\dagger M^\dagger M \chi \geq \frac{n|S|^2}{|\Gamma(S)|N} = \frac{\rho^2 nN}{|\Gamma(S)|}. \tag{11}$$

Combining Equations (10) and (11), we get

$$|\Gamma(S)| \geq \frac{\rho^2 nN}{N\rho^2 + \lambda\rho N(1-\rho)} = \frac{\rho n}{\rho + \lambda(1-\rho)} = \frac{1}{C \cdot (\rho + \lambda(1-\rho))} \cdot |S|.$$

■

B Strong Dispersers and Zero-Error List Recovery

For simplicity, we use a slightly different definition of list recovery, in which we bound the expected input lists size.

Definition B.1. *We say that a code $\mathcal{C} \subseteq [M]^D$ is (ℓ, L) list-recoverable if if for every $S_1, \ldots, S_D \subseteq [M]$ such that $\frac{1}{D}\sum_{i \in [D]} |S_i| \leq \ell$, there are at most L codewords $c \in \mathcal{C}$ such that $c \in S_1 \times \ldots \times S_n$.*

Definition B.2 (strong disperser)**.** *A function $\mathsf{Disp}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a strong (k,s) disperser if for every (n,k) source $\boldsymbol{X}$[17] and an independent uniform $\boldsymbol{Y} \sim \{0,1\}^d$ it holds that*

$$|\mathrm{Supp}\left(\boldsymbol{Y} \circ \mathsf{Disp}(\boldsymbol{X}, \boldsymbol{Y})\right)| > 2^{s+d}.$$

This object is sometimes referred to as a strong (k, ε) disperser for $\varepsilon = 1 - 2^{s-m}$.

Denote $D = 2^d$ and $M = 2^m$. Given a function $f\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, we denote by $\mathcal{C}_f\colon \{0,1\}^n \to [M]^D$ the encoding

$$\mathcal{C}_f(x) = (f(1), \ldots, f(D)),$$

where we identify $\{0,1\}^d$ with $[D]$.

Claim B.3. *Let $\mathsf{Disp}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be some function such that $\mathcal{C}_{\mathsf{Disp}} \subseteq [M]^D$ is (ℓ, L) list-recoverable. Then, Disp is a strong $(k = \log L, s = \log \ell)$ disperser.*

[17] We model an (n,k) source $\boldsymbol{X}$ as a random variable distributed over $\{0,1\}^n$ such that for any element x in its support, $\Pr[\boldsymbol{X} = x] \leq 2^{-k}$. In particular, this implies that the support of $\boldsymbol{X}$ is of size larger than 2^k.

Proof: Assume towards a contradiction that Disp is not such a disperser, so there exists an (n, k) source $\boldsymbol{X}$ for which the support of $\boldsymbol{Y} \circ \mathsf{Disp}(\boldsymbol{X}, \boldsymbol{Y})$ is small. In particular,

$$\sum_{i \in [D]} |\mathrm{Supp}\left(\mathsf{Disp}(\boldsymbol{X}, i)\right)| \leq D \cdot 2^s = D \cdot \ell.$$

Define $\mathcal{L}_i = \{\mathsf{Disp}(x, i) : x \in \mathrm{Supp}(\boldsymbol{X})\}$, so $\sum_{i \in [D]} |\mathcal{L}_i| \leq D \cdot \ell$. By definition, for every $i \in [D]$ and $x \in \mathrm{Supp}(\boldsymbol{X})$ we have that $\mathcal{C}_{\mathsf{Disp}}(x)_i \in \mathcal{L}_i$. By the list-recovery property it means that $|\mathrm{Supp}(\boldsymbol{X})| < L = 2^k$, in contradiction. ■

Claim B.4. *Let* $\mathsf{Disp} \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *be a strong* (k, s) *disperser. Then,* $\mathcal{C}_{\mathsf{Disp}} \colon \{0,1\}^n \to [M]^D$ *is* $(\ell = 2^s, L = 2^k)$ *list-recoverable.*

Proof: Let $\mathcal{L}_1, \ldots, \mathcal{L}_D \subseteq [M]$ be such that $\sum_{i \in [D]} |\mathcal{L}_i| \leq D \cdot \ell$. Let $T \subseteq [M] \times [D]$ be such that $(z, i) \in T$ if and only if $z \in \mathcal{L}_i$. Let

$$\mathcal{L} = \left\{u \in \mathcal{C}_{\mathsf{Disp}} : \forall i \in [D], u_i \in \mathcal{L}_i\right\},$$

and assume towards a contradiction that $|\mathcal{L}| \geq L$. The set $\mathcal{L}$ is in one-to-one correspondence with the set

$$\mathcal{A} = \left\{x \in \{0,1\}^n : \forall i \in [D], \mathsf{Disp}(x, i) \in \mathcal{L}_i\right\}.$$

Let $\boldsymbol{Y}$ be uniform over $[D]$, and let $\boldsymbol{A}$ be uniform over $\mathcal{A}$ independently of $\boldsymbol{Y}$. Thus,

$$\mathrm{Supp}\left(\mathsf{Disp}(\boldsymbol{A}, \boldsymbol{Y}) \circ \boldsymbol{Y}\right) \subseteq T,$$

and $T \leq D \cdot \ell$. This contradicts the disperser property,

$$|\mathrm{Supp}\left(\mathsf{Disp}(\boldsymbol{A}, \boldsymbol{Y}) \circ \boldsymbol{Y}\right)| > D \cdot 2^s = D \cdot \ell,$$

observing that $\boldsymbol{A}$ is an $(n, \log L)$ source. ■