For-All Sparse Recovery in Near-Optimal Time*

Anna C. Gilbert[1,**], Yi Li[2,***], Ely Porat[3], and Martin J. Strauss[4,†]

[1] Department of Mathematics, University of Michigan
annacg@umich.edu
[2] Max-Planck Institute for Informatics
yli@mpi-inf.mpg.de
[3] Department of Computer Science, Bar-Ilan University
porately@cs.biu.ac.il
[4] Department of Mathematics and Department of EECS, University of Michigan
martinjs@umich.edu

Abstract. 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 is used for all signals $\mathbf{x}$. The best existing sublinear algorithm by Porat and Strauss (SODA'12) 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 paper, 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 $\alpha, \beta, \gamma > 0$. 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 algorithmic innovation is a novel encoding procedure that is reminiscent of network coding and that reflects the structure of the hashing stages.

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 [6,15]. 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 $\mathbf{y} = \Phi\mathbf{x}$, then we

* Omitted details and proofs can be found at arXiv:1402.1726 [cs.DS].

** Supported in part by DARPA/ONR N66001-08-1-2065.

*** Supported in part by NSF CCF 0743372 when the author was at the University of Michigan.

† Supported in part by NSF CCF 0743372 and DARPA/ONR N66001-08-1-2065.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 538–550, 2014.

Table 1. Summary of the best previous results and the result obtained in this paper. Some constant factors are omitted for clarity. "LP" denotes (at least) the runtime for a linear program of size at least N. The column "A/E" indicates whether the algorithm works in the forall (A) model or the foreach (E) model. The column "noise" indicates whether the algorithm tolerates noisy measurements. The constants c could be different in different occurrences. The lower bound on number of measurements in table above is, in fact, the best upper bound attained by super-linear algorithms.

Paper	A/E	Number of Measurements	Column sparsity/ Update time	Decode time	Approx. error	Noise
[2]	E	$k \log^c N$	$\log^c N$	$N \log^c N$	$\ell_2 \leq C\ell_2$	
[4]	E	$k \log^c N$	$\log^c N$	$k \log^c N$	$\ell_2 \leq C\ell_2$	
[8]	E	$\epsilon^{-1} k \log(N/k)$	$\log^c N$	$\epsilon^{-1} k \log^c N$	$\ell_2 \leq (1+\epsilon)\ell_2$	Y
[5,1]	A	$k \log(N/k)$	$k \log(N/k)$	LP	$\ell_2 \leq (C/\sqrt{k})\ell_1$	Y
[10]	A	$\epsilon^{-2} k \log^c N$	$\epsilon^{-2} k \log^c N$	$\epsilon^{-4} k^2 \log^c N$	$\ell_2 \leq (\epsilon/\sqrt{k})\ell_1$	Y
[9]	A	$k \log^c N$	$\log^c N$	$k \log^c N$	$\ell_1 \leq (C \log N)\ell_1$	Y
[14]	A	$\epsilon^{-2} k \log(N/k)$	$\epsilon^{-1} \log(N/k)$	$N \log(N/k)$	$\ell_1 \leq (1+\epsilon)\ell_1$	Y
[18] (any positive integer ℓ)	A	$\ell^c \epsilon^{-3} k \log(N/k)$	$\ell^c \epsilon^{-3} \log(N/k) \log k$	$\ell^c \epsilon^{-3} k (N/k)^{1/\ell}$	$\ell_1 \leq (1+\epsilon)\ell_1$	Y
This paper (any $\beta > 0$, restrictions on ϵ apply)	A	$\epsilon^{-2} k \log(N/k)$	$\epsilon^{-1} \log(N/k)$	$k^{1+\beta}(\epsilon^{-1} \log N)^c$	$\ell_1 \leq (1+\epsilon)\ell_1$	Y
Lower bound 'A'		$\epsilon^{-2} k \log(N/k)$	$\epsilon^{-1} \log(N/k)$	$\epsilon^{-2} k \log(N/k)$	$\ell_2 \leq (\epsilon/\sqrt{k})\ell_1$	Y

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 parameter k, and distortion parameter ϵ. Let $\mathbf{x}_{[k]}$ denote the best k-term approximation to $\mathbf{x}$, the "heavy hitters" of $\mathbf{x}$, *i.e.*, $\mathbf{x}$ 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 1 for an overview of all of problem variations. In this paper, we address the ℓ_1/ℓ_1-forall problem, formally defined below.

Definition 1. *An* (ℓ_1/ℓ_1) approximate sparse recovery system *consists of parameters* N, k, ϵ, *an* m*-by-*N measurement matrix $\boldsymbol{\Phi}$, *and a* decoding algorithm $\mathcal{D}$ *that satisfy the following property: for every vector* $\mathbf{x} \in \mathbb{R}^n$, *given* $\boldsymbol{\Phi}\mathbf{x}$, *the system approximates* $\mathbf{x}$ *by* $\widehat{\mathbf{x}} = \mathcal{D}(\boldsymbol{\Phi}\mathbf{x})$, *which satisfies* $\|\widehat{\mathbf{x}} - \mathbf{x}\|_1 \leq (1+\epsilon)\|\mathbf{x}_{[k]} - \mathbf{x}\|_1$.

What makes this problem challenging is that we must simultaneously keep the number of measurements small, ensure 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 $\log N$, it is easy to optimize the runtime. Similarly, if we severely restrict the distortion parameter ϵ, we may also increase the number of measurements by factors of ϵ. In many applications, all three quantities are important; i.e., 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 1. A number of parameter values are incommensurate: we can achieve better approximation guarantees (in the ℓ_2/ℓ_2 norm) but only in the for-each model; 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 paper is the mixed-norm (or ℓ_2/ℓ_1) for-all result. In this setting, the goal is to give Φ and $\mathcal{D}$, such that, for every $\mathbf{x}$, we have

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

It is known that if $(\Phi, \mathcal{D})$ solves the ℓ_2/ℓ_1 problem it also solves the ℓ_1/ℓ_1 problem [3]. 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 Φ is randomly chosen subject to $\mathcal{F}$, we have

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

The ℓ_2/ℓ_2 for-each problem with constant failure probability was solved in [8], where the authors gave an algorithm with constant-factor-optimal runtime and number of measurements. The failure probability was recently improved to exponentially small [11], 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 ℓ_1/ℓ_1 for-all setting was given in [18], though 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 did not achieve runtime polynomial in $k \log(N)/\epsilon$.
- The algorithm required a precomputed table of size $Nk^{0.2}$.
- The result was far from optimal in its dependence of the number of measurements on ϵ.

Our Results. In this work, we rectify the above limitations, assuming the (modest) restriction that $\epsilon < \log k / \log N$ and $k \leq \sqrt{N}$. We also make the measurement dependence on ϵ optimal. The best lower bound for the ℓ_1/ℓ_1 for-all problem is $\Omega(k/\epsilon^2 + (k/\epsilon)\log(\epsilon N/k))$ [16], 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 $\epsilon < (\log k / \log N)^{\gamma}$ for any constant $\gamma > 0$, which is suboptimal only by a logarithmic factor. When $k \leq \log^c N$ for some $c > 0$, the runtime is reduced to $O(k \operatorname{poly}(\log N, 1/\epsilon))$.

Theorem 1 (Main Theorem). *Let $\beta, \gamma > 0$. There is an approximate sparse recovery system that uses $m = O\big(\frac{k}{\epsilon}(\log \frac{N}{k})((\frac{\log N}{\log k})^{\gamma} + \frac{1}{\epsilon})\big)$ measurements and runs in time $O(k^{1+\beta} \operatorname{poly}(\log N, 1/\epsilon))$, provided that $N = \Omega(\max\{k^2, k/\epsilon^2\})$. When $\epsilon = O\big((\frac{\log k}{\log N})^{\gamma}\big)$, the number of measurements $m = O(k/\epsilon^2 \log(N/k))$.*

Overview of Techniques. Our overall approach builds on [18] and [11] with several critical innovations. In Fig. 1 is a framework which captures both the algorithm in [18] and the algorithm in this paper.

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 [18] this encoding is trivial, namely, $\mathbf{m}'_i = \mathbf{m}_i$; while in our work it is a more complicated procedure (see Fig. 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 $\log(N/k)/(\epsilon \log(B/k))$ repetitions. The index i in each repetition is now associated with a chunk of $\mathbf{m}'_i$. In [18], the aggregated buckets are hashed into (k/ϵ) buckets and there are $\log(B/k)/\epsilon$ repetitions. Thus, altogether, there are $O(\epsilon^{-3} k \log(N/k))$ measurements. In our work, there are only $\log(B/k)$ repetitions, saving a factor of $1/\epsilon$, so the total number of measurements is $O(\epsilon^{-2} k \log(N/k))$.

The *identification* portion of the recovery algorithm is shown in Fig. 2. To recover the identity of heavy hitters, the algorithm reads off the measurements and recovers the message chunk associated with each bucket. This message chunk 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 [18], from a look-up table and searched exhaustively. In our work, this is done by the decoding procedure illustrated in Fig. 4. We encode the "linking information" into the message chunks so that we can collect across the repetitions enough heavy buckets which 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 $\mathbf{m}_i$, whence 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 smaller (easier) sparsity parameter $k/2 < k$ and 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 Fig. 3 and 4 for a detailed illustration of these steps. For each message $\mathbf{m}$, the Parvaresh-Vardy code encodes it into a longer message $\mathbf{m}'$, which automatically exhibits a chunk structure, so that if a few number of the chunks are correct, the original $\mathbf{m}$ will be recovered. Suppose there are D chunks. 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 i-th chunk of $\mathbf{m}'$, append to it the information of the neighbours of the i-th vertex in G. Then we apply Reed-Solomon to protect the appended chunks.

To decode, we first recover the appended message chunks. 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 chunk (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 chunks) of a smaller expander graph of size $\Omega(D)$ in $\log D$ steps. This small fraction of good message chunks of $\mathbf{m}'$ will enable the PV code to recover the original message $\mathbf{m}$ successfully. Recall that d is a constant, 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; e.g., [14], and optimal in runtime up to a power.
- We conjecture that our algorithm can be extended from the 1-norm to the mixed norm guarantee and that the restriction on ϵ can be weakened or eliminated. Thus our algorithm may be a stepping stone to the final algorithm.
- Our work is not the first to consider list recovery. Indyk et al. introduces the idea in the context of combinatorial group testing [13]. The idea of list recovery is also used in [11], where the list decoding, 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.

Paper 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 weak system culminates in Section 4, where we show how to achieve good identification via message encoding and decoding. Then we build the overall algorithm on the weak system in Section 5 and close with a short discussion in Section 6.

2 Preliminaries

Our main algorithm will be built on regular graph expanders and unbalanced bipartite expanders. In Let n, m, d, ℓ be positive integers and ϵ, κ be positive reals. The following two definitions are adapted from [12].

Definition 2 (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 3 (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).*

Theorem 2 ([7]). *For all sufficiently large n and even d, there exists a d-regular expander G such that $|V(G)| = n$ and $\lambda(G) \leq C\sqrt{d}$ for some absolute constant $C > 0$, where $\lambda(G)$ denote the second largest eigenvalue, in absolute value, of G.*

Theorem 3 ([19]). *Let G be a δ-regular expander of n nodes such that $\lambda(G) \leq C\sqrt{\delta}$, where δ is a (sufficiently large) constant. There exist absolute constants $\alpha, \zeta > 0$ and $\kappa > 1$ 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 $(|V(G')|, n/2, \kappa)$ graph expander.*

The rest of the section concerns hashing. Informally, we say an (N, B, d) *(one layer) hashing scheme*[1] is to hash N elements into B buckets and repeat d times independently. Each instance of such a hashing scheme induces a d-left-regular bipartite graph with Bd right nodes. An (N, B_1, d_1, B_2, d_2) (two-layer) hashing scheme[2] is to hash N elements into B_1 buckets and repeat d_1 times (those buckets will be referred to as first-layer buckets) and in each of the d_1 repetitions, hash B_1 elements into B_2 buckets and repeat d_2 times (those buckets will be referred to as second-layer buckets). Each instance of such a hashing scheme induces a $d_1 d_2$-left-regular bipartite graph with $B_2 d_1 d_2$ right nodes.

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

Definition 4 (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 $|S| \leq \ell$, there exists $S' \subset S$ with $|S'| \geq (1 - \eta)|S|$ such that for all $x \in S'$ it holds that $|\Gamma(x) \setminus \Gamma(S \setminus \{x\})| \geq (1 - \zeta)d$.*

3 Weak System

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 $\mathbf{z}$ small norm (by normalization we may assume that $\|\mathbf{z}\|_1 \leq 3/2$). We call $\mathbf{y}$ the *head* and $\mathbf{z}$ the *tail*. We aim to recover the elements in $\mathbf{y}$. Introduced in [18], 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 [18] that when I contains the entire head, we can always recover a good fraction of the head. In fact, we only need I to contain a good fraction of the head instead of the entire head, with a slight modification of the original proof.

Definition 5 (Weak System). *A* Weak system *consists of parameters N, s, η, ζ, an m-by-N* measurement matrix $\boldsymbol{\Phi}$*, 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 $|\operatorname{supp}(\mathbf{y})| \leq s$ and $\|\mathbf{z}\|_1 \leq 3/2$, given the measurements $\boldsymbol{\Phi}\mathbf{x}$ and a subset*

[1] When N is clear from the context, we simply write (B, d) hashing scheme.
[2] When N is clear from the context, we simply write (B_1, d_1, B_2, d_2) hashing scheme.

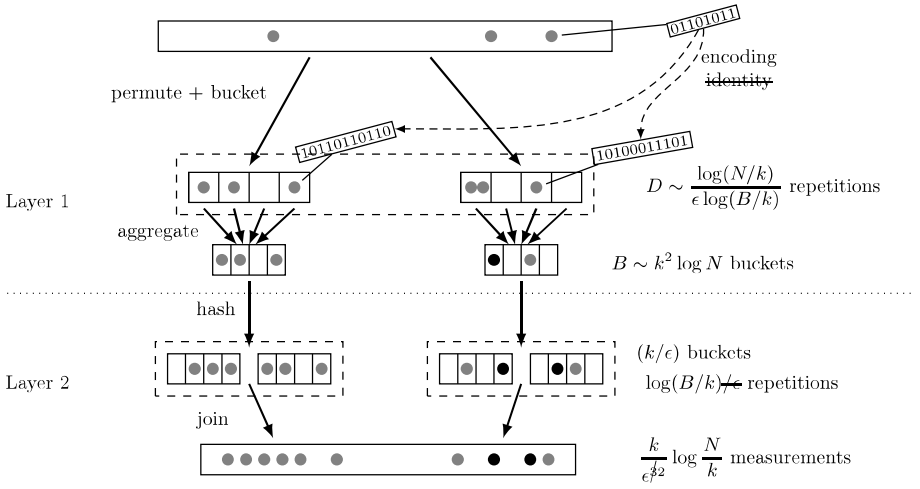

Fig. 1. Algorithm to generate the measurements. Darker spots indicate a bigger value of the bucket/measurement. Strikethroughs are used to show where our approach or our object sizes differ from [18].

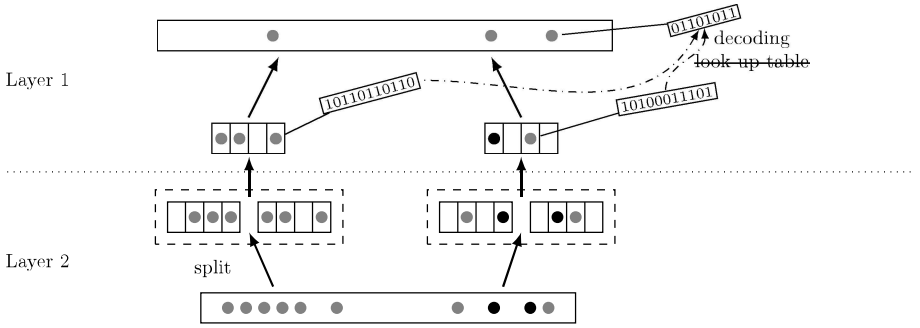

Fig. 2. Algorithm to recover from the measurements

$I \subseteq [N]$ such that $|I \cap \operatorname{supp}(\mathbf{y})| \geq (1 - \zeta/2)|\operatorname{supp}(\mathbf{y})|$, 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 $|\operatorname{supp}(\widehat{\mathbf{x}})| = O(s)$, $|\operatorname{supp}(\widehat{\mathbf{y}})| \leq \zeta s$, and $\|\widehat{\mathbf{z}}\|_1 \leq \|\mathbf{z}\|_1 + \eta$.

Theorem 4 (Weak). *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 $(O(k/\epsilon), \epsilon, \zeta)$-isolation property. With appropriate instantiations of constants, Algorithm 1 yields a correct Weak system running in time $O(\frac{|I|}{\eta\zeta^2} \log \frac{N}{s})$.*

To complete the construction of a Weak system, it remains to show that a bipartite expander as required by Theorem 4 exists. Indeed, it can be attained

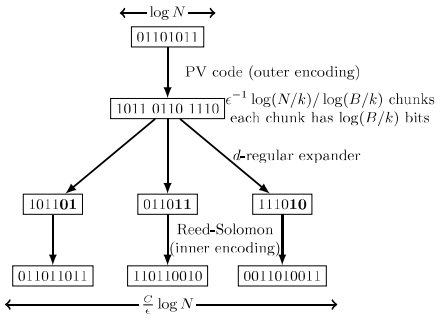

Fig. 3. Encoding scheme. The PV code exhibits a chunk structure. Suppose that there are D chunks. Choose a d-regular expander on D vertices as desired. For the i-th chunk of the PV code, append to it the information of the neighbours of the i-th vertex in the expander. Then apply Reed-Solomon to each appended message chunk. Recall that $k = O(\sqrt{N})$ so $\log(N/k) = \Theta(\log N)$.

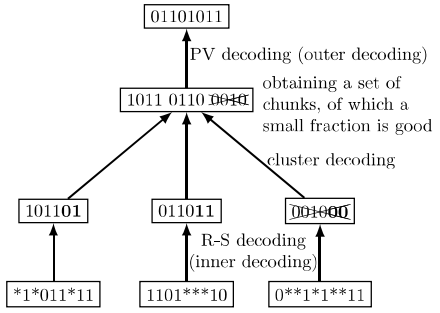

Fig. 4. Decoding scheme. The asterisks in the bottom layer indicates corrupted measurements (owing to collision or noise). The Reed-Solomon decoding either recovers the message chunk (with linking information) or produces a useless one (crossed out). Then the clustering procedure finds a set of chunks,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

Algorithm 1. Weak system

Input: N, s, $\boldsymbol{\Phi}$ (adjacency matrix of a d-left-regular expander G), $\boldsymbol{\Phi}\mathbf{x}$, and I
Output: $\widehat{\mathbf{x}}$
for $j \leftarrow 1$ *to* d **do**
  for *each* $i \in I$ **do**
    $\mathbf{x}_i^{(j)} \leftarrow \underset{u\in\Gamma(\{i\})}{\text{median}} \sum_{(u,v)\in E} \mathbf{x}_u$ /* each sum is an element of input $\boldsymbol{\Phi}\mathbf{x}$ */
for *each* $i \in I$ **do**
  $\mathbf{x}'_i \leftarrow \text{median}_{1\le j\le d}\, \mathbf{x}_i^{(j)}$
$\widehat{\mathbf{x}} \leftarrow$ top $O(s)$ elements of $\mathbf{x}'$
return $\widehat{\mathbf{x}}$

by both one-layer and two-layer hashing schemes, with appropriate parameters. We state the two-layer result below as our construction will use it.

Lemma 1. *Let* $\epsilon \in (0, \frac{1}{4})$, $\alpha > 1$, $k \ge 1$ *and* $N = \Omega\big(\max\{\frac{k}{\epsilon^2}, k^2\}\big)$. *Consider a two-layer* (B_1, d_1, B_2, d_2) *hashing scheme with* $B_1 = \Omega\big(\frac{k}{\zeta^\alpha \epsilon^{2\alpha}}\big)$, $d_1 = \Omega\big(\frac{\alpha}{\alpha-1} \cdot \frac{1}{\zeta\epsilon}\frac{\log(N/k)}{\log(B/k)}\big)$, $B_2 = \Omega\big(\frac{k}{\zeta\epsilon}\big)$ *and* $d_2 = \Omega\big(\frac{1}{\zeta}\log\frac{B_1}{k}\big)$. *With probability* $\ge 1 - N^{-\Omega(1)}$, *such a two-layer hashing scheme gives an* $(B_2 d_1 d_2, d_1 d_2, 4k, \epsilon)$ *bipartite expander with the* $(O(k/\epsilon), \epsilon, \zeta)$*-isolation property.*

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.

Algorithm 2. Encoding/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 chunks
    Outer encoding: Encode the chunks with cluster info (from a regular expander graph)
    and against errors, getting {m_{i,j}}_{j=1}^{d_1}
for j = 1 to d_1 do
    Inner encoding: Encode m_{i,j}, for i ∈ [N]
// Decoding with (B_1, d_1, B_2, d_2) hashing scheme
for j = 1 to d_1 do
    // length B_1, (B_2 d_2)-measurement Sparse Recovery Channel
    Inner decoding: Recover m̂_j in the Weak List sense
    Record Side Info: Tag each element of m̂_j with j
Outer decoding: From m̂ = ⋃_j m̂_j's, find chunk clusters and correct errors; produce I
```

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. To do this, we encode the index i in such a way that if we recover the buckets containing i in enough repetitions we shall be able to reconstruct i. To that end, we introduce the following model of weak list recovery in the *sparse recovery channel*.

Definition 6. *The* (m, N, s) Sparse Recovery Channel *takes an* m*-by-*N *matrix* $\boldsymbol{\Phi}$ *as input, chooses a signal* $\mathbf{x}$ *with decomposition* $\mathbf{x} = \mathbf{y} + \mathbf{z}$ *with* $|\operatorname{supp}(\mathbf{y})| \le s$ *and* $\|\mathbf{z}\|_1 \le O(1)$*, and outputs* $\boldsymbol{\Phi}\mathbf{x}$.

Note that $\mathbf{x}$ may depend on $\boldsymbol{\Phi}$. Also note that *any* signal may be chosen by the channel and normalized so that $\|\mathbf{z}\|_1 \le 3/2$. It will be convenient to assign the normalization at this point to match the Weak system (Defintion 5). Next, we define the *Weak Recovery Criterion* appropriate for this channel. See Fig. 5.

Definition 7 (Weak List Recovery Criterion). *Fix parameters* m, N, s, ϵ. *Let* $\mathbf{m}$ *be a vector of* β*-bit messages and* $i \in [N]$. *Suppose* $\widehat{\mathbf{m}}$ *is a list of possible index-message pairs. We say that* $\widehat{\mathbf{m}}$ *is* correct in the List Weak sense *if, for at least* $|\operatorname{supp}(\mathbf{y})| - s/8$ *indices* i *in* $\operatorname{supp}(\mathbf{y})$*, we have* $(i, \mathbf{m}_i) \in \widehat{\mathbf{m}}$.

The encoding/decoding scheme is given in Algorithm 2. We break each message $\mathbf{m}_i$ (which could be much longer than $\log N$ bits) associated with position i into d_1 chunks, $\mathbf{m}_{i,1}, \ldots, \mathbf{m}_{i,d_1}$. Now in the j-th 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 bucket noise is mild, 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.

The following lemma is a simple case to illustrate our idea of encoding, in which we show how to code $\beta = \log(B/k)$ bits in the length-B Sparse Recovery

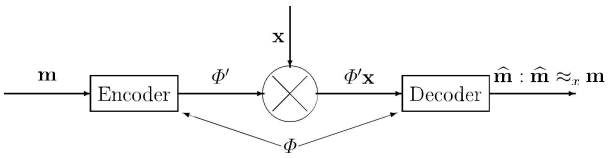

Fig. 5. Sparse recovery channel. The encoder and decoder agree on some matrix $\boldsymbol{\Phi}$. The encoder takes messages $\mathbf{m}$ and produces a measurement matrix $\boldsymbol{\Phi}'$ based on $\mathbf{m}$ and $\boldsymbol{\Phi}$. The channel is fed with $\boldsymbol{\Phi}'$ and $\mathbf{x}$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

Channel and how to recover the messages associated with $\Omega(k)$ heavy hitters in the length B signal in time approximately B.

Lemma 2. *Let $B = \Omega(k)$ and $\beta = O(\log(B/k))$. There is a coding scheme for the length-B m-measurememt Sparse Recovery Channel for $m = O(k/\epsilon \log(B/k))$ in the weak list recovery sense in which decoding runs in time $O(B \log^3(B/k))$.*

The rest of this section is devoted to an expander-based coding scheme for the most difficult part of identification, that is, to match $\mathbf{m}_{i,j}$ with $\mathbf{m}_{i,j'}$ $(j \neq j')$ in order to find enough fraction of $\mathbf{m}_i$ in the end. We resolve this by embedding "linking information" in $\mathbf{m}_{i,j}$.

Parameters. We assume that the constants $\beta, \gamma > 0$ are fixed and the parameters B_1, d_1, B_2, d_2 are as in Lemma 1 such that $B_1 = \Omega\big((\frac{k}{\epsilon^2})^{1+\beta} \log \frac{N}{k}\big)$. Let G be a graph of d_1 nodes with constant degree δ that satisfies Theorem 2, and α, ζ, κ be constants provided by Theorem 3 when applied to G. Without loss generality we can assume that $\alpha \leq 1/2$. Let $c \leq m$ be positive integer constants, h a positive integer and $\epsilon = O\big(\big(\frac{\alpha}{m}\big)^{\frac{m}{m-c}} \big(\frac{\log(B_1/k)}{\log(N/k)}\big)^{\gamma}\big)$. 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) $(\alpha d_1 - (h-1)m \log_{B_1} N) \cdot h^m > d_1^c$ and (d) $c \geq \log \delta / \log \kappa$.

Encoding. We shall use Reed-Solomon for inner encoding. Next, we define our outer coding, which uses the Parvaresh-Vardy code [17]. Take N disconnected copies of G and call the union G_N, where each node is indexed by a pair $(i, r) \in [N] \times [d_1]$. 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_i over $\mathbb{F}$ with degree $\log_{B_1} N - 1$. For each $(i, r) \in G_N$, associate with it an element $p(i, r) \in \mathbb{F}^{m+1}$ defined by

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

where $x_{i,r} \in \mathbb{F}$ are distinct for all r. This is possible because of Property (a).

Attach to a node (i, r) a message $\mathbf{m}_{i,r}$ containing $p(i, r)$ as well as $H(i, v_1(r))$,..., $H(i, v_\delta(r))$, 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 j-th 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 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. As mentioned in the introduction, we wish to collect enough $p(i,r)$ for different values of r and the same i. To this end, we do clustering as follows.

Suppose that there are k heavy hitters at position $i_1, \ldots, i_k$. Let $\tilde{G}$ 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 an ideal situation where the messages are recovered correctly for each heavy hitter i in all d_1 repetitions (which implies no collisions and small bucket noise). Each message has the form $(p(i,r), h_1, \ldots, h_\delta)$, where $h_j = H(i, v_j(r))$ $(j \in [\delta])$. Add an arc $(i,r) \to (h_j, v_j(r))$ for each $j \in [\delta]$.

Since the messages are recovered correctly, the graph $\tilde{G}$ will contain several disjoint copies of the expander graph G, say $G_{i_1}, \ldots, G_{i_k}$. There will be arcs incoming to G_{i_j} from nodes not in any G_{i_j}, but there will be 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. In this case, the columns $i_1, \ldots, i_k$ are exact copies of the expander graph G.

The heavy hitters may not, however, be recovered in some repetitions and the messages could be seriously corrupted. Adding arcs introduces two kinds of errors: (i) We lose a node in G_{i_j} 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 $G_{i_{j'}}$ $(j \neq j')$, owing to errorous message. We know that for a heavy hitter i, only a few messages $\{\mathbf{m}_{i,r}\}_r$ are ruined and the i-th column of G_N will contain a large connected subgraph G' of G, by Theorem 3. Hence, if we start a breadth-first search at an appropriate node with depth $c \log_\delta d_1$, the whole G' will be visited. In other words, we shall obtain a large set of $\{p(i,r)\}$, a small number of which will be associated with the same i, but it is sufficient to extract f_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 identifying the "appropriate nodes", we perform a breadth-first search at every node in $\tilde{G}$.

Guarantee. We show that the system described above meets the aforementioned guarantee, using Property (b)–(d).

Theorem 5. *Let $\beta, \gamma > 0$. The encoding and decoding strategy above are correct in the sense of weak list recovery, against the channel described in that section. It uses $O(\epsilon^{-2} s \log(N/s))$ measurements and runs in time $O(s^{1+\beta} \operatorname{poly}(\log N, 1/\epsilon))$, provided that $N = \Omega(\max\{s^2, s/\epsilon^2\})$ and $\epsilon = O\big((\log s/\log N)^\gamma\big)$.*

5 Overall Algorithm

The construction of an approximate recovery system from a weak system is similar to existing works [8,18]. We use Theorem 5 for identification and Theorem 4

for estimation. Below we simply restate our main theorem result from Theorem 1 in a slightly different form.

Theorem 6. *Let $\beta, \gamma > 0$. There is an approximate recovery system that uses $O(\epsilon^{-2} k \log(N/k))$ measurements and runs in time $O(k^{1+\beta} \operatorname{poly}(\log N, 1/\epsilon))$, provided that $N = \Omega(\max\{k^2, k/\epsilon^2\})$ and $\epsilon = O((\log k/\log N)^{\gamma})$.*

We remark that (a) the constants in big O-notations and the power in $\operatorname{poly}(\log N, 1/\epsilon)$ depend on β and γ; (b) the constraint that $k = O(\sqrt{N})$ could be relaxed to $k = O(N^{1-\alpha})$ for any $\alpha > 0$, the hidden constants will depend on α; (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 $k \leq \operatorname{poly}(\log N)$, it suffices to choose $B_1 = \Theta(k \log(N/k)/\epsilon^{2(1+\beta)})$, leading to runtime $O(k \operatorname{poly}(\log N, 1/\epsilon))$; (d) for large ϵ we can take $d_1 = (\log(N/k)/\log(B_1/k))^{1+\alpha}$ for an arbitrary $\alpha > 0$, which gives an algorithm which uses more measurements $O(k \log^{1+\alpha}(N/k)/\epsilon^2)$ but suboptimal by only a logarithmic factor.

6 Discussions

At the core part of this paper lies the following list recovery problem: Suppose that there are $d_1 = \frac{1}{\epsilon} \frac{\log(N/k)}{\log(B/k)}$ lists $L_1, \ldots, L_{d_1}$ with $|L_i| = O(k/\epsilon)$ for all $i \in [d_1]$, 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 i's in $[d_1]$. We used an expander structure to reduce the problem to kd_1/ϵ subproblems, each of which has a smaller number of nodes. 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 $O(k \operatorname{poly}(\log N, 1/\epsilon))$, 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. We also remark that the Parvaresh-Vardy code in the outer coding and the Reed-Solomon code in the inner coding could be replaced with other codes of similar parameters, or better parameters, which would lead to an improvement of the algorithm.

References

1. Candès, E., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE T. Info. Theory 52(2), 489–509 (2006)
2. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 693–703. Springer, Heidelberg (2002)
3. Cohen, A., Dahmen, W., Devore, R.: Compressed sensing and best k-term approximation. J. Amer. Math. Soc., 211–231 (2009)

4. Cormode, G., Muthukrishnan, S.: Combinatorial algorithms for compressed sensing. In: SIROCCO, pp. 280–294 (2006)
5. Donoho, D.L.: Compressed sensing. IEEE T. Info. Theory 52(4), 1289–1306 (2006)
6. Duarte, M.F., Davenport, M.A., Takhar, D., Laska, J.N., Kelly, K.F., Baraniuk, R.G.: Single-pixel imaging via compressive sampling. IEEE Signal Processing Magazine 25(2), 83–91 (2008)
7. Friedman, J., Kahn, J., Szemerédi, E.: On the second eigenvalue of random regular graphs. In: STOC, pp. 587–598 (1989)
8. Gilbert, A., Li, Y., Porat, E., Strauss, M.: Approximate sparse recovery: Optimizing time and measurements. SIAM J. Comput. 41(2), 436–453 (2012)
9. Gilbert, A., Strauss, M., Tropp, J., Vershynin, R.: Algorithmic linear dimension reduction in the ℓ_1 norm for sparse vectors. In: Allerton (2006)
10. Gilbert, A., Strauss, M., Tropp, J., Vershynin, R.: One sketch for all: fast algorithms for compressed sensing. In: ACM STOC, pp. 237–246 (2007)
11. Gilbert, A.C., Ngo, H.Q., Porat, E., Rudra, A., Strauss, M.J.: ℓ_2/ℓ_2-foreach sparse recovery with low risk. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 461–472. Springer, Heidelberg (2013)
12. Guruswami, V., Umans, C., Vadhan, S.: Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. J. ACM 56(4), 20:1–20:34
13. Indyk, P., Ngo, H.Q., Rudra, A.: Efficiently decodable non-adaptive group testing. In: SODA, pp. 1126–1142 (2010)
14. Indyk, P., Ruzic, M.: Near-optimal sparse recovery in the ℓ_1 norm. In: FOCS, pp. 199–207 (2008)
15. Lustig, M., Donoho, D., Pauly, J.M.: Sparse MRI: The application of compressed sensing for rapid MR imaging. Magn. Reson. Med. 58(6), 1182–1195 (2007)
16. Nelson, J., Nguy˜ên, H.L., Woodruff, D.P.: On deterministic sketching and streaming for sparse recovery and norm estimation. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX/RANDOM 2012. LNCS, vol. 7408, pp. 627–638. Springer, Heidelberg (2012)
17. Parvaresh, F., Vardy, A.: Correcting errors beyond the guruswami-sudan radius in polynomial time. In: FOCS, pp. 285–294 (2005)
18. Porat, E., Strauss, M.J.: Sublinear time, measurement-optimal, sparse recovery for all. In: SODA, pp. 1215–1227 (2012)
19. Upfal, E.: Tolerating linear number of faults in networks of bounded degree. In: PODC, pp. 83–89 (1992)