For-All Sparse Recovery in Near-Optimal Time

ANNA C. GILBERT, Department of Mathematics. University of Michigan, Ann Arbor
YI LI, Division of Mathematics, SPMS. Nanyang Technological University
ELY PORAT, Department of Computer Science. Bar-Ilan University
MARTIN J. STRAUSS, Department of Mathematics. University of Michigan, Ann Arbor

An *approximate sparse recovery system* in ℓ_1 norm consists of parameters k, ϵ, N; an m-by-N measurement Φ; and a recovery algorithm $\mathcal{R}$. Given a vector, $\mathbf{x}$, the system approximates x by $\widehat{\mathbf{x}} = \mathcal{R}(\Phi\mathbf{x})$, which must satisfy $\|\widehat{\mathbf{x}}-\mathbf{x}\|_1 \leq (1+\epsilon)\|\mathbf{x}-\mathbf{x}_k\|_1$. We consider the "for all" model, in which a single matrix Φ, possibly "constructed" non-explicitly using the probabilistic method, is used for all signals $\mathbf{x}$. The best existing sublinear algorithm by Porat and Strauss [2012] uses $O(\epsilon^{-3}k\log(N/k))$ measurements and runs in time $O(k^{1-\alpha}N^{\alpha})$ for any constant $\alpha > 0$.

In this article, we improve the number of measurements to $O(\epsilon^{-2}k\log(N/k))$, matching the best existing upper bound (attained by super-linear algorithms), and the runtime to $O(k^{1+\beta}\,\mathrm{poly}(\log N, 1/\epsilon))$, with a modest restriction that $k \leq N^{1-\alpha}$ and $\epsilon \leq (\log k/\log N)^{\gamma}$ for any constants $\alpha, \beta, \gamma > 0$. When $k \leq \log^{c} N$ for some $c > 0$, the runtime is reduced to $O(k\,\mathrm{poly}(N, 1/\epsilon))$. With no restrictions on ϵ, we have an approximation recovery system with $m = O(k/\epsilon\log(N/k)((\log N/\log k)^{\gamma} + 1/\epsilon))$ measurements.

The overall architecture of this algorithm is similar to that of Porat and Strauss [2012] in that we repeatedly use a weak recovery system (with varying parameters) to obtain a top-level recovery algorithm. The weak recovery system consists of a two-layer hashing procedure (or with two unbalanced expanders for a deterministic algorithm). The algorithmic innovation is a novel encoding procedure that is reminiscent of network coding and that reflects the structure of the hashing stages. The idea is to encode the signal position index i by associating it with a unique message $\mathbf{m}_i$, which will be encoded to a longer message $\mathbf{m}'_i$ (in contrast to Porat and Strauss [2012] in which the encoding is simply the identity). Portions of the message $\mathbf{m}'_i$ correspond to repetitions of the hashing, and we use a regular expander graph to encode the linkages among these portions.

The decoding or recovery algorithm consists of recovering the portions of the longer messages $\mathbf{m}'_i$ and then decoding to the original messages $\mathbf{m}_i$, all the while ensuring that corruptions can be detected and/or corrected. The recovery algorithm is similar to list recovery introduced in Indyk et al. [2010] and used in Gilbert et al. [2013]. In our algorithm, the messages $\{\mathbf{m}_i\}$ are independent of the hashing, which enables us to obtain a better result.

CCS Concepts: • **Theory of computation → Sketching and sampling;**

Additional Key Words and Phrases: Compressive sensing, list decoding, sparse recovery

---

A. C. Gilbert was supported in part by DARPA/ONR N66001-08-1-2065. Y. Li was supported by NSF CCF 0743372 when he was at University of Michigan. M. J. Strauss was supported in part by NSF CCF 0743372 and DARPA/ONR N66001-08-1-2065.
A preliminary version appeared in the Proceedings of ICALP 2014, *LNCS* 8572, pp 538–550.
Authors' addresses: A. C. Gilbert and M. J. Strauss, 530 Church St, Ann Arbor, MI 48109, USA; emails: {annacg, martinjs}@umich.edu; Y. Li, 21 Nanyang Link, Singapore 637371; email: yili@ntu.edu.sg; E. Porat, Ramat-Gan, 5290002 Israel; email: porately@cs.biu.ac.il.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2017 ACM 1549-6325/2017/03-ART32 $15.00
DOI: http://dx.doi.org/10.1145/3039872

ACM Reference Format:
Anna C. Gilbert, Yi Li, Ely Porat, and Martin J. Strauss. 2017. For-all sparse recovery in near-optimal time. ACM Trans. Algorithms 13, 3, Article 32 (March 2017), 26 pages.
DOI: http://dx.doi.org/10.1145/3039872

1. INTRODUCTION

Sparse signal recovery is a critical data-acquisition and processing problem that arises in many modern scientific and computational applications, including signal and image processing, machine learning, data networking, and medicine [Duarte et al. 2008; Lustig et al. 2007]. It is a method for acquiring linear measurements or observations of a signal with a measurement matrix Φ, and an algorithm, $\mathcal{D}$, for recovering the significant components of the original signal. We model this problem mathematically by assuming that we *measure* a vector $\mathbf{x}$ and collect observation $\mathbf{y} = \Phi\mathbf{x}$, and then we run a *recovery algorithm* and produce an approximation $\widehat{\mathbf{x}} = \mathcal{D}(\Phi, \mathbf{y})$ to $\mathbf{x}$ with the guarantee that the approximation error $\|\widehat{\mathbf{x}} - \mathbf{x}\|$ is bounded above.

More quantitatively, let us denote the length of the vector $\mathbf{x}$ by N, the sparsity (or compression) parameter k, and distortion parameter ϵ. Let $\mathbf{x}_{[k]}$ denote the best k-term approximation to $\mathbf{x}$, the "heavy hitters" of $\mathbf{x}$, that is, $\mathbf{x}$ with all but the k largest-magnitude terms zeroed out. There are many different ways to assess the error of the recovery algorithm and the quality of the measurement matrix, depending on the particular application. (See Table I for an overview of all of problem variations.) In this article, we address the ℓ_1/ℓ_1-for-all problem,[1] which is to give a measurement matrix Φ and a recovery algorithm $\mathcal{D}$, such that, for any input vector $\mathbf{x}$, we have

$$\|\widehat{\mathbf{x}} - \mathbf{x}\|_1 \leq (1 + \epsilon)\|\mathbf{x}_{[k]} - \mathbf{x}\|_1.$$

The goal is to use the minimum number of measurements (rows of Φ), namely, $O(k \log(N/k)/\epsilon^2)$, and to keep the runtime of $\mathcal{D}$ to polynomial in $k \log(N)/\epsilon$. Since the measurement matrix Φ is chosen independently of the input vector $\mathbf{x}$, it corresponds to *non-adaptive* measurements. (We do not know whether adaptivity would help in this setting.)

What makes this problem challenging is that we must simultaneously keep the number of measurements small, ensure that the recovery algorithm is highly efficient, and achieve a good approximation for all input vectors. If we increase the number of measurements by factors of $\log N$, then it is easy to optimize the runtime [Berinde et al. 2008; Cheraghchi and Indyk 2016]. Similarly, if $\epsilon < 1/N$, the desired bound allows at least $1/\epsilon > N$ measurements, and the problem becomes trivial. In many applications, all three quantities are important; that is, in medical imaging applications, the measurements reflect the time a patient is observed, the recovery time drives the effectiveness of real-time imaging systems, and the recovery accuracy determines the diagnostic effectiveness of the imaging system.

Related work. There has been considerable work on this problem in a variety of parameter settings, and we summarize the results in Table I. A number of parameter values are incommensurate: We can achieve better approximation guarantees (using the ℓ_2/ℓ_2 norm) but only in the for-each model, and in the for-all signal model, we can achieve ℓ_2/ℓ_1 error guarantees. A somewhat harder problem than the one we address in this article is the mixed-norm (or ℓ_2/ℓ_1) for-all result. In this setting, the goal is to

[1]More generally, the expression ℓ_p/ℓ_q means that we measure the approximation error $\|\widehat{\mathbf{x}} - \mathbf{x}\|_p$ with the ℓ_p norm and we compare it to the ℓ_q error of the best k-term approximation, $\|\mathbf{x}_{[k]} - \mathbf{x}\|_q$.

Table I. Summary of the Best Previous Results and the Result Obtained in This Paper

Paper	A/E	Number of Measurements	Column sparsity/ Update time	Decode time	Approx. error	Noise
[Charikar et al. 2002]	E	$k\log^{O(1)}N$	$\log^{O(1)}N$	$N\log^{O(1)}N$	$\ell_2 \le C\ell_2$	
[Cormode and Muthukrishnan 2006]	E	$k\log^{O(1)}N$	$\log^{O(1)}N$	$k\log^{O(1)}N$	$\ell_2 \le C\ell_2$	
[Gilbert et al. 2012]	E	$\epsilon^{-1}k\log(N/k)$	$\log^{O(1)}N$	$\epsilon^{-1}k\log^{O(1)}N$	$\ell_2 \le (1+\epsilon)\ell_2$	Y
[Donoho 2006; Candès et al. 2006]	A	$k\log(N/k)$	$k\log(N/k)$	LP	$\ell_2 \le (C/\sqrt{k})\ell_1$	Y
[Gilbert et al. 2007]	A	$\epsilon^{-2}k\log^{O(1)}N$	$\epsilon^{-2}k\log^{O(1)}N$	$\epsilon^{-4}k^2\log^{O(1)}N$	$\ell_2 \le (\epsilon/\sqrt{k})\ell_1$	Y
[Gilbert et al. 2006]	A	$k\log^{O(1)}N$	$\log^{O(1)}N$	$k\log^{O(1)}N$	$\ell_1 \le (C\log N)\ell_1$	Y
[Indyk and Ruzic 2008]	A	$\epsilon^{-2}k\log(N/k)$	$\epsilon^{-1}\log(N/k)$	$N\log(N/k)$	$\ell_1 \le (1+\epsilon)\ell_1$	Y
[Porat and Strauss 2012]	A	$\ell^{O(1)}\epsilon^{-3}k\log(N/k)$	$\ell^{O(1)}\epsilon^{-3}\log(N/k)\log k$	$\ell^{O(1)}\epsilon^{-3}k(N/k)^{1/\ell}$	$\ell_1 \le (1+\epsilon)\ell_1$	Y
This article	A	$\epsilon^{-2}k\log N$	$\epsilon^{-1}\log N$	$k^{1+\beta}(\epsilon^{-1}\log N)^{O(1)}$	$\ell_1 \le (1+\epsilon)\ell_1$	
Lower bound 'A'	A	$\epsilon^{-2}k\log(N/k)$	$\epsilon^{-1}\log(N/k)$	$\epsilon^{-2}k\log(N/k)$	$\ell_2 \le (\epsilon/\sqrt{k})\ell_1$	Y

Summary of the best previous results and the result obtained in this article. The measurement and time complexities are subject to O-notations, which are suppressed for clarity. In Porat and Strauss [2012], ℓ is an arbitrary positive constant integer and the $O(1)$ in exponents are absolute constants; in the result of this article, β is an arbitrary positive constant, the $O(1)$ in the exponent in decode time takes the form of $c_1 + c_2\beta$ (where c_1, c_2 are absolute constants), and restrictions on k and ϵ apply. "LP" denotes (at least) the time to solve a linear program of size at least N. The column "A/E" indicates whether the algorithm works in the for-all (A) model or the for-each (E) model. The column "noise" indicates whether the algorithm tolerates noisy measurements, that is, the observation $\mathbf{y} = \Phi\mathbf{x} + \boldsymbol{\nu}$. Measurement and decode time dependence on ϵ, where applicable, is polynomial. The lower bound on number of measurements in table above is, in fact, the best upper bound attained by super-linear algorithms.

give Φ and $\mathcal{D}$, such that, for any $\mathbf{x}$, we have

$$\|\widehat{\mathbf{x}} - \mathbf{x}\|_2 \le \frac{\epsilon}{\sqrt{k}}\|\mathbf{x}_{[k]} - \mathbf{x}\|_1. \tag{1}$$

It is known that if $(\Phi, \mathcal{D})$ solves the ℓ_2/ℓ_1 problem, then it also solves the ℓ_1/ℓ_1 problem [Cohen et al. 2009].

In another direction, the ℓ_2/ℓ_2 for-each problem is to give *distribution* $\mathcal{F}$ on Φ, and $\mathcal{D}$, such that, for any $\mathbf{x}$, if $\Phi \sim \mathcal{F}$, we have

$$\Pr_{\Phi\sim\mathcal{F}}\left\{\|\widehat{\mathbf{x}} - \mathbf{x}\|_2 \le (1+\epsilon)\|\mathbf{x}_{[k]} - \mathbf{x}\|_2\right\} \ge 1 - O(1).$$

The ℓ_2/ℓ_2 for-each problem with constant failure probability was solved in Gilbert et al. [2012], where the authors gave an algorithm with constant-factor-optimal runtime and number of measurements. The failure probability was recently improved to exponentially small in Gilbert et al. [2013], but the technique is not likely to give an ℓ_1/ℓ_1 for-all result without additional logarithmic factors in the number of measurements.

The first sublinear-time algorithm in the for-all setting (for the ℓ_1/ℓ_1 norm) was given in Porat and Strauss [2012], although that algorithm had a number of limitations.

—The runtime, while sublinear, was $\sqrt{kN}$ or, more generally, of the form $k^{1-\alpha}N^{\alpha}$ for any constant $\alpha > 0$. That algorithm does not achieve polynomial running time in $k\log(N)/\epsilon$.
—The algorithm requires a precomputed table of size $Nk^{0.2}$.
—The dependence on ϵ is $1/\epsilon^3$, far from optimal dependence of $1/\epsilon^2$.

Our results. In this work, we rectify the above limitations, assuming the (modest) restriction that $\epsilon < \log k/\log N$. We also make the measurement dependence on ϵ optimal. The best lower bound for the ℓ_1/ℓ_1 for-all problem is $\Omega(k/\epsilon^2 + (k/\epsilon)\log(\epsilon N/k))$ [Nelson et al. 2014], which is also the best lower bound for the ℓ_2/ℓ_1 for-all problem. Our algorithm uses $O(k/\epsilon^2\log(N/k))$ measurements when $\epsilon < (\log k/\log N)^{\gamma}$, which is suboptimal only by a logarithmic factor. When $k \le \log^c N$ for some $c > 0$, the runtime is reduced to $O(k\,\mathrm{poly}(N, 1/\epsilon))$.

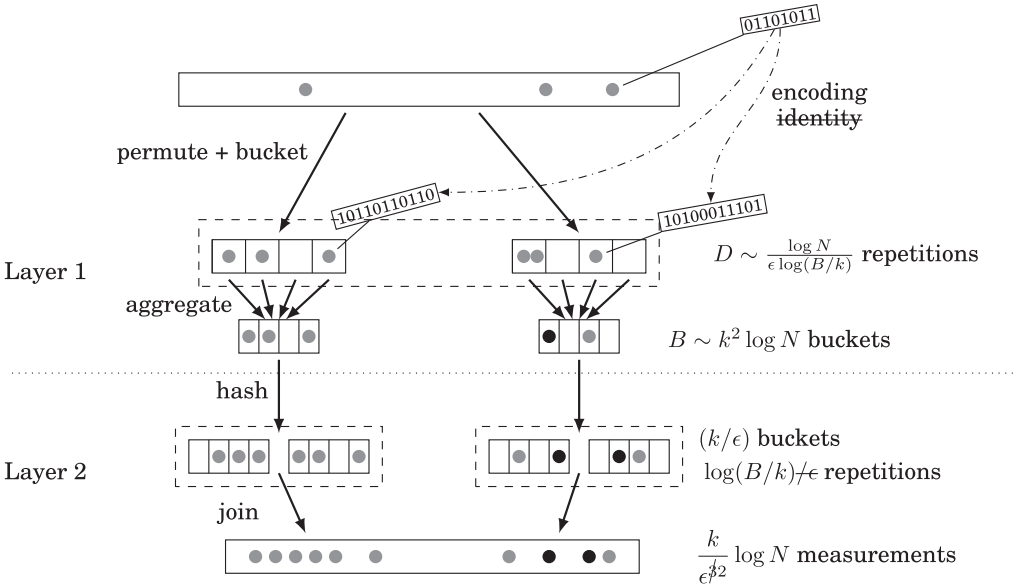

Fig. 1. Algorithm to generate the measurements. Darker spots indicate a bigger value of the bucket/measurement. Strikethroughs are used to show where our approach or our object sizes differ from those of Porat and Strauss [2012].

THEOREM 1.1 (MAIN THEOREM). *Let $\beta, \gamma > 0$. There is an approximate sparse recovery system consisting of an $m \times N$ measurement matrix $\mathbf{\Phi}$ and a decoding algorithm $\mathcal{D}$ that satisfy the following property: For any vector $\mathbf{x} \in \mathbb{R}^n$, given $\mathbf{\Phi x}$, the system approximates $\mathbf{x}$ by $\widehat{\mathbf{x}} = \mathcal{D}(\mathbf{\Phi x})$, which satisfies*

$$\|\widehat{\mathbf{x}} - \mathbf{x}\|_1 \leq (1+\epsilon)\|\mathbf{x}_{[k]} - \mathbf{x}\|_1.$$

Provided that $N = \Omega(\max\{k^2, k/\epsilon^2\})$, the matrix $\mathbf{\Phi}$ has $m = O(k/\epsilon \log(N)((\log N/\log k)^\gamma + 1/\epsilon))$ rows, and the decoding algorithm $\mathcal{D}$ runs in time $O(k^{1+\beta}\,\mathrm{poly}(\log N, 1/\epsilon))$. When $\epsilon = O((\frac{\log k}{\log N})^\gamma)$, the number of rows is $m = O(k/\epsilon^2 \log N)$. If, in addition, $k \leq \log^{O(1)} N$, then the runtime can be reduced to $O(k\,\mathrm{poly}(\log N, 1/\epsilon))$.

Overview of techniques. Our overall approach builds on Porat and Strauss [2012] and Gilbert et al. [2013], with several critical innovations. In Figure 1 is a framework that captures both the algorithm in Porat and Strauss [2012] and the algorithm in this article.

First, we describe the encoding procedure at a high level. Initially, each $i \in [N]$ is associated with a unique message $\mathbf{m}_i$, which is encoded to a longer message $\mathbf{m}'_i$. In Porat and Strauss [2012], this encoding is trivial, namely, $\mathbf{m}'_i = \mathbf{m}_i$, whereas in our work, it is a more complicated procedure (see Figure 3). The first hash assigns one of B buckets to each $i \in [N]$, while maintaining the original index i; the *aggregation* step sums each bucket. There are $\frac{\log N}{\epsilon \log(B/k)}$ repetitions. The index i in each repetition is now associated with a block of $\mathbf{m}'_i$. In Porat and Strauss [2012], the aggregated buckets are hashed into (k/ϵ) buckets and there are $\log(B/k)/\epsilon$ repetitions. Thus, altogether, there are $O(\epsilon^{-3}k \log N)$ measurements (recall that $\log N = \Theta(\log(N/k))$ when $k = O(\sqrt{N})$). In our work, there are only $\log(B/k)$ repetitions, saving a factor of $1/\epsilon$, so the total number of measurements is $O(\epsilon^{-2}k \log N)$.

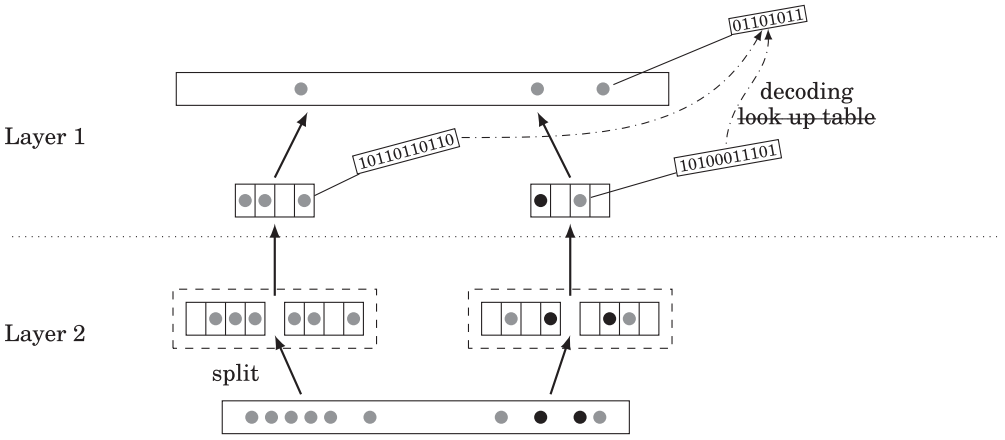

Fig. 2. Algorithm to recover from the measurements.

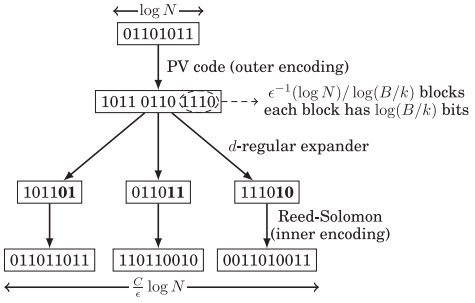

Fig. 3. Encoding scheme. The Parvaresh-Vardy code automatically has a block structure. Suppose that there are D blocks. Choose a d-regular expander on D vertices as desired. For the ith block of the PV code, append to it the information of the neighbours of the ith vertex in the expander. Then apply Reed-Solomon to each appended message block. Note that the codes are non-binary.

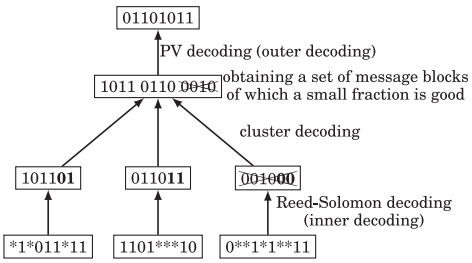

Fig. 4. Decoding scheme. The asterisks in the bottom layer indicate corrupted measurements (due to collision or noise). The Reed-Solomon decoding either recovers the message block (with linking information) or produces a wrong one (crossed out) that is useless in recovering the original message. Then the clustering procedure finds a set of message blocks, of which a small fraction is good. This is sufficient for the Parvaresh-Vardy decoding to succeed.

The *identification* portion of the recovery algorithm is shown in Figure 2. To recover the identity of heavy hitters, the algorithm reads off the measurements and recovers the message block associated with each bucket. This message block is supposed to be associated with the heavy hitter in the bucket. Then all B buckets are examined exhaustively. The pre-image of each heavy bucket under the first hash is determined, in Porat and Strauss [2012], from a lookup table and searched exhaustively. In our work, this is done by the decoding procedure illustrated in Figure 4. We encode the "linking information" into the message blocks so we can collect across the repetitions enough heavy buckets that contain the same heavy hitter i (whose actual value is unknown at this stage of the algorithm). Thus, we obtain a (small) fraction of $\mathbf{m}'_i$, which is sufficient for the Parvaresh-Vardy decoding algorithm to produce the exact $\mathbf{m}_i$, from which we recover the value of i immediately.

The *estimation* portion of the recovery algorithm estimates the coefficient at each of those candidate positions by reading the aggregated bucket value of the corresponding heavy buckets at the first hash level.

Putting these pieces together, we have a *weak recovery system*, which identifies all but $k/2$ of the heavy hitters. We then repeat with a smaller (easier) sparsity parameter $k/2 < k$ and a smaller (harder) distortion parameter $(3/4)\epsilon < \epsilon$, resulting in a number of measurements whose leading term is $(k/2)(4/3\epsilon)^2 = (8/9)k/\epsilon^2 < k/\epsilon^2$. Summing the geometric progression gives the result we need. Finally, we note that our algorithm works (deterministically) with any unbalanced expander having the appropriate properties.

Encoding and decoding details. See Figure 3 and Figure 4 for a detailed illustration of these steps. For each message $\mathbf{m}$, the Parvaresh-Vardy code[2] encodes it into a longer message $\mathbf{m}'$, which automatically exhibits a block structure, so if a few numbers of the blocks are correct, the original $\mathbf{m}$ will be recovered. Suppose there are D blocks. Now, choose a d-regular expander graph G (d is a constant) on D nodes such that after removing $O(D)$ nodes from G, the remaining graph still contains an expander of size $\Omega(D)$. For the ith block of $\mathbf{m}'$, append to it the information of the neighbours of the ith vertex in G. Then we apply Reed-Solomon to protect the appended blocks.

To decode, we first recover the appended message blocks. The two-layer hash guarantees that, for the same heavy hitter, at most $O(D)$ of them will be wrong and the remaining ones are all correct. Now, consider a breadth-first search from a correct message block (whose "linking information" is therefore correct). By the special property of the expander graph G, we shall be able to visit all nodes (i.e., all corresponding message blocks) of a smaller expander graph of size $\Omega(D)$ in $\log D$ steps. This small fraction of good message blocks of $\mathbf{m}'$ will enable the P-V code to recover the original message $\mathbf{m}$ successfully. Recall that d is a constant, and the total number of vertices visited is $O(d^{\log D}) = O(\text{poly}(D)) = O(\text{poly}(\log N))$ for appropriate D. This enables a sublinear recovery time.

Our contributions.

—We give an algorithm for sparse recovery in the for-all setting, under a modest restriction on the distortion factor ϵ, having the number of measurements that matches the best upper bound, attained by super-linear algorithms, for example, Indyk and Ruzic [2008], and optimal in runtime up to a power.
—Our work is not the first to consider list recovery. Indyk et al. introduces the idea in the context of combinatorial group testing [Indyk et al. 2010]. List recovery is also used in Cheraghchi [2013]. The list recovery used in Gilbert et al. [2013], however, would affect the hashing, and the hashing was thus required to be sufficiently random. In our algorithm, the messages $\{\mathbf{m}_i\}$ are independent of the hashing, which enables us to obtain a better result.
—Finally, our encoding/decoding techniques are reminiscent of network coding and may have other contexts for soft-decoding or network coding.

Article organization. In Section 2, we review some properties of expanders. In Section 3, we show that, provided with good identification results, unbalanced expanders with appropriate properties will give a weak system. Our construction of a weak system culminates in Section 4, where we shall show how to achieve good identification via message encoding and decoding. Then we build the overall algorithm on the weak

[2]There is no particular reason why we have chosen the Parvaresh-Vardy code; it can be replaced with other codes with similar or better performance, for example, folded Reed-Solomon code.

system in Section 5. Finally, we close with a short discussion and open problems in Section 6.

2. PRELIMINARIES

Our main algorithm will be built on regular graph expanders and unbalanced bipartite expanders (or, rather, the adjacency matrices of such graphs). In an abuse of terminology, we will also use two different types of hashing schemes that can be implemented as (random) unbalanced bipartite expanders. In some contexts, it is more natural to describe and to analyze the structures as hashing schemes and, in others, it is more natural to use the properties of expanders. In this section, we review some properties of expanders and define precisely our hashing schemes. We also show that, up to an appropriate interpretation of the parameters, the two combinatorial structures are equivalent.

2.1. Expander Graphs

Let n, m, d, ℓ be positive integers and ϵ, κ be positive reals. The following two definitions are adapted from Guruswami et al. [2009].

Definition 2.1 (*Expander*). An (n, ℓ, κ)-expander is a graph $G(V, E)$, where $|V| = n$, such that for any set $S \subseteq V$ with $|S| \leq \ell$ it holds that $|\Gamma(S)| \geq \kappa|S|$.

Definition 2.2 (*Bipartite Expander*). An $(n, m, d, \ell, \epsilon)$-bipartite expander is a d-left-regular bipartite graph $G(L \cup R, E)$, where $|L| = n$ and $|R| = m$ such that for any $S \subseteq L$ with $|S| \leq \ell$ it holds that $|\Gamma(S)| \geq (1 - \epsilon)d|S|$, where $\Gamma(S)$ is the neighbour of S (in R).

When n and m are clear from the context, we abbreviate the expander as an (ℓ, d, ϵ)-bipartite expander.

Consider the adjacency matrix A_G of an d-regular expander G. It always holds that the largest eigenvalue of A_G in absolute value is d. Let $\lambda(G)$ denote the largest absolute value of any other eigenvalue. The following theorem is classical.

THEOREM 2.3 (FRIEDMAN ET AL. [1989]). *There exists absolute constants $c > 1$ and $C > 0$ such that for all sufficiently large n and even d, there exists a d-regular $(n, n/2, c)$-expander G such that $\lambda(G) \leq C\sqrt{d}$.*

Next we present a result by Upfal [1992], implicitly used in the proof of Lemmas 1 and 2 therein. It states that there exists an expander graph of n nodes and constant degree such that after removing a constant fraction of nodes the remaining subgraph contains an expander of size $\Omega(n)$.

THEOREM 2.4 (UPFAL [1992]). *Let G be an $(n, n/2, c)$-expander such that G is δ-regular and $\lambda(G) \leq C\sqrt{\delta}$, where δ is a (sufficiently large) constant and $c > 1$, $C > 0$ are absolute constants. There exist constants $\alpha, \zeta > 0$, and $\kappa > 1$, depending on c and C, such that after removing an arbitrary set of at most ζn nodes from G, the remaining graph contains a subgraph G' such that $|V(G')| \geq \alpha n$ and G' is a $(|V(G')|, n/2, \kappa)$-expander.*

2.2. Hashing Schemes

We employ two types of hashing schemes in our algorithm. To aid in the exposition of the analysis, it is useful to describe these in terms of their action on particular elements of a vector (i.e., they "hash items into buckets"). The parameters N, B_1, B_2, d_1, d_2 of the hashing schemes are positive integers. We adopt the conventional notation that $[m] = \{1, 2, \ldots, m\}$.

Definition 2.5 (*One-layer Hashing Scheme*). The (N, B, d) (one-layer) hashing scheme is the uniform distribution on the set of all functions $f : [N] \to [B]^d$. We write $f(x) = (f_1(x), \ldots, f_d(x))$, where f_i's are independent (N, B) hashing schemes.

Each instance of such a hashing scheme induces a d-left-regular bipartite graph with Bd right nodes. When N is clear from the context, we simply write (B, d) hashing scheme.

Definition 2.6 (*Two-layer Hashing Scheme*). An (N, B_1, d_1, B_2, d_2) (two-layer) hashing scheme is a distribution μ on the set of all functions $f : [N] \to [B_2]^{d_1 d_2}$ defined as follows. Let g be a random function subject to the (N, B_1, d_1) hashing scheme and $\{h_{i,j}\}_{i\in[d_1], j\in[d_2]}$ be a family of independent functions subject to the (B_1, B_2, d_2) hashing scheme that are also independent of g. Then μ is defined to be the distribution induced by the mapping

$$x \mapsto \big(h_{1,1}(g_1(x)), \ldots, h_{1,d_2}(g_1(x)), h_{2,1}(g_2(x)), \ldots, h_{2,d_2}(g_2(x)), \ldots, h_{d_1,1}(g_{d_1}(x)), \ldots, h_{d_1,d_2}(g_{d_1}(x))\big).$$

Each instance of such a hashing scheme gives a d_1d_2-left-regular bipartite graph of $B_2d_1d_2$ right nodes. When N is clear from the context, we simply write a (B_1, d_1, B_2, d_2) hashing scheme. Conceptually, we hash N elements into B_1 buckets and repeat d_1 times, and those buckets will be referred to as first-layer buckets; in each of the d_1 repetitions, we hash B_1 elements into B_2 buckets and repeat d_2 times, and those buckets will be referred to as second-layer buckets.

We note that bipartite expander graphs can be used as hashing schemes because of their isolation property.

Definition 2.7 (*Isolation Property*). An $(n, m, d, \ell, \epsilon)$-bipartite expander G is said to satisfy the (ℓ, η, ζ)-isolation property if, for any set $S \subset L(G)$ with $|S| \le \ell$, there exists $S' \subset S$ with $|S'| \ge (1 - \eta)|S|$ such that for all $x \in S'$ it holds that $|\Gamma(\{x\}) \setminus \Gamma(S \setminus \{x\})| \ge (1 - \zeta)d$.

2.3. Bipartite Expanders and Hashing Schemes

All proofs use standard techniques and are shown in the Appendix.

2.3.1. One-Layer Hashing.

Lemma 2.8 (Expanding Property). *For any $\epsilon \in (0, 1/4)$, $k \ge 1$, $\alpha \ge 1$, and $N = \Omega(\alpha k)$, a random one-layer (B, d) hashing scheme gives an $(N, Bd, d, \alpha k, \epsilon)$-bipartite expander with probability $\ge 1 - 1/N^c$, where $B = \Omega(\frac{\alpha k}{\epsilon})$ and $d = \Omega(\frac{1}{\epsilon} \log \frac{N}{k})$.*

Lemma 2.9 (Isolation Property). *For any $\epsilon, \zeta \in (0, 1/4)$, $k \ge 1$, $\alpha \ge 1$ and $N = \Omega(k/\epsilon)$, a random one-layer (B, d) hashing scheme gives a bipartite graph with (L, ϵ, ζ)-isolation property with probability $\ge 1 - 1/N^c$, where $B = \Omega(\frac{k}{\zeta\epsilon})$, $d = \Omega(\frac{1}{\zeta\epsilon} \log \frac{N}{k})$, and $L = O(k/\epsilon)$.*

2.3.2. Two-Layer Hashing.

Now we show that a two-layer hashing scheme also gives a desirable bipartite expander.

Lemma 2.10 (Expanding Property). *Let $\epsilon \in (0, 1/4)$, $k \ge 1$, and $N = \Omega(\max\{k/\epsilon^2, k^2\})$. A random two-layer (B_1, d_1, B_2, d_2) hashing scheme gives an $(N, B_2d_1d_2, d_1d_2, 4k, \epsilon)$-bipartite expander with probability $\ge 1 - 1/N^c$, where $B_1 = \Omega(\frac{k}{\epsilon^2})$, $d_1 = \Omega(\frac{1}{\epsilon} \frac{\log N}{\log(B_1/k)})$, $B_2 = \Omega(\frac{k}{\epsilon})$, and $d_2 = \Omega(\log \frac{B_1}{k})$ with appropriate choices of constants.*

Remark 2.11. The constraint that $k = O(\sqrt{N})$ could be weakened to $k = O(N^{1-\xi})$ for any $\xi > 0$. The constants hidden in various $\Omega(\cdot)$ notations above will depend on ξ.

ALGORITHM 1: Weak Recovery System.

Input: $N, s, \mathbf{\Phi}$ (adjacency matrix of a d-left-regular expander G),$\mathbf{\Phi x}$, and I
Output: $\widehat{\mathbf{x}}$
  for each $i \in I$ **do**
    $\mathbf{x}'_i \leftarrow \text{median}_{u \in \Gamma(\{i\})} \sum_{(u,v)\in E} \mathbf{x}_u$ ▷ each sum is an element of input $\mathbf{\Phi x}$
  $\widehat{\mathbf{x}} \leftarrow$ top $O(s)$ elements of $\mathbf{x}'$
  return $\widehat{\mathbf{x}}$

We show that this two-layer hashing scheme also gives a good isolation property.

LEMMA 2.12 (ISOLATION PROPERTY). *Let $\epsilon > 0$, $\alpha > 1$ be arbitrary constants and (B_1, d_1, B_2, d_2) be a two-layer hashing scheme with $B_1 = \Omega(\frac{k}{\zeta^\alpha \epsilon^{2\alpha}})$, $d_1 = \Omega(\frac{\alpha}{\alpha-1} \cdot \frac{1}{\zeta\epsilon} \frac{\log N}{\log(B/k)})$, $B_2 = \Omega(\frac{k}{\zeta\epsilon})$, and $d_2 = \Omega(\frac{1}{\zeta} \log \frac{B_1}{k})$. Then, with probability $\geq 1 - 1/N^c$, the two-layer hashing scheme with parameters prescribed above gives a bipartite graph with the (L, ϵ, ζ)-isolation property, where $L = O(k/\epsilon)$.*

3. WEAK RECOVERY SYSTEM

To simplify our analysis, we decompose a signal $\mathbf{x}$ into two parts of disjoint support, $\mathbf{x} = \mathbf{y} + \mathbf{z}$, where $\mathbf{y}$ has small support and $\mathbf{z}$ has small norm. By normalization, we may assume that $\|\mathbf{z}\|_1 \leq 3/2$, where $3/2$ is chosen for the simplicity of constants in the proofs and can be replaced with an arbitrary positive number. We call $\mathbf{y}$ the *head* and $\mathbf{z}$ the *tail*. To simplify the language, we may also use a head to refer to $\text{supp}(\mathbf{y})$. We aim to recover the elements in $\mathbf{y}$. Introduced in Porat and Strauss [2012], a *weak system* takes an additional input, some set I of indices (called the candidate set), and tries to estimate $\mathbf{x}_i$ for $i \in I$, hoping to recover some head items with estimate error dependent on $\|\mathbf{z}\|_1$. It is shown in Porat and Strauss [2012] that when I contains the entire head, we can always recover a good fraction of the head. In this article, we make a slight modification on the definition of weak system as below. We only need I to contain a good fraction of the head instead of the entire head.

Definition 3.1 (*Weak Recovery System*). A *Weak recovery system* consists of parameters N, s, η, ζ; an m-by-N*measurement matrix* $\mathbf{\Phi}$; and a *decoding algorithm* $\mathcal{D}$ that satisfy the following property: For any $\mathbf{x} \in \mathbb{R}^N$ that can be written as $\mathbf{x} = \mathbf{y} + \mathbf{z}$, where $|\text{supp}(\mathbf{y})| \leq s$ and $\|\mathbf{z}\|_1 \leq 3/2$, given the measurements $\mathbf{\Phi x}$ and a subset $I \subseteq [N]$ such that $|I \cap \text{supp}(\mathbf{y})| \geq (1 - \zeta/2)|\text{supp}(\mathbf{y})|$, the decoding algorithm $\mathcal{D}$ returns $\widehat{\mathbf{x}}$, such that $\mathbf{x}$ admits the following decomposition:

$$\mathbf{x} = \widehat{\mathbf{x}} + \widehat{\mathbf{y}} + \widehat{\mathbf{z}},$$

where $|\text{supp}(\widehat{\mathbf{x}})| = O(s)$, $|\text{supp}(\widehat{\mathbf{y}})| \leq \zeta s$, and $\|\widehat{\mathbf{z}}\|_1 \leq \|\mathbf{z}\|_1 + \eta$. Intuitively, $\widehat{\mathbf{y}}$ and $\widehat{\mathbf{z}}$ will be the head and the tail of the residual $\mathbf{x} - \widehat{\mathbf{x}}$, respectively.

THEOREM 3.2 (WEAK RECOVERY). *Suppose that Φ is the adjacency matrix of an $(N, Bd, d, 4s, \eta)$-bipartite expander such that (a) $d = O(\frac{1}{\eta\zeta^2} \log \frac{N}{s})$ and $B = O(\frac{d}{\zeta\eta})$ and (b) it satisfies $(O(k/\eta), \eta, \zeta)$-isolation property. With appropriate instantiations of constants, Algorithm 1 yields a correct Weak recovery system that runs in time $O(|I|\eta^{-1}\zeta^{-2} \log(N/s))$.*

The proof is essentially the same as Porat and Strauss [2012, Lemma 4] but we hereby give a clearer abstraction by separating the deterministic argument from the randomized guarantees.

First, we need the following two lemmata.

LEMMA 3.3 (NOISE). *Let $\alpha > 1$ and $t > \alpha s$. Let Φ be the adjacency matrix of an $(n, m, d, 2\alpha s, \epsilon)$-bipartite expander with $\epsilon < 1/2$. Let $\mathbf{x} \in \mathbb{R}^n$ be such that $|\mathbf{x}_1| \geq |\mathbf{x}_2| \geq \cdots \geq |\mathbf{x}_n|$. Let $I = \{1, \ldots, \alpha s\}$, then*

$$\|(\Phi(\mathbf{x} - \mathbf{x}_{[t]}))_{\Gamma(I)}\|_1 \leq 4\epsilon d(\|\mathbf{x} - \mathbf{x}_{[t]}\|_1 + \alpha s|\mathbf{x}_{t+1}|).$$

PROOF. Partition $\{1, \ldots, N\}$ into blocks $I \cup H_1 \cup B_1 \cup B_2 \cup \ldots$, where $H_1 = \{\alpha s + 1, \ldots, t\}$ and $B_i = \{t + (i-1)\alpha s + 1, \ldots, t + i\alpha s\}$ for $i \geq 1$. Consider $\mathbf{x}$ restricted to a block B_i. We consider the following two cases.

Case 1. $\mathbf{x}_{B_i}$ is quasi-flat, that is, $|\mathbf{x}_{t+i\alpha s}| \geq |\mathbf{x}_{t+(i-1)\alpha s+1}|/2$. (It is called quasi-flat because all entries are within a factor of 2 of each other.) Consider all $d|B_i|$ edges in the expander emanating from B_i. Suppose that Z edges of them are incident to $\Gamma(I)$, then

$$|\Gamma(I) \cup \Gamma(B_i)| \leq d(|I| + |B_i|) - Z.$$

On the other hand, by the expansion property,

$$|\Gamma(I) \cup \Gamma(B_i)| \geq (1 - \epsilon)d(|I| + |B_i|),$$

which implies that

$$Z \leq \epsilon d(|I| + |B_i|) \leq 2\epsilon\alpha kd.$$

It follows that

$$\|(\Phi\mathbf{x}_{B_i})_{\Gamma(I)}\|_1 \leq Z \cdot \max_{i \in B_i} |\mathbf{x}_i| \leq 2\epsilon\alpha kd \cdot |\mathbf{x}_{t+(i-1)\alpha s+1}| \leq 4\epsilon d\|\mathbf{x}_{B_i}\|_1,$$

where the last inequality follows from the fact that $\mathbf{x}_{B_i}$ is quasi-flat so $\alpha s|\mathbf{x}_{t+(i-1)\alpha s+1}| \leq 2\|\mathbf{x}_{B_i}\|_1$.

Case 2. $\mathbf{x}_{B_i}$ is not quasi-flat, and then $|\mathbf{x}_{t+i\alpha s}| < |\mathbf{x}_{t+(i-1)\alpha s+1}|/2$. Let

$$J = \{i \in B_i : |\mathbf{x}_i| < |\mathbf{x}_{t+(i-1)\alpha s+1}|/2\}.$$

Increase $|\mathbf{x}_i|$ for all $i \in J$ so $|\mathbf{x}_i| = |\mathbf{x}_{t+(i-1)\alpha s+1}|/2$ and $\mathbf{x}_{B_i}$ becomes flat, and this increases $\|\mathbf{x}_{B_i}\|_1$ by at most $\alpha s|\mathbf{x}_{t+(i-1)\alpha s+1}|/2$. Invoking Case 1, we obtain

$$\|(\Phi\mathbf{x}_{B_i})_{\Gamma(I)}\|_1 \leq 4\epsilon d\left(\|\mathbf{x}_{B_i}\|_1 + \frac{\alpha s|\mathbf{x}_{t+(i-1)\alpha s+1}|}{2}\right).$$

Now we go back to the entire $\mathbf{x}$. Suppose that $\mathbf{x}_{B_{i_1}}, \ldots, \mathbf{x}_{B_{i_q}}$ are not quasi-flat, and then by triangle inequality we shall have

$$\|(\Phi(\mathbf{x} - \mathbf{x}_t))_{\Gamma(I)}\|_1 \leq 4\epsilon d\|\mathbf{x} - \mathbf{x}_t\|_1 + 4\epsilon d \cdot \frac{\alpha s}{2} \sum_{p=1}^{q} \left|\mathbf{x}_{t+(i_p-1)\alpha s+1}\right|.$$

Observe that, for $p \geq 2$,

$$|\mathbf{x}_{t+(i_p-1)\alpha s+1}| \leq |\mathbf{x}_{t+i_{p-1}\alpha s+1}| \leq \frac{|\mathbf{x}_{t+(i_{p-1}-1)\alpha s+1}|}{2},$$

where the last inequality follows from the quasi-flatness of $\mathbf{x}_{B_{i_{p-1}}}$. Hence,

$$|\mathbf{x}_{t+(i_p-1)\alpha s+1}| \leq \frac{|\mathbf{x}_{t+1}|}{2^{p-1}}, \quad p \geq 1,$$

whence it follows that

$$\|(\Phi(\mathbf{x} - \mathbf{x}_{[t]}))_{\Gamma(I)}\|_1 \leq 4\epsilon d(\|\mathbf{x} - \mathbf{x}_{[t]}\|_1 + \alpha s|\mathbf{x}_{t+1}|). \quad \square$$

In the usual decomposition, the head contains the entries with large coordinate values, which will be referred to as *heavy hitters*. If a heavy hitter fails to be recovered, then it

must have been displaced by another entry, called a decoy, in the recovered signal. The next lemma bounds the number of decoys.

LEMMA 3.4 (DECOYS). *Let $\theta, \epsilon \in (0,1)$ and $\beta, \zeta > 0$ such that $0 < \zeta < \frac{1}{2} - \frac{80\beta}{\theta}$. Suppose that G is a $(4s, d, \beta\epsilon)$-bipartite expander that satisfies the $(\frac{9s}{\epsilon}, \frac{\epsilon\theta}{18}, \zeta)$-isolation property. Let $\mathbf{x} \in \mathbb{R}^n$ be a signal satisfying the assumption in the Weak system, and let $\mathbf{x}' \in \mathbb{R}^n$ be the estimates defined as*

$$\mathbf{x}'_i = \underset{u \in \Gamma(\{i\})}{\text{median}} \sum_{(u,v) \in E} \mathbf{x}_u, \quad i \in [N].$$

Define

$$D = \{i \in \text{supp}(\mathbf{y}) : |\mathbf{x}_i - \mathbf{x}'_i| \geq \epsilon/(4s)\},$$

and then $|D| < \theta s$.

PROOF. Suppose that $|D| \geq \theta s$. By definition, it holds that $|D| \leq s$. Also assume that $|\mathbf{x}_1| \geq |\mathbf{x}_2| \geq \cdots \geq |\mathbf{x}_n|$. Suppose that $|\mathbf{x}_i| \geq \epsilon/(2s)$ for all $i \in H := \text{supp}(\mathbf{y})$; otherwise we can place the violated i's into $\mathbf{z}$, causing $\|\mathbf{z}\|_1$ to increase by at most $s \cdot \epsilon/(2s) = \epsilon/2$, so we would have $\|\mathbf{z}\|_1 \leq 2$. Let $T = H \cup D \cup \{i : |\mathbf{x}_i| \geq \epsilon/(4s)\}$, and then $t := |T| \leq \|\mathbf{z}\|_1/(\epsilon/(4s)) + |D| + |H| \leq 9s/\epsilon$.

Note that $|\mathbf{x}_{t+1}| \leq \epsilon/(4s)$. Taking $\alpha = 2$ in Lemma 3.3, we know that

$$\|(\Phi(\mathbf{x} - \mathbf{x}_{[t]}))_{\Gamma(H \cup D)}\|_1 \leq 4 \cdot \beta\epsilon d \left(\frac{3}{2} + \frac{\epsilon}{2} + 2s \cdot \frac{\epsilon}{4s}\right) \leq 10\beta\epsilon d.$$

By the isolation property, there are at most $\frac{9s}{\epsilon} \cdot \frac{\epsilon\theta}{18} = \frac{\theta s}{2}$ elements in T that are not isolated in at least $(1-\zeta)d$ nodes from other elements in T. This implies that at least $\theta s/2$ elements in D are isolated in at least $(1-\zeta)d$ nodes from other elements in T.

A decoy at position i receives at least $\epsilon/(4s)$ noise in at least $(1/2 - \zeta)d$ isolated nodes of $\Gamma(\{i\})$; hence, in total, a decoy element receives at least $\epsilon(1/2 - \zeta)d/(4s)$ noise. Therefore, the $\theta s/2$ decoys overall should receive noise at least

$$\frac{\epsilon(\frac{1}{2} - \zeta)d}{4s} \cdot \frac{\theta s}{2} > 10\beta\epsilon d \geq \|(\Phi(x - x_t))_{\Gamma(H \cup D)}\|_1,$$

which is a contradiction. Therefore, $|D| < \theta s$. □

Now we are ready to show Theorem 3.2.

PROOF OF THEOREM 3.2 The proof is essentially the same as that of Porat and Strauss [2012, Lemma 4]. It follows from Lemma 3.4 that with appropriate choices of constants, that there are at most $\zeta s/4$ decoys and at least $(1-\zeta/4)s$ elements i in $\text{supp}(\mathbf{y})$ satisfying $|\mathbf{x}_i - \mathbf{x}'_i| \leq \eta/(4s)$. Let $I' = I \cap \text{supp}(\mathbf{y})$. We describe below the construction of $\widehat{\mathbf{x}}$, $\widehat{\mathbf{y}}$, and $\widehat{\mathbf{z}}$.

—Elements $i \in \text{supp}(\widehat{\mathbf{x}})$ with a good estimate (to within $\pm\eta/(4s)$ contribute $\mathbf{x}_i - \widehat{\mathbf{x}}_i$ to $\widehat{\mathbf{z}}$. There are at most s of these, each contributing $\eta/(4s)$, for a total contribution of $\eta/4$ to $\widehat{\mathbf{z}}$.
—Elements $i \in \text{supp}(\widehat{\mathbf{x}})$ with a bad estimate (not to within $\pm\eta/(4s)$) contribute $\mathbf{x}_i - \widehat{\mathbf{x}}_i$ to $\widehat{\mathbf{y}}$. There are at most $\zeta s/4$ of these.
—Elements $i \in \text{supp}(\mathbf{z}) \setminus \text{supp}(\widehat{\mathbf{x}})$ contribute $\mathbf{x}_i$ to $\widehat{\mathbf{z}}$. The ℓ_1 norm of these is at most $\|\mathbf{z}\|_1$.
—Elements $i \in I' \setminus \text{supp}(\widehat{\mathbf{x}})$ with a good estimate that are nevertheless displaced by another element $i' \in \text{supp}(\widehat{\mathbf{x}}) \setminus \text{supp}(\mathbf{y})$ with a good estimate contribute to $\widehat{\mathbf{z}}$. There are at most s of these. While the value $\mathbf{x}_i$ may be large and make a large contribution

to $\widehat{\mathbf{z}}$, this is offset by $\mathbf{x}_{i'}$ satisfying $|\mathbf{x}_{i'}| \geq |\widehat{\mathbf{x}}_{i'}| - \eta/(4s) \geq |\widehat{\mathbf{x}}_i| - \eta/(4s) \geq |\mathbf{x}_i| - \eta/(2s)$, which contributes to $\mathbf{z}$ but not to $\widehat{\mathbf{z}}$. Thus the net contribution to $\widehat{\mathbf{z}}$ is at most $\eta/(2s)$ for each of the s of these i, for a total $\eta/2$ contribution to $\widehat{\mathbf{z}}$.
—Elements $i \in I' \setminus \text{supp}(\widehat{\mathbf{x}})$ that themselves have bad estimates or are displaced by elements with bad estimates contribute $\mathbf{x}_i$ to $\widehat{\mathbf{y}}$. There are at most $\zeta s/4$ bad estimates overall, so there are at most $\zeta s/4$ of these.
—Elements $i \in I \setminus I'$ contribute to $\widehat{\mathbf{y}}$. There are at most $\zeta s/2$ of these.

It is clear that $|\text{supp}(\widehat{\mathbf{y}})| \leq \zeta s$ and $\|\widehat{\mathbf{z}}\|_1 \leq \|\mathbf{z}\|_1 + \eta$, as desired. The runtime is easy to verify. □

To complete the construction of a weak recovery system, we refer the reader to Section 2.3 to show that a bipartite expander as required by Theorem 3.2 exists. We show, by probabilistic methods, that it can be attained by both one-layer and two-layer hashing schemes, with appropriate parameters. For example, if we combine Lemma 2.8, Lemma 2.9, and Theorem 3.2, we have a clean formulation, in the language of expanders, of the result on weak system in Porat and Strauss [2012].

4. IDENTIFICATION OF HEAVY HITTERS

In the previous section, we showed how to estimate all candidates in a candidate set I quickly. The main bottleneck in a highly efficient algorithm is finding a non-trivial set $I \subset [N]$ of candidates, which we address in this section.

The overall strategy is as follows. Using the two-layer hashing scheme (B_1, d_1, B_2, d_2), we expect that a heavy hitter dominates the first-layer buckets where it lands in $\Omega(d_1)$ repetitions. In each of these repetitions, it is a heavy hitter in a signal of length B_1, and we expect to recover it using the Weak algorithm applied to the signal of length B_1 with $I = [B_1]$. After finding the heavy buckets in each repetition, the remaining problem is to extract the position of a heavy hitter i from the $\Omega(d_1)$ repetitions that contain i. Recall, as previewed in the introduction, that we shall assign to each index $i \in [N]$ a message $\mathbf{m}_i$, which uniquely identifies the index i. The message will be encoded in the measurement matrix $\mathbf{\Phi}$, and we expect to recover the message $\mathbf{m}_i$ for heavy hitters i from the measurements $\mathbf{\Phi x}$ and thus the index i. The recovery of the message is to be done block by block, which motivates the following definition of *Weak List Recovery Criterion*.

Definition 4.1 (*Weak List Recovery Criterion*). Fix N, s. Suppose that $\mathbf{x} \in \mathbb{R}^N$ can be written as $\mathbf{x} = \mathbf{y} + \mathbf{z}$, where $|\text{supp}(\mathbf{y})| \leq s$ and $\|\mathbf{z}\|_1 \leq 3/2$. Let $\mathbf{m} = (\mathbf{m}_1, \ldots, \mathbf{m}_N)$, where each $\mathbf{m}_i$ is a binary string (also called a *message*) of length β. Suppose $\widehat{\mathbf{m}}$ is a list of possible index-message pairs, that is, $\widehat{\mathbf{m}} \subseteq [N] \times \{0,1\}^\beta$. We say that $\widehat{\mathbf{m}}$ is *correct in the weak list recovery sense* if $(i, \mathbf{m}_i) \in \widehat{\mathbf{m}}$ for at least $|\text{supp}(\mathbf{y})| - s/8$ indices i in $\text{supp}(\mathbf{y})$.

The encoding/decoding scheme is given in Algorithm 2. We break each message $\mathbf{m}_i$ associated with position i into d_1 blocks, $\mathbf{m}_{i,1}, \ldots, \mathbf{m}_{i,d_1}$. Note that $\mathbf{m}_i$ could be much longer than $\log N$ bits in order to guarantee a successful list recovery. Now, in the jth repetition of the d_1 repetitions, we obtain a signal $\widetilde{\mathbf{x}}$ of length B. Each $\widetilde{\mathbf{x}}_\ell$ is associated with a message that can be viewed as a weighted sum of $\mathbf{m}_{i,j}$ for positions i hashed into bucket ℓ. If a heavy hitter i is isolated in bucket ℓ and the noise is mild in this bucket, then this weighted sum would be approximately $\mathbf{m}_{i,j}$, and we expect to recover $\mathbf{m}_{i,j}$ from the second-layer hashing, with inner encoding and decoding. Now we assume that we have recovered $\mathbf{m}_{i,j}$ for heavy hitter i in sufficiently many repetitions j. The central difficulty is to match $\mathbf{m}_{i,j}$ with $\mathbf{m}_{i,j'}$ with $j \neq j'$ in order to find enough fraction of $\mathbf{m}_i$ in the end. In order to solve this, we shall encode some linking information in

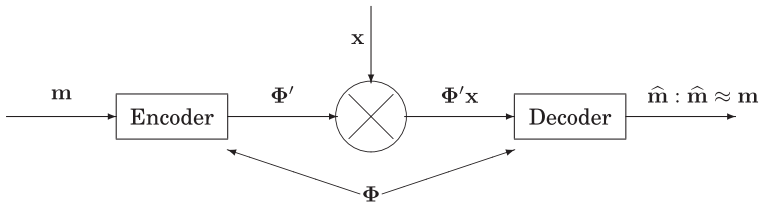

Fig. 5. The encoder and decoder agree on some matrix $\mathbf{\Phi}$. The encoder takes messages $\mathbf{m}$ and produces a measurement matrix $\mathbf{\Phi}'$ based on $\mathbf{m}$ and $\mathbf{\Phi}$. The system takes input $\mathbf{x}$ and produces measurements $\mathbf{\Phi}'\mathbf{x}$, from which the decoder tries to recover $\widehat{\mathbf{m}}$ in the sense of weak list recovery.

ALGORITHM 2: Encding/Decoding Paradigm.

// Encoding with (B_1, d_1, B_2, d_2) hashing scheme
for $i = 1$ to N **do**
  Break: Break the information of i into d_1 blocks
  Outer encoding: Encode the blocks with cluster info (from a regular expander graph) and against errors, getting $\{\mathbf{m}_{i,j}\}_{j=1}^{d_1}$
end for
for $j = 1$ to d_1 **do**
  Inner encoding: Encode $\mathbf{m}_{i,j}$, for $i \in [N]$
end for
// Decoding with (B_1, d_1, B_2, d_2) hashing scheme
for $j = 1$ to d_1 **do**
  Inner decoding: Recover $\widehat{\mathbf{m}}_j$ in the Weak List sense
  Record Side Info: Tag each element of $\widehat{\mathbf{m}}_j$ with j
end for
Outer decoding: From $\widehat{\mathbf{m}} = \bigcup_j \widehat{\mathbf{m}}_j$'s, find block clusters and correct errors; produce I

the node that will enable us to match $\mathbf{m}_{i,j}$ with $\mathbf{m}_{i,j'}$. This will be the topic of the next subsection, in which we shall use the Parvaresh-Vardy code to overcome this difficulty.

Next we illustrate our idea of encoding with a simple case of the sparse recovery problem, where we wish to find k heavy hitters among B positions. We shall encode messages with length $\beta = \log(B/k)$ using $k\log(B/k)$ measurements and recover the messages associated with $\Omega(k)$ heavy hitters in time approximately B. To better illustrate the idea, we refer the reader to Figure 5.

LEMMA 4.2. *Fix k, B, ϵ, where $B = \Omega(k/\epsilon)$ and consider the sparse recovery problem of finding k heavy hitters among B positions. Let $\beta = O(\log(B/k))$. There is a coding scheme to encode messages of length β using $m = O((k/\epsilon)\log(B/k))$ measurements and recover the messages in the weak list recovery sense with decoding running in time $O(B\log^3(B/k))$. This scheme also uses a lookup table of size β.*

PROOF. As an outer code, use Reed-Solomon over an alphabet of size $\beta/\log\beta$. This is concatenated with a random code of length $\log\beta$ as an inner code. The inner code can be decoded in constant time from a lookup table of size β, and the outer code can be decoded by solving a linear system of size approximately β in time $O(\beta^2)$. Hence, for each index $i \in [B]$, the message $\mathbf{m}_i$ of length β is encoded into a longer message $\mathbf{m}'_i$ of length β', where $\beta' = \Theta(\beta)$. It suffices to demonstrate how to encode and decode the longer messages $\mathbf{m}'_i$.

We use a Weak system (Theorem 3.2) with a $(\Theta(k), d, \epsilon)$-bipartite expander that exhibits a $(\Theta(k), d)$ hashing scheme and satisfies $(O(k/\epsilon), \epsilon, O(1))$-isolation property, where $d = \Theta(\log(B/k)) \geq \beta'$. Let $\mathbf{\Phi}$ be the adjacency matrix of the bipartite expander. Without loss of generality, we may assume that $\beta' = d$.

Now we describe the construction of the new measurement matrix $\mathbf{\Phi}'$, which has twice as many rows as $\mathbf{\Phi}$. Viewing $\mathbf{\Phi}$ as a hashing matrix with β' repetitions. For each $i \in [B]$ and $j \in [\beta']$, we need to encode the jth bit of the messages $\mathbf{m}'_i$ in the jth repetition. As there are β' repetitions, a total of β' bits will be encoded for each index $i \in [B]$, as desired. For each row ρ of $\mathbf{\Phi}$ in the jth repetition of hashing, we construct a $2 \times N$ submatrix ρ' as follows. For each $i \in [B]$, the ith column of ρ' is $\binom{\rho_i}{0}$ when $\mathbf{m}'_{i,j} = 1$ and $\binom{0}{\rho_i}$ when $\mathbf{m}'_{i,j} = 0$. Note that either is a column of two zeroes when $\rho_i = 0$. In this way, each row of $\mathbf{\Phi}$ induces two rows of $\mathbf{\Phi}'$.

Finally, we show how to recover the messages. To decode one bit, consider any $\binom{a}{b}$ to be a *relaxed* encoding equivalent to $\binom{\rho_i}{0}$ if $|a| > |b|$ and $\binom{0}{\rho_i}$ otherwise, where ρ is a row of $\mathbf{\Phi}$. We know that there exist $\Omega(k)$ heavy hitters, each dominates the buckets where it lands in $\Omega(d)$ repetitions. In each such repetition, our bit encoding scheme ensures that the associated bit can be recovered successfully, and, hence, for each such heavy hitter, we shall collect $\Omega(d)$ bits, enough to recover the encoded message of β' bits and thus the original message of β bits (using Theorem 3.2 for the weak system with $I = [B]$).

The runtime is $O(B\beta^2 \log(B/k))$ for exhaustive recovery in the Weak system.

Remark 4.3. In the proof above, the matrix $\mathbf{\Phi}$ is not necessarily the matrix of a one-layer hashing scheme. If Φ is a "layer-structured" matrix, then the same row-doubling construction $\mathbf{\Phi}'$ can be employed. For the one-layer (B, d) hashing scheme, the matrix $\mathbf{\Phi}$ can be viewed as having B layers, where each corresponds to a repetition. For the two-layer (B_1, d_1, B_2, d_2)-hashing scheme, the matrix $\mathbf{\Phi}$ can be viewed as having $B_1 B_2$ layers (each layer is a repetition in the second-layer hashing). This observation will be used in our main construction (Lemma 4.5).

Remark 4.4. The Reed-Solomon code is used in the proof. In general, any code that has a constant rate and constant error radius and can be decoded in linear time (up to polylogarithmic factors) will work. The decoding runtime in the lemma statement will be adjusted accordingly.

4.1. Expander Encoding

Parameters. We assume that the constants $\beta > 0$ and $\gamma \in (0, 1)$ are fixed; the parameters B_1, d_1, B_2, d_2 are as in Lemma 2.12 such that $B_1 = \Theta((\frac{k}{\epsilon^2})^{1+\beta} \log \frac{N}{k})$ and $d_1 = (\frac{1}{\epsilon} + (\frac{\log N}{\log k})^{\gamma}) \frac{\log N}{\log(B_1/k)}$; $c \leq m$ are constant integers; h is an integer; and $\epsilon = O((\frac{\alpha}{m})^{\frac{m}{m-c}} (\frac{\log(B_1/k)}{\log N})^{\gamma})$.

Let G be a graph of d_1 nodes with constant degree δ that satisfies Theorem 2.3, and α, ζ, κ be constants provided by Theorem 2.4 when applied to G. Without loss generality, we can assume that $\alpha \leq 1/2$. Adjust the hidden constants together with c, m, and h appropriately (depending on β and γ) such that

(a) $B_1 > d_1$;
(b) $(h-1)m \log_{B_1} N < \alpha d_1$;
(c) $(\alpha d_1 - (h-1)m \log_{B_1} N) \cdot h^m > d_1^c$;
(d) $c \geq \log \delta / \log \kappa$.

We note that an instance of m, h is to choose $m \geq c(1 + 1/\gamma)$ and $h = \Theta(d_1^{c/m})$.

Encoding. We shall use Reed-Solomon for inner encoding. Next, we define our outer coding, which uses the Parvaresh-Vardy code [Parvaresh and Vardy 2005]. Take N disconnected copies of G and call the union G_N, where each node is indexed by a pair $(i, r) \in [N] \times [d_1]$. See Figure 6. Also, let $\mathbb{F}$ be a field such that $|\mathbb{F}| = \Theta(B_1)$ is a power

of 2 and $E(x)$ be an irreducible monic polynomial over $\mathbb{F}$ such that $\deg E(x) = \log_{B_1} N$. View each $i \in [N]$ as a polynomial f over $\mathbb{F}$ with degree $\log_{B_1} N - 1$. For each $(i, r) \in G_N$, associate with it an element $p(i, r) \in \mathbb{F}^{m+1}$ as

$$p(i,r) = (x_{i,r},\ f(x_{i,r}), (f^h \bmod E)(x_{i,r}), \ldots, (f^{h^{m-1}} \bmod E)(x_{i,r})),$$

where f is a polynomial associated with $i \in [N]$ and $x_{i,r} \in \mathbb{F}$ so $x_{i,r}$ are distinct for different r. This is possible because of Property (a).

Attach to a node (i, r) a message $\mathbf{m}_{i,r}$ containing the information of $p(i, r)$ as well as $H(i, v_1(r)), \ldots, H(i, v_\delta(r))$, where $v_1(r), \ldots, v_\delta(r)$ are the neighbours of r in G, and $H(i, j) \in [B_1]$ gives the bucket index where i lands in the jth outer hashing repetition. It is clear that $\mathbf{m}_{i,r}$ has $\Theta(\log B_1) = O(d_2)$ bits, and therefore we can encode it in d_2 hash repetitions; see Lemma 4.2.

Decoding. In each of the d_1 repetitions, we shall recover $O(k/\epsilon)$ heavy buckets and thus obtain $O(k/\epsilon)$ nodes with their messages. Even when the messages are recovered correctly, we only know that a message corresponds to $\mathbf{m}_{i,r}$ for some $i \in [N]$, and we do not know which i it is. However, if we can determine that enough messages are associated with the same i, then we would have obtained enough $p(i, r)$ for different values of r, and then we should be able to find f and thus recover the position i.

To determine enough $p(i, r)$ for the same i, we do clustering as follows. Suppose that there are k heavy hitters at position $i_1, \ldots, i_k$. Let $\widetilde{G}$ be a graph of $d_1 \times O(k/\epsilon)$ nodes, arranged in a $d_1 \times O(k/\epsilon)$ grid. For now we assume that the messages are recovered correctly for each heavy hitter i in all d_1 repetitions. (This means that there are no collisions, and the noise in the buckets are all small.) Each message has the form $p(i, r), h_1, \ldots, h_\delta$, where $h_j = H(i, v_j(r))$ for $1 \le j \le \delta$. Add an arc $(i, r) \to (h_j, v_j(r))$ for each $1 \le j \le \delta$.

Since the messages are recovered correctly, the graph $\widetilde{G}$ will contain several disjoint copies of the expander graph G, say, $G_{i_1}, \ldots, G_{i_k}$, although each G_{i_j} is not necessarily aligned within the same column in $\widetilde{G}$. There will be arcs incoming to G_{i_j} from nodes not in any G_{i_j}, but there are no outgoing arcs from G_{i_j}. In this case, we can recover each G_{i_j} perfectly and collect the full set $\{\mathbf{m}_{i_j,r}\}_{r=1}^{d_1}$ and thus recover i_j. Let us rearrange the nodes within each row and align each copy of G in the same column for clarity. In this case, the columns $i_1, \ldots, i_k$ are exact copies of the expander graph G. See Figure 7 for an illustration.

The heavy hitters may not, however, be recovered in some repetitions and the messages could be seriously corrupted. When we are adding the arcs, we introduce two kinds of errors, respectively:

(i) We lose a node in G_{i_j}, that is, the node is not present in $\widetilde{G}$ because the heavy hitter i_j is not recovered in that repetition;
(ii) We connect a node in G_{i_j} to a node in some other $G_{i_{j'}}$ $(j \neq j')$, due to an errorous message.

As before, we align each "ideal copy" of G in the same column. See Figure 8 for an example. We know that for a heavy hitter i, only a few messages $\{\mathbf{m}_{i,r}\}_r$ are ruined, and the ith column of G_N will contain a large connected subgraph G' of G, by Theorem 2.4. Hence, if we start a breadth-first search from an appropriate node with depth $c \log_\delta d_1$, then the whole G' will be visited. In other words, we shall obtain a large set of $\{p(i, r)\}$, only a small number of which will be associated with the same i, but we expect to obtain enough $\{p(i, r)\}$ of the same i, which turns out to be sufficient to extract f associated

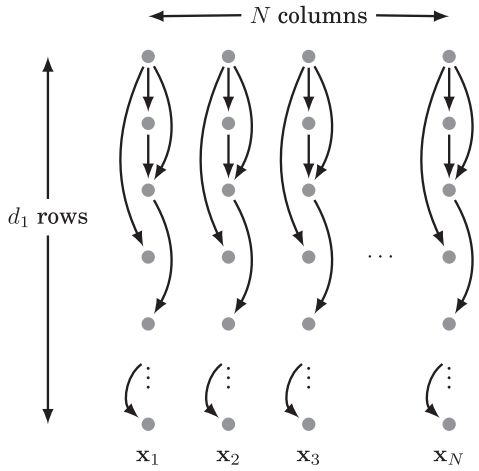

Fig. 6. Underlying graph G_N. Suppose that $\mathbf{x}_1$ is in the tail and that $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_N$ are heavy hitters.

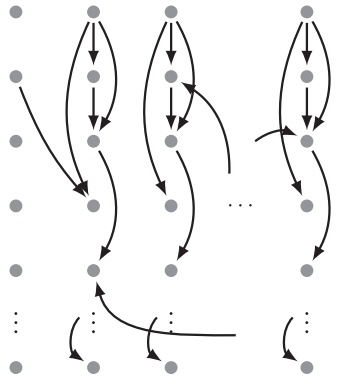

Fig. 7. Recovered graph $\tilde{G}$ in ideal situation, with expander copies clairvoyantly aligned in a column. Since the first column corresponds to a tail item and is thus not expected to be recovered, it is almost absent in the recovered graph. For each heavy-hitter column, the whole copy of the expander graph is expected to be recovered. There may exist some arcs from a non-heavy-hitter column to a heavy-hitter column but not vice versa.

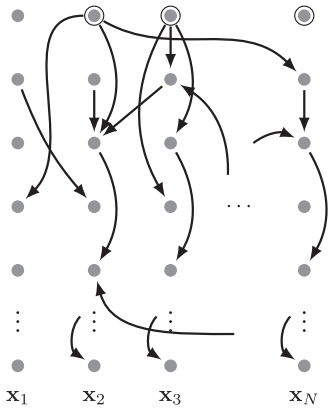

Fig. 8. Recovered graph $\tilde{G}$, with "supposed" expander copies clairvoyantly aligned in columns. The first column corresponds to a tail item, so it is almost absent. The top node in the second column is corrupted, so it points to wrong columns but, nevertheless, the correct rows, because the row information is hard wired. The top node in the third column is correctly recovered, but the second node in the column is corrupted. The top node in the last column has a small bucket value in the first repetition, so it is absent $\tilde{G}$. If we perform BFS at the top node in the third column, then we may include a lot of nodes in the second column.

with i using a good error-correcting code such as the Parvaresh-Vardy code that allows us to recover the codeword from a large fraction of errors. Without attempting to identify the "appropriate node" described above, we shall perform this breadth-first search on every node in $\widetilde{G}$.

Guarantee. We shall show that the system described above meets the aforementioned guarantee.

LEMMA 4.5. *Let $\beta > 0$ and $\gamma \in (0,1)$ be constants. The encoding and decoding strategy of Section 4.1 are correct in the sense of the guarantee of that section, against the channel described in that section. It uses $O(\epsilon^{-2} s \log N)$ measurements and runs in time $O(s^{1+\beta}\,\mathrm{poly}(\log N, 1/\epsilon))$, provided that $N = \Omega(\max\{s^2, s/\epsilon^2\})$ and $\epsilon = O((\frac{\log s}{\log N})^{\gamma})$.*

PROOF. Combining Lemma 2.10, Lemma 2.8, and Lemma 2.12, one can show that there exists an $(4s, d_1 d_2, \epsilon)$-bipartite expander such that

(a) the bipartite expander exhibits a (B_1, d_1, B_2, d_2) hashing structure, where the parameters are as in Lemma 2.12, and each second-layer hashing satisfies $(O(s/\epsilon), O(\epsilon), O(1))$-isolation property;
(b) the bipartite expander satisfies the $(O(s/\epsilon), O(\epsilon), O(1))$-isolation property.

As in the proof of Lemma 3.4, suppose that $|\mathbf{x}_i| \geq \epsilon/s$ for all $i \in \mathrm{supp}(\mathbf{y})$; otherwise we can place the violated i's into $\mathbf{z}$, causing $\|z\|_1$ to increase by at most $s \cdot \epsilon/s = \epsilon$, so we would have $\|z\|_1 \leq 2$. Call the elements in $\mathrm{supp}(\mathbf{y})$ heavy hitters. If $|\mathrm{supp}(\mathbf{y})| \leq s/8$, then our goal is automatically achieved, so we assume that $|\mathrm{supp}(\mathbf{y})| > s/8$.

Step 1. Overall, we know from Lemma 3.4 that we have at most $s/8$ decoys, or we can recover $|\mathrm{supp}(\mathbf{y})| - s/8$ heavy hitters from the second-layer bucket values, where successful recovery means that each of them dominates in at least $\alpha_2 d_1 d_2$ second-layer buckets, that is, the bucket noise is at most $\nu = \epsilon/(2s)$. For each of them, in at least $\beta_1 d_1$ of d_1 outer repetitions, it dominates in at least $\beta_2 d_2$ inner repetitions, where $(1-\beta_1)(1-\beta_2) > 1-\alpha_2$. Because whenever an element dominates in the second-layer bucket, it must dominate the first-layer bucket incident to that second-layer bucket, we conclude that there exists a set $S \subseteq \mathrm{supp}(\mathbf{y})$, $|S| \geq |\mathrm{supp}(\mathbf{y})| - s/8$, such that each $i \in S$ dominates at least $\beta_1 d_1$ first-layer buckets among all d_1 repetitions, and in each of such repetitions, it dominates at least $\beta_2 d_2$ second-layer buckets.

We can choose the hidden constants in the bipartite expander parameters such that $\beta_1 \geq 1-\zeta$ and β_2 matches the error tolerance of the coding scheme we described in Lemma 4.2, where ζ is the parameter we set in Section 4.1.

Step 2. It follows from above that each $i \in S$ will be recovered in at least $\beta_1 d_1$ outer repetitions, since its bucket value is $\geq \epsilon/s - \nu \geq \epsilon/(2s)$. Indeed, in every repetition of outer hashing, we collect top $O(s/\epsilon)$ (first-layer) buckets, so we will include every bucket with value $\geq \epsilon/(2s)$ and thus the heavy hitter i. In this case, the message associated with the heavy hitter will be recovered correctly, as the inner encoding can tolerate $1-\beta_2$ fraction of error. Therefore, we know that for each $i \in S$, the associated messages will be correctly recovered in $\beta_1 d_1$ outer repetitions.

Step 3. As described in the previous section, we shall form a graph $\tilde{G}$. Note that for $i \in S$, $\beta_1 d_1$ nodes in the column are good nodes (i.e., with correct message). For each of them, perform a breadth-first search of $O(\log_\delta d_1)$ steps, collecting at most d_1^c nodes. Since the column contains at most $(1-\beta)d_1 \leq \zeta d_1$ bad nodes, by Theorem 2.4 and Property (d) of our choices of parameters, there exists a good node in the ith column such that if we perform a breadth-first search of $c \log_\delta d_1$ steps, we shall collect αd_1 good nodes that are all in the ith column. The Parvaresh-Vardy code with our choice of parameters (Properties (b) and (c)) enables us to include it in the list. We shall briefly describe the decoding below. Having collected at most d_1^c points $(x, r(x)) \in \mathbb{F}^{m+1}$, we consider all polynomials $Q(x, y_0, \ldots, y_{m-1})$ of degree at most $d_X = \alpha d_1 - (h-1)m \log_{B_1} N$ in its first variable and at most $h-1$ in each such that $Q(x, r(x)) = 0$ for all i. Our choice of parameters (Property (c), that is, $d_X h^m > d_1^c$) guarantees that such Q exists. Then,

the existence of αd_1 good nodes (in the BFS visited nodes) indicates that the equation

$$Q(x, f_i(x), (f_i^h \bmod E)(x), \ldots, (f_i^{h^{m-1}} \bmod E)(x)) = 0$$

has αd_1 roots in $\mathbb{F}$ for f_i corresponding to the coordinate $i \in S$. By our choice of parameters (Property (b)), the univariate polynomial $Q(x)$ has degree less than αd_1 and must be identically zero. This means that $f_i(x)$ is a root of $Q^*(z) = Q(x, z, z^h, \ldots, z^{h^{m-1}}) = 0$ over $\mathbb{F}[x]/E(x)$. We can find f_i by factoring Q^* and thus recover the position i of the heavy hitter.

In the end, our candidate list will contain all $i \in S$, that is, we shall have recovered $|\operatorname{supp}(\mathbf{y})| - s/8$ heavy hitters.

Number of Measurements. The number of measurements is $O(B_2 d_1 d_2) = O(\epsilon^{-2} s \log(N/s))$.

Size of Lookup Table. The inner decoding uses a lookup table of size $O(\log B_1) = O(\frac{s}{\epsilon} + \log\log N)$. The algorithm also stores the expander graph G, which takes space $O(d_1)$. Both are smaller than the space cost of the recovered graph $O(sd_1/\epsilon)$, so their contribution to the space complexity can be neglected.

Runtime. For each of d_1 repetitions, we shall recover every bucket with value $\geq \epsilon/(2s)$ in $O(B_1 \log^3(B_1/k)) = O(s^{1+\beta} \operatorname{poly}(\log N, 1/\epsilon))$ time. There are $O(s/\epsilon)$ of them in each repetition. Then we form a graph of size $O(sd_1/\epsilon)$. Forming this graph takes time $O(s^{1+\beta} \operatorname{poly}(\log N, 1/\epsilon))$ from the argument above. Then we do breadth-first search of $c \log_\delta d_1$ steps on every node in $\widetilde{G}$. Each BFS takes $O(d_1^c)$ time. Each decoding of the BFS nodes takes $\operatorname{poly}(d_1, \log|B_1|) = \operatorname{poly}(\log N, 1/\epsilon)$ time and can be done deterministically (see, e.g., Clifford et al. [2009, Theorem 4.3]), since $|\mathbb{F}|$ has a small characteristic. Hence extracting heavy hitters i from the recovered graph $\tilde{G}_N$ takes time $O(s \operatorname{poly}(\log N, 1/\epsilon))$ and, therefore, the overall runtime is $O(s^{1+\beta} \operatorname{poly}(\log N, 1/\epsilon))$. In the end, we shall obtain a candidate list of size $O(s \operatorname{poly}(\log N, 1/\epsilon))$. □

5. TOPLEVEL SYSTEM

Now we define a Toplevel system, similarly to Gilbert et al. [2012] and Porat and Strauss [2012], which is an algorithm that solves our overall problem.

Definition 5.1. An *approximate sparse recovery system* (briefly, a *Toplevel* system) consists of parameters N, k, ϵ; an m-by-N *measurement matrix* $\mathbf{\Phi}$; and a *decoding algorithm* $\mathcal{D}$ that satisfy the following property: For any vector $\mathbf{x} \in \mathbb{R}^n$, given $\mathbf{\Phi x}$, the system approximates $\mathbf{x}$ by $\widehat{\mathbf{x}} = \mathcal{D}(\mathbf{\Phi x})$, which satisfies

$$\|\widehat{\mathbf{x}} - \mathbf{x}\|_1 \leq (1+\epsilon)\|\mathbf{x}_{[k]} - \mathbf{x}\|_1.$$

Using this definition, we restate our main result from Theorem 1.1 in a slightly different form.

THEOREM 5.2. *Let $\beta > 0$ and $\gamma \in (0,1)$ be constants. There exists $\alpha = \alpha(\beta,\gamma) \in (0,1)$ such that Algorithm 3 yields a Toplevel system and uses $O(\epsilon^{-2} k \log N)$ measurements and runtime $O(k^{1+\beta} \operatorname{poly}(\log N, 1/\epsilon))$, provided that $N = \Omega(\max\{k^2, k/\epsilon^2\})$ and $\epsilon = O((\frac{\log k}{\log N})^\gamma)$. It also uses a lookup table of size $O(\log N)$.*

The proof follows easily using the results on the weak system. We need Lemma 4.5 for identification and Theorem 3.2 for estimation.

PROOF. Suppose that in Lemma 4.5, the exponent of $1/\epsilon$ in runtime is $c = c(\beta,\gamma) > 2$. Choose $\alpha < 1$ such that $\alpha^c > 1/2$. Assume that $\epsilon \leq 1/2$.

ALGORITHM 3: Toplevel System

Input: $\mathbf{\Phi}$, $\mathbf{\Phi x}$, N, k, ϵ
Output: $\widehat{\mathbf{x}}$
  $\widehat{\mathbf{x}} \leftarrow 0$
  $\mu \leftarrow \mathbf{\Phi x}$
  for $j \leftarrow 0$ to $\log k$ **do**
    Run Algorithm 2 on μ with length N, $s \leftarrow k/2^j$, $\eta \leftarrow \epsilon\alpha^j(1-\alpha)^2$ and obtain a candidate list I
    Run Algorithm 1 on candiate set I with $s \leftarrow k/2^j$ and $\eta \leftarrow \epsilon\alpha^j(1-\alpha)$
    Let $\mathbf{x}'$ be the result
    $\widehat{\mathbf{x}} \leftarrow \widehat{\mathbf{x}} + \mathbf{x}'$
    $\mu \leftarrow \mu - \mathbf{\Phi x}'$
  end for
  return $\widehat{\mathbf{x}}$

Using Lemma 4.5 for identification and Theorem 3.2 for estimation, with appropriate choice of constants, we claim that at the beginning of the jth step, $\mathbf{x} = \mathbf{y} + \mathbf{z}$, where $|\operatorname{supp}(\mathbf{y})| \le k/2^j$ and

$$\|\mathbf{z}\|_1 \le 1 + \epsilon(1 + \alpha + \alpha^2 + \cdots + \alpha^{j-1})(1-\alpha).$$

We shall prove this claim by induction. Letting $s = k/2^j$, $\eta = \epsilon(1-\alpha)^2\alpha^j$ for identification, which introduces at most η into the tail and the tail remains at most 3/2 by assuming that all head items, that is, the non-zero elements in $\mathbf{y}$, are all larger than η/s.

The identification procedure returns a candidate I that contains 3/4 fraction of $\operatorname{supp}(\mathbf{y})$ (note that when the head is flat, we can change $\operatorname{supp}(\mathbf{y})$ to be a superset that satisfies this condition without changing the norm of $\mathbf{z}$). Then the estimation procedure, with $s = O(k/2^j)$ and $\eta = \epsilon\alpha^{j+1}(1-\alpha)$, will give us

$$\mathbf{x} = \widehat{\mathbf{x}} + \widehat{\mathbf{y}} + \widehat{\mathbf{z}},$$

where $|\operatorname{supp}(\mathbf{x})| = O(s)$, $|\operatorname{supp}(\hat{\mathbf{y}})| \le s/2$ and

$$\|\widehat{\mathbf{z}}\|_1 \le \|\mathbf{z}\|_1 + \epsilon(1-\alpha)^2\alpha^j + \epsilon\alpha^{j+1}(1-\alpha) = \|\mathbf{z}\|_1 + \epsilon(1-\alpha)\alpha^j.$$

It is easy to verify that $\|\hat{\mathbf{z}}\|_1 \le 1 + \epsilon \le 3/2$, and thus Lemma 4.5 for identification and Theorem 3.2 can be applied at the next round and the inductive hypothesis is satisfied. Therefore, in the end, we obtain that

$$\|\widehat{\mathbf{x}} - \mathbf{x}\|_1 \le (1+\epsilon)\,\|\mathbf{x} - \mathbf{x}_k\|_1.$$

The number of measurements used for identification is

$$O\left(\sum_j \frac{1}{\epsilon^2\alpha^{2j}} \cdot \frac{k}{2^j}\log N\right) = O\left(\frac{k}{\epsilon^2}\log N \sum_j \left(\frac{1}{2\alpha^2}\right)^j\right) = O\left(\frac{k}{\epsilon^2}\log N\right)$$

and the number of measurements used for estimation is

$$O\left(\sum_j \frac{1}{\epsilon^2\alpha^{j}} \cdot \frac{k}{2^j}\log N\right) = O\left(\frac{k}{\epsilon^2}\log N \sum_j \left(\frac{1}{2\alpha}\right)^j\right) = O\left(\frac{k}{\epsilon^2}\log N\right),$$

and, hence, the total number of measurements is $O(\epsilon^{-2}k\log(N/k))$ as claimed.

It can be verified in a similar fashion that the total runtime is $O(k^{1+\beta}\operatorname{poly}(\log N, 1/\epsilon))$, for which we need our choice of α satisfying that $\alpha^c > 1/2$. □

Remark 5.3. We note that

(a) the constants in big O-notations and the power in $\text{poly}(\log N, 1/\epsilon)$ depend on β and γ;
(b) as in Remark 2.11, the constraint that $k = O(\sqrt{N})$ could be weakened to $k = O(N^{1-\xi})$ for any $\xi > 0$;
(c) the factor $k^{1+\beta}$ in the runtime is due to our choice of $B_1 = \Omega((k/\epsilon^2)^{1+\beta} \log(N/k))$ such that $\log B_1 = O(\log(B_1/k)) = O(d_2)$. When $k \le (\log N)^c$ for some $c > 0$, since $B_1 = \Omega(k/\epsilon^{2(1+\beta)})$, choosing $B_1 = \Theta(k \log(N/k)/\epsilon^{2(1+\beta)})$ would suffice. It leads to runtime $O(k \,\text{poly}(\log N, 1/\epsilon))$.
(d) For large ϵ, we can take $d_1 = (\log N/\log(B_1/k))^{1+\alpha}$ for $\alpha > 0$, which gives an algorithm which uses more measurements $O(k\epsilon^{-2} \log^{1+\alpha} N)$ but suboptimal by only a logarithmic factor from the best known lower bound.

6. DISCUSSIONS AND OPEN PROBLEMS

6.1. Codes

At the core part of this article lies the following list recovery problem: Suppose that there are $d_1 = \frac{1}{\epsilon} \cdot \frac{\log N}{\log(B/k)}$ lists $L_1, \ldots, L_{d_1}$ with $|L_i| = O(k/\epsilon)$ for all $i = 1, \ldots, d_1$, we want to recover all possible codewords $c = (c_1, \ldots, c_{d_1})$ such that $c_i \in L_i$ for at least $\Omega(d_1)$ different is. It is natural to be tempted to apply Parvaresh-Vardy code directly without the expander structure. Indeed, it works for some configurations of k and ϵ with a runtime of $O(k \,\text{poly}(\log N, 1/\epsilon))$ but only for small k and ϵ. A direct application already fails even for $k = \exp(\sqrt{\log n})$. The runtime resulting from a direct application is also better for very small k; however, obtaining the precise range is difficult and beyond the scope of our work, as it relies on the precise complexity of factorizing a polynomial, which is not explicit in the literature.

Instead, we use an expander structure to reduce the problem to kd_1/ϵ subproblems, each of which has a smaller number of nodes. Specifically, the abstract problem is the following.

Problem 6.1. *Let $\mathcal{C}$ be a q-ary code of block length n. For every sequence of subsets $S_1, \ldots, S_n \subseteq [q]$ such that $\sum_{i=1}^n |S_i| \le \ell$, find all codewords $(c_1, \ldots, c_n) \in \mathcal{C}$ such that $|\{i : c_i \in S_i\}| \ge \alpha n$ in time $O(\text{poly}(\ell, n, \log q))$.*

Note that instead of the usual assumption on individual size of each S_i we have a bound on the sum of the sizes of S_i here. Our choice of parameters is $q = \Theta(B_1)$, $n = d_1 = \Theta((\log_{B_1} N)/\epsilon)$, $\ell = \text{poly}(d_1)$. The rate of the code is $\Theta(\epsilon)$. Our restrictions of ϵ comes from the application of the Parvaresh-Vardy code. Potential extractor codes [Ta-Shma and Zuckerman 2004] and [Cheraghchi 2013] for context would yield improvement over this article.

6.2. Open Problems

Below we list a few open problems.

Restriction on ϵ. The algorithm in this article restricts ϵ to $(\frac{\log k}{\log N})^\gamma$ for any $\gamma > 0$ because of its way of applying the Parvaresh-Vardy code. In a sense, our construction reduces the problem to a list recovery problem, as discussed in the previous subsection. We ask if it is possible to find an improvement by applying a better list recoverable code. The ultimate goal is to relax the restriction of ϵ to $\epsilon \le \epsilon_0$ for some constant $\epsilon_0 > 0$.

Sparse Recovery in ℓ_2/ℓ_1 norm. The ultimate problem is the ℓ_2/ℓ_1 problem with error guarantee as in Equation (1). We hope that the algorithm in this article offers new

ideas for the mixed-norm problem. Again the difficulty is in identification, as an RIP_2 matrix would be sufficient for estimation.

Post-measurement Noise. In many algorithms on the sparse recovery problem, the input to the decoding algorithm is $\mathbf{\Phi x} + \nu$ instead of $\mathbf{\Phi x}$, where ν is an arbitrary noise vector. It is expected that our algorithm, with small changes if necessary, can tolerate substantial noise in ℓ_1 norm, since the underlying structure is of a similar type to Porat and Strauss [2012]. We leave to future work full analysis and possible improved algorithms.

APPENDIX

A. PROOF OF LEMMA 2.8

PROOF. Let p_s be the probability of a fixed set of s elements hashed into less than $(1-\epsilon)ds$ elements. By symmetry, this probability is independent of the s positions and thus is well defined; hence the probability

$$\Pr\{\text{hashing does not give an expander}\} = \sum_{s=2}^{4k} \binom{N}{s} p_s. \tag{2}$$

Our goal is to show that

$$p_s \le \exp\left(-cs \ln \frac{eN}{s}\right) \tag{3}$$

for some absolute constant $c > 2$, for which it suffices to show that

$$p_s \le \exp\left(-cs \ln \frac{N}{k} \ln \frac{Ck}{s}\right) \tag{4}$$

for some $c, C > 0$. Indeed, it follows from Equation (4) that

$$p_s \le \exp\left(-cs \ln \frac{N}{k} \ln \frac{Ck}{s}\right) \le \exp\left\{-cs\left(\ln \frac{N}{k} + \ln \frac{Ck}{s}\right)\right\} = \exp\left(-cs \ln \frac{CN}{s}\right)$$

and Equation (3) holds. Assume for the moment that Equation (3) is proved, and then we can bound Equation (2) to be

$$\begin{aligned}
\sum_{s=2}^{\alpha k} \binom{N}{s} p_s &\le \sum_{s=2}^{\alpha k} \exp\left\{s \ln \frac{eN}{s} - cs \ln \frac{CN}{s}\right\} \\
&\le \sum_{s=2}^{\alpha k} \exp\left\{-(c-1)s \ln \frac{C'N}{s}\right\} \\
&\le \sum_{s=2}^{\alpha k} \exp\left(-(c-1)s \log N\right) < \frac{1}{N^{c'}},
\end{aligned}$$

as desired.

Now we compute p_s. Fix a set S of s elements. Suppose that they are hashed into X_i $(i = 1, \ldots, d)$ buckets in d repetitions, respectively. We have that $1 \le X_i \le s$ and $\sum X_i \le (1-\epsilon)sd$. Define the event

$$E_i(X_i) = \{S \text{ is hashed into } X_i \text{ rows in } i\text{-th reptition}\},$$

and we shall compute $\Pr\{E_i(X_i)\}$.

When E_i happens, there are $s-X_i$ repetitions. Consider that we hash the element one by one, choosing $b_1, \ldots, b_d \in \{1, \ldots, B\}$ sequentially. We have a collision when selecting b_i if $b_i \in \{b_1, \ldots, b_{i-1}\}$. The probability that a collision occurs at step i, even conditioned on $b_1, \ldots, b_{i-1}$, is at most $i/B \le s/B$. Therefore,

$$\Pr\{E_i(X_i)\} \le \binom{s}{s-X_i}\left(\frac{s}{B}\right)^{s-X_i},$$

and, hence,

$$p_s = \sum \Pr\{E_1(X_1), \ldots, E_d(X_d)\} = \sum \prod_{i=1}^{d} \binom{s}{s-X_i}\left(\frac{s}{B}\right)^{s-X_i} = \sum \left(\frac{s}{B}\right)^{sd-\sum X_i} \prod_{i=1}^{d} \binom{s}{s-X_i},$$

where the summation is over all possible configurations of $\{X_i\}$. Invoking the combinatorial identity

$$\sum_{k_1+k_2+\cdots+k_m=n} \binom{r_1}{k_1}\binom{r_2}{k_2}\cdots\binom{r_m}{k_m} = \binom{r_1+r_2+\cdots+r_m}{n} \tag{5}$$

and writing $X = \sum X_i$, we see that

$$p_s \le \sum_{X=d}^{(1-\epsilon)sd} \left(\frac{s}{B}\right)^{sd-\sum X_i} \binom{sd}{sd-\sum X_i} \le \sum_{X=\epsilon sd}^{d} \binom{sd}{X}\left(\frac{s}{B}\right)^{X}$$

Now we invoke the Chernoff bound,

$$\sum_{k=\epsilon n}^{n} \binom{n}{k}\lambda^k \le \left(\frac{e\lambda}{\epsilon}\right)^{\epsilon n}, \quad \lambda < \epsilon, \tag{6}$$

to obtain that

$$p_s \le \left(\frac{es}{\epsilon B}\right)^{\epsilon sd} \le \exp\left(-cs \log \frac{N}{k} \ln \frac{Ck}{s}\right),$$

as desired, where the constants $c, C > 0$ can be made arbitrarily big. □

B. PROOF OF LEMMA 2.9

Proof. Let S be a set of size $s \le L$. We shall bound the probability p_s (which is defined by symmetry) that at least ϵs elements of S collide with each other in at least ζd repetitions. When this happens, there are at least $\epsilon\zeta ds$ colliding element-repetition pairs. As in Lemma 2.8, it suffices to have Equation (4) for some $c, C > 0$ that can be made arbitrarily large.

In one repetition, one element of S collide with others with probability $\le s/B$. By a coupling argument as in Porat and Strauss [2012], among all sd element-repetition pairs with expected $\mu = s^2d/B$ failed pairs, there are at least $\zeta\epsilon sd$ failed pairs with probability

$$\left(\frac{e\mu}{\zeta\epsilon ds}\right)^{\zeta\epsilon sd} = \left(\frac{es}{\zeta\epsilon B}\right)^{\zeta\epsilon sd} \le \exp\left(-cs \log \frac{N}{k} \ln \frac{Ck}{s}\right),$$

as desired, where the absolute constants $C, c > 0$ can be made arbitrary large. □

C. PROOF OF LEMMA 2.10

Proof. Let p_s be the probability of a fixed set of s elements hashed into fewer than $(1-\epsilon)ds$ elements. By symmetry, this probability is independent of the s positions and

thus is well defined. Hence the probability

$$\Pr\{\text{hashing does not give an expander}\} = \sum_{s=2}^{4k} \binom{N}{s} p_s. \tag{7}$$

Similarly to Lemma 2.8, it suffices to show that

$$p_s \le \exp\left(-cs \ln \frac{N}{k}\right). \tag{8}$$

Assume for the moment that this is proved; then we can bound Equation (7) to be

$$\begin{aligned}
\sum_{s=2}^{4k} \binom{N}{s} p_s &\le \sum_{s=2}^{4k} \exp\left\{s \ln \frac{eN}{s} - cs \ln \frac{N}{k}\right\} \\
&\le \sum_{s=2}^{4k} \exp\left\{s \ln(eN) - \frac{c}{2} s \ln(eN)\right\} \quad (k \le \sqrt{N/e}) \\
&\le \sum_{s=2}^{4k} \exp\left(-\left(\frac{c}{2} - 1\right) s \log(eN)\right) < \frac{1}{N^{c'}},
\end{aligned}$$

as desired.

Now we prove Equation (8). Fix a set S of s elements. The outer layer of hashing has d_1 blocks of size B_1, and let Y_i $(i = 1, \ldots, d_1)$ be the number of hashed row of the s elements in ith block. The inner layer has $d_1 d_2$ blocks, indexed by $(i, j)_{1 \le i \le d_1, 1 \le j \le d_2}$ of size B_2, and let X_{ij} be the number of hashed row of the s elements in the (i, j)-th block. Define the events

$$\begin{aligned}
E_i(Y_i) &= \{S \text{ is hashed into } Y_i \text{ rows in } i\text{-th outer block}\} \\
E_{ij}(X_{ij}) &= \{S \text{ hashed into } X_{ij} \text{ rows in } (i, j)\text{-th inner block}\}.
\end{aligned}$$

First, we calculate $\Pr\{E_i\}(Y_i)$. Consider that we pick a row at one time for an element in S in order. When $E_i(Y_i)$ happens, there are at least $s - Y_i$ collisions, and, hence,

$$\Pr\{E_i(Y_i)\} \le \binom{s}{s - Y_i} \left(\frac{s}{B_1}\right)^{s - Y_i}$$

and, similarly,

$$\Pr\{E_{ij}(X_{ij}) | E_i(Y_i)\} \le \binom{Y_i}{Y_i - X_{ij}} \left(\frac{Y_i}{B_2}\right)^{Y_i - X_{ij}}.$$

It follows that

$$\begin{aligned}
p_s &= \sum \Pr\{E_{11}(X_{11}), \ldots, E_{d_1 d_2}(X_{d_1 d_2}) | E_1(Y_1), \ldots, E_{d_1}(Y_{d_1})\} \Pr\{E_1(Y_1), \ldots, E_{d_1}(Y_{d_1})\} \\
&\le \sum \prod_i \Pr\{E_i\}(Y_i) \prod_{i,j} \Pr\{E_{ij}(X_{ij}) | E_i(Y_i)\} \\
&\le \sum \prod_i \binom{s}{Y_i} \left(\frac{s}{B_1}\right)^{s - Y_i} \cdot \prod_{i,j} \binom{Y_i}{X_{ij}} \left(\frac{Y_i}{B_2}\right)^{Y_i - X_{ij}} \\
&\le \sum \left(\frac{s}{B_1}\right)^{sd_1 - \sum Y_i} \left(\frac{s}{B_2}\right)^{d_2 \sum Y_i - \sum X_{ij}} \prod_i \binom{s}{Y_i} \prod_{i,j} \binom{Y_i}{X_{ij}},
\end{aligned}$$

where the summation is taken over all possible configurations of $\{X_i\}$ and $\{Y_i\}$ so $s \geq Y_i \geq \max_j X_{ij}$ and $\sum X_{ij} \leq (1-\epsilon)sd_1d_2$.

Invoking the combinatorial equality (5) and letting $X = \sum X_{ij}$ and $Y = \sum Y_i$, we obtain that

$$\begin{aligned} p_s &\leq \sum_{Y=d_1}^{sd_1} \binom{sd_1}{Y} \left(\frac{s}{B_1}\right)^{sd_1-Y} \sum_{X=d_1d_2}^{\min\{d_2Y,(1-\epsilon)sd_1d_2\}} \binom{d_2Y}{X} \left(\frac{s}{B_2}\right)^{d_2Y-X} \\ &\leq \sum_{Y=d_1}^{(1-\epsilon/2)sd_1} \binom{sd_1}{Y} \left(\frac{s}{B_1}\right)^{sd_1-Y} \sum_{X=d_1d_2}^{d_2Y} \binom{d_2Y}{X} \left(\frac{s}{B_2}\right)^{d_2Y-X} \\ &\quad + \sum_{Y=(1-\epsilon/2)sd_1}^{sd_1} \binom{sd_1}{Y} \left(\frac{s}{B_1}\right)^{sd_1-Y} \sum_{X=d_1d_2}^{(1-\epsilon)sd_1d_2} \binom{d_2Y}{X} \left(\frac{s}{B_2}\right)^{d_2Y-X} \\ &=: S_1 + S_2. \end{aligned} \tag{9}$$

We bound S_1 and S_2 separately. First,

$$\begin{aligned} S_1 &\leq \sum_{Y=d_1}^{(1-\epsilon/2)sd_1} \binom{sd_1}{Y} \left(\frac{s}{B_1}\right)^{sd_1-Y} \left(1+\frac{s}{B_2}\right)^{d_2Y} \\ &\leq \left(1+\frac{s}{B_2}\right)^{sd_1d_2} \sum_{Y=\frac{\epsilon}{2}sd_1}^{sd_1} \binom{sd_1}{Y} \left(\frac{s}{B_1}\right)^{Y}. \end{aligned}$$

It follows from Chernoff bound (6) that

$$\begin{aligned} S_1 &\leq \left(1+\frac{s}{B_2}\right)^{sd_1d_2} \left(\frac{es}{\frac{\epsilon}{2}B_1}\right)^{\epsilon sd_1/2} \\ &\leq \exp\left\{-\frac{1}{2}\epsilon sd_1\left(\ln\frac{\epsilon B_1}{2es}\right) + sd_1d_2\ln\left(1+\frac{s}{B_2}\right)\right\} \\ &\leq \exp\left\{-\frac{1}{4}\epsilon sd_1\ln\frac{B_1}{k} + c_2\epsilon sd_1d_2\right\} \quad (\text{since } B_1 \gtrsim k/\epsilon^2) \\ &\leq \exp\left\{-c_3 s\ln\frac{N}{k}\right\}, \end{aligned} \tag{10}$$

where the absolute constant $c_2 > 0$ can be made arbitrarily close to 0 and the absolute constant c_3 can be made arbitrarily large.

Now we bound S_2. When $Y \geq (1-\epsilon/2)sd_1$, then

$$\frac{(1-\epsilon)sd_1d_2}{d_2Y} \leq 1 - \frac{\epsilon}{2}.$$

Again invoking Chernoff bound,

$$\sum_{X=d_1d_2}^{(1-\epsilon)sd_1d_2} \binom{d_2Y}{X} \left(\frac{s}{B_2}\right)^{d_2Y-X} \leq \left(\frac{es}{\frac{\epsilon}{2}B_2}\right)^{d_2Y-(1-\epsilon)sd_1d_2} \leq \left(\frac{s}{C'k}\right)^{d_2Y-(1-\epsilon)sd_1d_2},$$

where $C' > 0$ is an absolute constant that can be made arbitrarily large. So

$$\begin{aligned} S_2 &\leq \sum_{Y=(1-\epsilon/2)sd_1}^{sd_1} \binom{sd_1}{Y} \left(\frac{s}{B_1}\right)^{sd_1-Y} \left(\frac{s}{C'k}\right)^{\epsilon sd_1 d_2/2} \\ &\leq \sum_{Y=0}^{(\epsilon/2)sd_1} \binom{sd_1}{Y} \left(\frac{s}{B_1}\right)^{Y} \left(\frac{s}{C'k}\right)^{\epsilon sd_1 d_2/2} \\ &\leq 2\left(\frac{s}{C'k}\right)^{\epsilon sd_1 d_2/2}. \end{aligned}$$

It immediately follows, similarly to upper-bounding S_1, that

$$S_2 \leq \exp\left\{-c_4 s \ln \frac{N}{k} \ln \frac{C'k}{s}\right\}, \tag{11}$$

where $c_4 > 0$ can be made arbitrarily large. Plugging Equations (10) and (11) into Equation (9) we see that Equation (8) holds. This completes the proof. □

D. PROOF OF LEMMA 2.12

PROOF. Fix a set S of size s. Let event $\mathcal{E}$ be that at least $(1-\epsilon/2)s$ elements in S are isolated in at least $(1-\zeta/2)d_1$ first-layer buckets. Similarly to Lemma 2.9, we know that

$$\Pr\{\mathcal{E}^c\} \leq \left(\frac{c's}{\zeta\epsilon B_1}\right)^{\zeta\epsilon sd_1} \leq e^{-cs\log\frac{N}{k}},$$

where c' is an absolute constant and $c > 0$ can be made arbitrarily large. In the above, we used that fact that since $B_1 = \Omega(k/(\zeta^\alpha \epsilon^{2\alpha}))$ it holds that

$$\ln \frac{\zeta\epsilon^2 B_1}{c_1 k} \geq \left(1 - \frac{1}{\alpha}\right) \ln \frac{B_1}{k}.$$

We condition on event $\mathcal{E}$. Among the $(1-\epsilon/2)s$ elements, we shall show that at least $(1-\epsilon)$ of them are isolated in at least $(1-\zeta)d_1d_2$ second-layer buckets. That means that there are a total of at least $\frac{\epsilon}{2}\frac{\eta}{2}sd_1d_2$ failed element-reptitions. But now, the probability of each collision is always bounded by s/B_2 even conditioned on previous outcomes, and we can proceed as in Lemma 2.9 to conclude that there are at least $\theta\zeta\epsilon sd_1d_2$ (for some absolute constant θ) with probability at most

$$\left(\frac{es}{\theta\zeta\epsilon B_2}\right)^{\theta\zeta\epsilon sd_1d_2} \leq e^{-c''s\log\frac{N}{k}},$$

as desired, where the constant $c'' > 0$ can be made arbitrarily large. □

ACKNOWLEDGMENT

We thank the anonymous reviewer for the valuable comments and suggestions that greatly contributed to improving this article.

REFERENCES

Radu Berinde, Anna C. Gilbert, Piotr Indyk, Howard Karloff, and Martin J. Strauss. 2008. Combining geometry and combinatorics: A unified approach to sparse signal recovery. In *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 798–805.

E. Candès, J. Romberg, and T. Tao. 2006. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Inf. Theor.* 52, 2 (2006), 489–509.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP)*. 693–703.

Mahdi Cheraghchi. 2013. Noise-resilient group testing: Limitations and constructions. *Discr. Appl. Math.* 161, 1–2 (2013), 81–95. DOI: http://dx.doi.org/10.1016/j.dam.2012.07.022

Mahdi Cheraghchi and Piotr Indyk. 2016. Nearly optimal deterministic algorithm for sparse walsh-hadamard transform. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 298–317.

Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. 2009. From coding theory to efficient pattern matching. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 778–784.

Albert Cohen, Wolfgang Dahmen, and Ronald Devore. 2009. Compressed sensing and best k-term approximation. *J. Am. Math. Soc* (2009), 211–231.

G. Cormode and S. Muthukrishnan. 2006. Combinatorial algorithms for Compressed Sensing. In *Proc. 40th Annual Conference on Information Sciences and Systems*. Princeton, Princeton, NJ.

D. L. Donoho. 2006. Compressed Sensing. *IEEE Trans. Info. Theory* 52, 4 (Apr. 2006), 1289–1306.

M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, K. F. Kelly, and R. G. Baraniuk. 2008. Single-pixel imaging via compressive sampling. *IEEE Sign. Process. Mag.* 25, 2 (2008), 83–91.

J. Friedman, J. Kahn, and E. Szemerédi. 1989. On the second eigenvalue of random regular graphs. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*. 587–598.

Anna Gilbert, Martin Strauss, Joel Tropp, and Roman Vershynin. 2006. Algorithmic linear dimension reduction in the ℓ_1 norm for sparse vectors. In *Proceedings of 44th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*.

Anna C. Gilbert, Yi Li, Ely Porat, and Martin J. Strauss. 2012. Approximate sparse recovery: Optimizing time and measurements. *SIAM J. Comput.* 41, 2 (2012), 436–453.

Anna C. Gilbert, Hung Q. Ngo, Ely Porat, Atri Rudra, and Martin J. Strauss. 2013. ℓ_2/ℓ_2-Foreach sparse recovery with low risk. In *Automata, Languages, and Programming*. Lecture Notes in Computer Science, Vol. 7965. Springer, Berlin, 461–472.

A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. 2007. One sketch for all: Fast Algorithms for Compressed Sensing. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, New York, NY, 237–246. DOI: http://dx.doi.org/10.1145/1250790.1250824

Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. 2009. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. *J. ACM* 56, 4, Article 20 (July 2009).

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

Piotr Indyk and Milan Ruzic. 2008. Near-optimal sparse recovery in the L_1 norm. *Found. Comput. Sci.* (2008), 199–207.

Michael Lustig, David Donoho, and John M. Pauly. 2007. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magn. Res. Med.* 58, 6 (2007), 1182–1195.

Jelani Nelson, Huy L. Nguyen, and David P. Woodruff. 2014. On deterministic sketching and streaming for sparse recovery and norm estimation. *Lin. Algebr. Appl.* 441 (2014), 152–167.

Farzad Parvaresh and Alexander Vardy. 2005. Correcting errors beyond the guruswami-sudan radius in polynomial time. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 285–294.

Ely Porat and Martin J. Strauss. 2012. Sublinear time, measurement-optimal, sparse recovery for all. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1215–1227.

Amnon Ta-Shma and David Zuckerman. 2004. Extractor codes. *IEEE Trans. Inf. Theor.* 50, 12 (2004), 3015–3025. DOI: http://dx.doi.org/10.1109/TIT.2004.838377

Eli Upfal. 1992. Tolerating linear number of faults in networks of bounded degree. In *Proceedings of the 11th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 83–89.

Received April 2015; revised September 2016; accepted November 2016