

Nearly Optimal Deterministic Algorithm for Sparse Walsh-Hadamard Transform

MAHDI CHERAGHCHI*
University of California
Berkeley, CA 94720

PIOTR INDYK†
MIT
Cambridge, MA 02139

Abstract

For every fixed constant $\alpha > 0$, we design an algorithm for computing the k -sparse Walsh-Hadamard transform of an N -dimensional vector $x \in \mathbb{R}^N$ in time $k^{1+\alpha}(\log N)^{O(1)}$. Specifically, the algorithm is given query access to x and computes a k -sparse $\tilde{x} \in \mathbb{R}^N$ satisfying $\|\tilde{x} - \hat{x}\|_1 \leq c\|\hat{x} - H_k(\hat{x})\|_1$, for an absolute constant $c > 0$, where \hat{x} is the transform of x and $H_k(\hat{x})$ is its best k -sparse approximation. Our algorithm is fully deterministic and only uses non-adaptive queries to x (i.e., all queries are determined and performed in parallel when the algorithm starts).

An important technical tool that we use is a construction of nearly optimal and linear lossless condensers which is a careful instantiation of the GUV condenser (Guruswami, Umans, Vadhan, JACM 2009). Moreover, we design a deterministic and non-adaptive ℓ_1/ℓ_1 compressed sensing scheme based on general lossless condensers that is equipped with a fast reconstruction algorithm running in time $k^{1+\alpha}(\log N)^{O(1)}$ (for the GUV-based condenser) and is of independent interest. Our scheme significantly simplifies and improves an earlier expander-based construction due to Berinde, Gilbert, Indyk, Karloff, Strauss (Allerton 2008).

Our methods use linear lossless condensers in a black box fashion; therefore, any future improvement on explicit constructions of such condensers would immediately translate to improved parameters in our framework (potentially leading to $k(\log N)^{O(1)}$ reconstruction time with a reduced exponent in the poly-logarithmic factor, and eliminating the extra parameter α).

Finally, by allowing the algorithm to use randomness, while still using non-adaptive queries, the running time of the algorithm can be improved to $\tilde{O}(k \log^3 N)$.

*Email: {cheraghchi@berkeley.edu}. Work supported in part by a Qualcomm fellowship at Simons Institute for the Theory of Computing at UC Berkeley, and a Swiss National Science Foundation research grant PA00P2-141980. Part of work was done while the author was with MIT Computer Science and Artificial Intelligence Laboratory.

†Email: {indyk@mit.edu}.

Contents

1	Introduction	3
1.1	Our result	4
1.2	Techniques	5
2	Preliminaries	7
3	Obtaining nearly optimal sample complexity	9
3.1	Proof of Lemma 9	10
4	Obtaining nearly optimal reconstruction time	12
4.1	Augmentation of the sensing matrix	12
4.2	The sparse recovery algorithm	14
4.3	Analysis of the running time	16
4.4	Proof of Theorem 19	17
4.5	Proof of Lemma 21	21
5	Speeding up the algorithm using randomness	25
5.1	Proof of Theorem 31	29
5.1.1	Correctness analysis of the randomized sparse recovery algorithm	29
5.1.2	Analysis of the running time of the randomized sparse recovery algorithm	32
A	Proof of Theorem 12 (construction of the lossless condenser)	35
B	The Leftover Hash Lemma	37

1 Introduction

The Discrete Walsh-Hadamard transform (henceforth the Hadamard Transform or DHT) of a vector $x \in \mathbb{R}^N$, where $N = 2^n$, is a vector $\hat{x} \in \mathbb{R}^N$ defined as follows:

$$\hat{x}(i) = \frac{1}{\sqrt{N}} \sum_{j \in \mathbb{F}_2^n} (-1)^{\langle i, j \rangle} x(j) \quad (1)$$

where the coordinate positions are indexed by the elements of \mathbb{F}_2^n , $x(i)$ denoting the entry at position $i \in \mathbb{F}_2^n$ and the inner product $\langle i, j \rangle$ is over \mathbb{F}_2 . Equivalently, the Hadamard transform is a variation of the Discrete Fourier transform (DFT) defined over the hypercube \mathbb{F}_2^n . We use the notation $\hat{x} = \text{DHT}(x)$.

The standard divide and conquer approach of Fast Fourier Transform (FFT) can be applied to the Hadamard transform as well to compute DHT in time $O(N \log N)$. In many applications, however, most of the Fourier coefficients of a signal are small or equal to zero, i.e., the output of the DFT is (approximately) sparse. In such scenarios one can hope to design an algorithm with a running time that is *sub-linear* in the signal length N . Such algorithms would significantly improve the performance of systems that rely on processing of sparse signals.

The goal of designing efficient DFT and DHT algorithms for (approximately) sparse signals has been a subject of a large body of research, starting with the celebrated Goldreich-Levin theorem [7] in complexity theory¹. The last decade has witnessed the development of several highly efficient sub-linear time sparse Fourier transform algorithms. These recent algorithms have mostly focused on the Discrete Fourier transform (DFT) over the cyclic group \mathbb{Z}_N (and techniques that only apply to this group), whereas some (for example, [16]) have focused on the Hadamard transform. In terms of the running time, the best bounds to date were obtained in [9] which showed that a k -sparse approximation of the DFT transform can be computed in time $O(k(\log N)^2)$, or even in $O(k \log N)$ time if the spectrum of the signal has at most k non-zero coefficients. These developments as well as some of their applications have been summarized in two surveys [6] and [5].

While most of the aforementioned algorithms are randomized, from both theoretical and practical viewpoints it is desirable to design *deterministic* algorithms for the problem. Although such algorithms have been a subject of several works, including [1, 13, 12], there is a considerable efficiency gap between the deterministic sparse Fourier Transform algorithms and the randomized ones. Specifically, the best known deterministic algorithm, given in [12], finds a k -sparse approximation of the DFT transform of a signal in time $O(k^2(\log N)^{O(1)})$; i.e., its running time is *quadratic* in the signal sparsity. Designing a deterministic algorithm with reduced run time dependence on the signal sparsity has been recognized as a challenging open problem in the area (e.g., see Question 2 in [11]).

¹ This result is also known in the coding theory community as a list decoding algorithm for the Hadamard code, and crucially used in computational learning as a part of the Kushilevitz-Mansour Algorithm for learning low-degree Boolean functions [15].

1.1 Our result

In this paper we make a considerable progress on this question, by designing a deterministic algorithm for DHT that runs in time $O(k^{1+\alpha}(\log N)^{O(1)})$. Since our main interest is optimizing the exponent of k in the running time of the DHT algorithm, the reader may think of a parameter regime where the sparsity parameter k is not too insignificant compared to the dimension N (e.g., we would like to have $k \geq (\log N)^{\omega(1)}$, say $k \approx N^{\Theta(1)}$) so that reducing the exponent of k at cost of incurring additional poly-logarithmic factors in N would be feasible².

To describe the result formally, we will consider a formulation of the problem when the algorithm is given a query access to \hat{x} and the goal is to approximate the largest k terms of x using a deterministic sub-linear time algorithm³. More precisely, given an integer parameter k and query access to \hat{x} , we wish to compute a vector $\tilde{x} \in \mathbb{F}_2^N$ such that for some absolute constant $c > 0$,

$$\|\tilde{x} - x\|_1 \leq c \cdot \|H_k(x) - x\|_1, \quad (2)$$

where we use $H_k(x)$ to denote the approximation of x to the k largest magnitude coordinates; i.e., $H_k(x) \in \mathbb{R}^N$ is only supported on the k largest (in absolute value) coefficients of x and is equal to x in those positions. Note that if the input signal x has at most k non-zero coefficients, then $H_k(x) = x$ and therefore the recovery is exact, i.e., $\tilde{x} = x$. The goal formulated in (2) is the so-called ℓ_1/ℓ_1 recovery in the sparse recovery literature. In general, one may think of ℓ_p/ℓ_q recovery where the norm on the left hand side (resp., right hand side) of 2 is ℓ_p (resp., ℓ_q), such as ℓ_2/ℓ_1 or ℓ_2/ℓ_2 . However, in this work we only address the ℓ_1/ℓ_1 model as formulated in (2) (for a survey of different objectives and a comparison between them, see [4]).

The following statement formally captures our main result.

Theorem 1. *For every fixed constant $\alpha > 0$, there is a deterministic algorithm as follows. Let $N = 2^n$ and $k \leq N$ be positive integers. Then, given (non-adaptive) query access to any $\hat{x} \in \mathbb{R}^N$ where each coefficient of \hat{x} is $n^{O(1)}$ bits long, the algorithm runs in time $k^{1+\alpha}n^{O(1)}$ and outputs $\tilde{x} \in \mathbb{R}^N$ that satisfies (2) (where $\hat{x} = \text{DHT}(x)$) for some absolute constant $c > 0$.*

Remark 2. The parameter α in the above result is arbitrary as long as it is an absolute positive constant, for example one may fix $\alpha = .1$ throughout the paper. We remark that this parameter appears not because of our general techniques but solely as an artifact of a particular state-of-the-art family of unbalanced expander graphs (due to Guruswami, Umans, and Vadhan [8]) that we use as a part of the algorithm (as further explained below in the techniques section). Since we use such expander graphs as a black box, any future progress on construction of unbalanced expander graphs would immediately improve the running time achieved by Theorem 1, potentially leading to

²For this reason, and in favor of the clarity and modularity of presentation, for the most part we do not attempt to optimize the exact constant in the exponent of the $(\log N)^{O(1)}$ factor.

³Since the Hadamard transform is its own inverse, we can interchange the roles of x and \hat{x} , so the same algorithm can be used to approximate the largest k terms of \hat{x} given query access to x .

a nearly optimal time of $kn^{O(1)}$, with linear dependence on the sparsity parameter k which would be the best to hope for.

In the running time $k^{1+\alpha}n^{O(1)}$ reported by Theorem 1, the $O(1)$ in the exponent of n hides a factor depending on $1/\alpha$; i.e., the running time can be more precisely be written as $k^{1+\alpha}n^{2/\alpha+O(1)}$. However, since α is taken to be an absolute constant, this in turn asymptotically simplifies to $k^{1+\alpha}n^{O(1)}$. Since our main focus in this work is optimizing the exponent of k (and regard the sparsity k to not be too small compared to N , say $k \approx N^{\Theta(1)}$), we have not attempted to optimize the exponent of $\log N$ in the running time. However, as we will see in Section 5.1, if one is willing to use randomness in the algorithm, the running time can be significantly improved (eliminating the need for the parameter α) using a currently existing family of explicit expander graphs (based on the Left-over Hash Lemma). \square

As discussed in Remark 2 above, our algorithm employs state of the art constructions of explicit lossless expander graphs that to this date remain sub-optimal, resulting in a rather large exponent in the $\log N$ factor of the asymptotic running time estimate. Even though the main focus of this article is fully deterministic algorithms for fast recovery of the Discrete Hadamard Transform, we further observe that the same algorithm that we develop can be adapted to run substantially faster using randomness and sub-optimal lossless expander graphs such as the family of expanders obtained from the Leftover Hash Lemma. As a result, we obtain the following improvement over the deterministic version of our algorithm.

Theorem 3. *There is a randomized algorithm that, given integers k, n (where $k \leq n$), and (non-adaptive) query access to any $\hat{x} \in \mathbb{R}^N$ (where $N := 2^n$ and each coefficient of \hat{x} is $O(n)$ bits long), outputs $\tilde{x} \in \mathbb{R}^N$ that, with probability at least $1 - o(1)$ over the internal random coin tosses of the algorithm, satisfies (2) for some absolute constant $c > 0$ and $\hat{x} = \text{DHT}(x)$. Moreover, the algorithm performs a worst-case $O(kn^3(\log k)(\log n)) = \tilde{O}(k(\log N)^3)$ arithmetic operations.*

1.2 Techniques

Most of the recent sparse Fourier transform algorithms (both randomized and deterministic) are based on a form of “binning”. At a high level, sparse Fourier algorithms work by mapping (binning) the coefficients into a small number of bins. Since the signal is sparse, each bin is likely to have only one large coefficient, which can then be located (to find its position) and estimated (to find its value). The key requirement is that the binning process needs to be performed using few samples of \hat{x} , to minimize the running time. Furthermore, since the estimation step typically introduces some error, the process is repeated several times, either in parallel (where the results of independent trials are aggregated at the end) or iteratively (where the identified coefficients are eliminated before proceeding to the next step).

As described above, the best previous deterministic algorithm for the sparse Fourier Transform (over the cyclic group \mathbb{Z}_N), given in [12], runs in time $k^2 \cdot (\log N)^{O(1)}$. The algorithm satisfies the

guarantee⁴ in (2). The algorithm follows the aforementioned approach, where binning is implemented by *aliasing*; i.e., by computing a signal y such that $y_j = \sum_{i: i \bmod p=j} x_i$, where p denotes the number of bins. To ensure that the coefficients are isolated by the mapping, this process is repeated in parallel for several values of $p = p_1, p_2, \dots, p_t$. Each p_i is greater than k to ensure that there are more bins than elements. Furthermore, the number of different aliasing patterns t must be greater than k as well, as otherwise a fixed coefficient could always collide with one of the other k coefficients. As a result, this approach requires more than k^2 bins, which results in quadratic running time. One can reduce the number of bins by resorting to randomization: The algorithm can select only some of the p_i 's uniformly at random and still ensure that a fixed coefficient does not collide with any other coefficient with constant probability. In the deterministic case, however, it is easy to see that one needs to use $\Omega(k)$ mappings to isolate each coefficient, and thus the analysis of the algorithm in [12] is essentially tight.

In order to reduce the running time, we need to reduce the total number of mappings. To this end we relax the requirements imposed on the mappings. Specifically, we will require that the union of all coefficients-to-bins mappings forms a good *expander graph* (see section 2 for the formal definition). Expansion is a natural property to require in this context, as it is known that there exist expanders that are induced by only $(\log N)^{O(1)}$ mappings but that nevertheless lead to near-optimal sparse recovery schemes [2]. The difficulty, however, is that for our purpose we need to simulate those mappings on coefficients of the signal x , even though we can only access the spectrum \hat{x} of x . Thus, unlike in [2], in our case we cannot use arbitrary “black box” expanders induced by arbitrary mappings. Fortunately, there is a class of mappings that are easy to implement in our context, namely the class of *linear* mappings.

In this paper, we first show that an observation by one of the authors (as reported in [3]) implies that there exist explicit expanders that are induced by a small number of linear mappings. From this we conclude that there exists an algorithm that makes only $k^{1+\alpha}(\log N)^{O(1)}$ queries to \hat{x} and finds a solution satisfying (2). However, the expander construction alone does not yield an *efficient* algorithm. To obtain such an algorithm, we augment the expander construction with an extra set of queries that enables us to quickly identify the large coefficients of x . The recovery procedure that uses those queries is iterative, and the general approach is similar to the algorithm given in Appendix A of [2]. However, our procedure and the analysis are considerably simpler (thanks to the fact that we only use the so-called Restricted Isometry Property (RIP) for the ℓ_1 norm instead of ℓ_p for $p > 1$). Moreover, our particular construction is immediately extendable for use in the Hadamard transform problem (due to the linearity properties).

The rest of the article is organized as follows. Section 2 discusses notation and the straightforward observation that the sparse DHT problem reduces to compressed sensing with query access

⁴Technically, the guarantee proven in [12] is somewhat different, namely it shows that $\|\tilde{x} - x\|_2 \leq \|H_k(x) - x\|_2 + \frac{c}{\sqrt{k}} \cdot \|H_k(x) - x\|_1$. However, the guarantee of (2) can be shown as well [Mark Iwen, personal communication]. In general, the guarantee of (2) is easier to show than the guarantee in [12].

to the Discrete Hadamard Transform of the underlying sparse signal. Also the notion of Restricted Isometry Property, lossless condensers, and unbalanced expander graphs are introduced in this section. Section 3 focuses on the sample complexity; i.e., the amount of (non-adaptive) queries that the compressed sensing algorithm (obtained by the above reduction) makes in order to reconstruct the underlying sparse signal. Section 4 adds to the results of the preceding section and describes our main (deterministic and sublinear time) algorithm to efficiently reconstruct the sparse signal from the obtained measurements. Finally Section 5 observes that the performance of the algorithm can be improved when allowed to use randomness. Although the main focus of this article is on deterministic algorithms, the improvement using randomness comes as an added bonus that we believe is worthwhile to mention.

2 Preliminaries

Notation. Let $N := 2^n$ and $x \in \mathbb{R}^N$. We index the entries of x by elements of \mathbb{F}_2^n and refer to $x(i)$, for $i \in \mathbb{F}_2^n$, as the entry of x at the i th coordinate. The notation $\text{supp}(x)$ is used for *support* of x ; i.e., the set of nonzero coordinate positions of x . A vector x is called k -sparse if $|\text{supp}(x)| \leq k$. For a set $S \subseteq [N]$ we denote by x_S the N -dimensional vector that agrees with x on coordinates picked by S and is zeros elsewhere. We thus have $x_{\bar{S}} = x - x_S$. All logarithms in this work are to the base 2.

Equivalent formulation by interchanging the roles of x and \hat{x}

Recall that in the original sparse Hadamard transform problem, the algorithm is given query access to a vector $x \in \mathbb{R}^N$ and the goal is to compute a k -sparse \tilde{x} that approximates $\hat{x} = \text{DHT}(x)$. That is,

$$\|\tilde{x} - \hat{x}\|_1 \leq c \cdot \|\hat{x} - H_k(\hat{x})\|_1$$

for an absolute constant $c > 0$. However, since the Hadamard transform is its own inverse; i.e., $\text{DHT}(\hat{x}) = x$, we can interchange the roles of x and \hat{x} . That is, the original sparse Hadamard transform problem is equivalent to the problem of having query access to the Hadamard transform of x (i.e., \hat{x}) and computing a k -sparse approximation of x satisfying (2). *Henceforth throughout the paper, we consider this equivalent formulation which is more convenient for establishing the connection with sparse recovery problems.*

Approximation guarantees and the Restricted Isometry property: We note that the equation in (2) is similar to the ℓ_1/ℓ_1 recovery studied in compressed sensing. In fact the sparse Hadamard transform problem as formulated above is the same as ℓ_1/ℓ_1 when the measurements are restricted to the set of linear forms extracting Hadamard coefficients. Thus our goal in this work is to present a non-adaptive sub-linear time algorithm that achieves the above requirements for

all vectors x and in a deterministic and efficient fashion. It is known that the so-called Restricted Isometry Property for the ℓ_1 norm (RIP-1) characterizes the combinatorial property needed to achieve (2). Namely, we say that an $m \times N$ matrix M satisfies RIP-1 of order k with constant δ if for every k -sparse vector $x \in \mathbb{R}^N$,

$$(1 - \delta)\|x\|_1 \leq \|Mx\|_1 \leq (1 + \delta)\|x\|_1. \quad (3)$$

More generally, it is possible to consider RIP- p for the ℓ_p norm, where the norm used in the above guarantee is ℓ_p . As shown in [2], for any such matrix M , it is possible to obtain an approximation \tilde{x} satisfying (2) from the knowledge of Mx . In fact, such a reconstruction can be algorithmically achieved using convex optimization methods and in polynomial time in N .

Expanders and condensers. It is well known that RIP-1 matrices with zero-one entries (before normalization) are equivalent to adjacency matrices of unbalanced expander graphs, which are formally defined below.

Definition 4. A D -regular bipartite graph $G = (A, B, E)$ with A, B, E respectively defining the set of left vertices, right vertices, and edges, is said to be a (k, ϵ) -unbalanced expander graph if for every set $S \subseteq A$ such that $|S| \leq k$, we have $|\Gamma(S)| \geq (1 - \epsilon)D|S|$, where $\Gamma(S)$ denotes the neighborhood of S .

One direction of the above-mentioned characterization of binary RIP-1 matrices which is important for the present work is the following (which we will use only for the special case $p = 1$).

Theorem 5. ([2, Theorem 1]) Consider any $m \times N$ matrix Φ that is the adjacency matrix of a (k, ϵ) -unbalanced expander graph $G = (A, B, E)$, $|A| = N$, $|B| = m$, with left degree D , such that $1/\epsilon, D$ are smaller than N . Then, the scaled matrix $\Phi/D^{1/p}$ satisfies the RIP- p of order k with constant δ , for any $1 \leq p \leq 1 + 1/\log N$ and $\delta = C_0\epsilon$ for some absolute constant $C_0 > 1$.

Unbalanced expander graphs can be obtained from the truth tables of *lossless condensers*, a class of pseudorandom functions defined below. We first recall that the *min-entropy* of a distribution \mathcal{X} with finite support Ω is given by $H_\infty(\mathcal{X}) := \min_{x \in \Omega} \{-\log \mathcal{X}(x)\}$, where $\mathcal{X}(x)$ is the probability that \mathcal{X} assigns to the outcome x . The *statistical distance* between two distributions \mathcal{X} and \mathcal{Y} defined on the same finite space Ω is given by $\frac{1}{2} \sum_{s \in \Omega} |\mathcal{X}(s) - \mathcal{Y}(s)|$, which is half the ℓ_1 distance of the two distributions when regarded as vectors of probabilities over Ω . Two distributions \mathcal{X} and \mathcal{Y} are said to be ϵ -close if their statistical distance is at most ϵ .

Definition 6. A function $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^n$ is a (κ, ϵ) -lossless condenser if for every set $S \subseteq \mathbb{F}_2^n$ of size at most 2^κ , the following holds: Let $X \in \mathbb{F}_2^n$ be a random variable uniformly sampled from S and $Z \in [D]$ be uniformly random and independent of X . Then, the distribution of $(Z, h(X, Z))$ is ϵ -close in statistical distance to some distribution with min-entropy at least $\log(D|S|)$. A condenser is explicit if it is computable in polynomial time in n .

Ideally, the hope is to attain $r = \kappa + \log(1/\epsilon) + O(1)$ and $D = O(n/\epsilon)$. This is in fact achieved by a random function with high probability [8]. Equivalence of bipartite unbalanced expanders and lossless condensers was shown in [18]. Namely, we have the following.

Definition 7. Consider a function $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$. The (bipartite) graph associated with h is a bipartite graph $G = (\mathbb{F}_2^n, \mathbb{F}_2^r \times [D], E)$ with the edge set E defined as follows. For every $a \in \mathbb{F}_2^n$ and $(b, t) \in \mathbb{F}_2^r \times [D]$, there is an edge in E between a and (b, t) iff $h(a, t) = b$. For any choice of $t \in [D]$, we define the function $h_t: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^r$ by $h_t(x) := h(x, t)$. Then, the graph associated with h_t is defined as the subgraph of G induced by the restriction of the right vertices to the set $\{(b, t): b \in \mathbb{F}_2^r\}$. We say that h is *linear in the first argument* if h_t is linear over \mathbb{F}_2 for every fixed choice of t .

Lemma 8. ([18]) *A function $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$ is a (κ, ϵ) -lossless condenser if and only if the bipartite graph associated to h is a $(2^\kappa, \epsilon)$ -unbalanced expander.*

3 Obtaining nearly optimal sample complexity

Before focusing on the algorithmic aspect of sparse Hadamard transform, we demonstrate that deterministic sparse Hadamard transform is possible in information-theoretic sense. That is, as a warm-up we first focus on a sample-efficient algorithm without worrying about the running time. The key tool that we use is the following observation whose proof is discussed in Section 3.1.

Lemma 9. *Let $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$, where $r \leq n$, be a function computable in time $n^{O(1)}$ and linear in the first argument. Let $M \in \{0, 1\}^{D2^r \times 2^n}$ be the adjacency matrix of the bipartite graph associated with h (as in Definition 7). Then, for any $x \in \mathbb{R}^{2^n}$, the product Mx can be computed using only query access to $\hat{x} = \text{DHT}(x)$ from $D2^r$ deterministic queries to \hat{x} and in time $D2^r n^{O(1)}$.*

It is known that RIP-1 matrices suffice for sparse recovery in the ℓ_1/ℓ_1 model of (2). Namely.

Theorem 10. [2] *Let Φ be a real matrix with N columns satisfying RIP-1 of order k with sufficiently small constant $\delta > 0$. Then, for any vector $x \in \mathbb{R}^N$, there is an algorithm that given Φ and Φx computes an estimate $\tilde{x} \in \mathbb{R}^N$ satisfying (2) in time $N^{O(1)}$.*

By combining this result with Lemma 8, Theorem 5, and Lemma 9, we immediately arrive at the following result.

Theorem 11. *There are absolute constants $c, \epsilon > 0$ such that the following holds. Suppose there is an explicit linear $(\log k, \epsilon)$ -lossless condenser $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$ and let $N := 2^n$. Then, there is a deterministic algorithm running in time $N^{O(1)}$ that, given query access to $\hat{x} = \text{DHT}(x) \in \mathbb{R}^N$, non-adaptively queries \hat{x} at $D2^r$ locations and outputs $\tilde{x} \in \mathbb{R}^N$ such that*

$$\|\tilde{x} - x\|_1 \leq c \cdot \|x - H_k(x)\|_1.$$

Proof. Let $M \in \{0, 1\}^{D2^r \times N}$ be the adjacency matrix of the bipartite graph associated with the condenser h . By Lemma 8, M represents a (k, ϵ) -unbalanced expander graph. Thus by Theorem 5, M/D satisfies RIP of order k with constant $\delta = C_0\epsilon$. By Theorem 10, assuming ϵ (and thus δ) are sufficiently small constants, it suffices to show that the product Mx for a given vector $x \in \mathbb{R}^N$ can be computed efficiently by only querying \hat{x} non-adaptively at $D2^r$ locations. This is exactly what shown by Lemma 9. \square

One of the best known explicit constructions of lossless condensers is due to Guruswami et al. [8] that uses techniques from list-decodable algebraic codes. As observed by Cheraghchi [3], this construction can be modified to make the condenser linear. Namely, the above-mentioned result proves the following.

Theorem 12. [3, Corollary 2.23] *Let p be a fixed prime power and $\alpha > 0$ be an arbitrary constant. Then, for parameters $n \in \mathbb{N}$, $\kappa \leq n \log p$, and $\epsilon > 0$, there is an explicit linear (κ, ϵ) -lossless condenser $h: \mathbb{F}_p^n \times [D] \rightarrow \mathbb{F}_p^r$ satisfying $\log D \leq (1 + 1/\alpha)(\log(n\kappa/\epsilon) + O(1))$ and $r \log p \leq \log D + (1 + \alpha)\kappa$.*

For completeness, we include a proof of Theorem 12 in Appendix A. Combined with Theorem 11, we conclude the following.

Corollary 13. *For every $\alpha > 0$ and integer parameters $N = 2^n$, $k > 0$ and parameter $\epsilon > 0$, there is a deterministic algorithm running in time $N^{O(1)}$ that, given query access to $\hat{x} = \text{DHT}(x) \in \mathbb{R}^N$, non-adaptively queries \hat{x} at $O(k^{1+\alpha}(n \log k)^{2+2/\alpha}) = k^{1+\alpha}n^{O_\alpha(1)}$ coordinate positions and outputs $\tilde{x} \in \mathbb{R}^N$ such that*

$$\|\tilde{x} - x\|_1 \leq c \cdot \|x - H_k(x)\|_1,$$

for some absolute constant $c > 0$.

3.1 Proof of Lemma 9

For a vector $x \in \mathbb{F}_2^n$ and set $V \subseteq \mathbb{F}_2^n$, let $x(V)$ denote the summation

$$x(V) := \sum_{i \in V} x(i).$$

Lemma 9 is an immediate consequence of Lemma 15 below, before which we derive a simple proposition.

Proposition 14. *Let $V \subseteq \mathbb{F}_2^n$ be a linear space. Then for every $a \in \mathbb{F}_2^n$, we have*

$$x(a + V) = \frac{|V|}{\sqrt{N}} \sum_{j \in V^\perp} (-1)^{\langle a, j \rangle} \hat{x}(j).$$

Proof. We simply expand the summation according to the Hadamard transform formula (1) as follows.

$$\begin{aligned}
\sum_{i \in a+V} x(i) &= \sum_{i \in V} x(i+a) \\
&= \frac{1}{\sqrt{N}} \sum_{j \in \mathbb{F}_2^n} \sum_{i \in V} (-1)^{\langle a, j \rangle} (-1)^{\langle i, j \rangle} \hat{x}(j) \\
&= \frac{|V|}{\sqrt{N}} \sum_{j \in V^\perp} (-1)^{\langle a, j \rangle} \hat{x}(j),
\end{aligned}$$

where the last equality uses the basic linear-algebraic fact that

$$\sum_{i \in V} (-1)^{\langle i, j \rangle} = \begin{cases} |V| & \text{if } j \in V^\perp \\ 0 & \text{if } j \notin V^\perp. \end{cases}$$

□

Lemma 15. *Let $V \subseteq \mathbb{F}_2^n$ be a linear space and $W \subseteq \mathbb{F}_2^n$ be a linear space complementing V . That is, W is a linear sub-space such that $|W| \cdot |V| = N$ and $V + W = \mathbb{F}_2^n$. Then, the vector*

$$v := (x(a+V) : a \in W) \in \mathbb{R}^{|W|}$$

can be computed in time $O(|W| \log(|W|)n)$ and by only querying $\hat{x}(i)$ for all $i \in V^\perp$, assuming that the algorithm is given a basis for W and V^\perp .

Proof. We will use a divide and conquer approach similar to the standard Fast Hadamard Transform algorithm. Let $r := \dim(W) = \dim(V^\perp) = n - \dim(V)$. Fix a basis v_1, \dots, v_r of V^\perp and a basis w_1, \dots, w_r of W . For $i \in [r]$, let $V_i^\perp := \text{span}\{v_1, \dots, v_i\}$ and $W_i := \text{span}\{w_1, \dots, w_i\}$.

Let the matrix $H_i \in \{-1, +1\}^{2^i \times 2^i}$ be so that the rows and columns are indexed by the elements of W_i and V_i^\perp , respectively, with the entry at row i and column j defined as $(-1)^{\langle i, j \rangle}$. Using this notation, by Proposition 14 the problem is equivalent to computing the matrix-vector product $H_r z$ for any given $z \in \mathbb{R}^{2^r}$.

Note that $W_r = W_{r-1} \cup (w_r + W_{r-1})$ and similarly, $V_r^\perp = V_{r-1}^\perp \cup (v_r + V_{r-1}^\perp)$. Let $D_r \in \{-1, +1\}^{2^{r-1}}$ be a diagonal matrix with rows and columns indexed by the elements of W_{r-1} and the diagonal entry at position $w \in W_{r-1}$ be defined as $(-1)^{\langle w, v_r \rangle}$. Similarly, let $D'_r \in \{-1, +1\}^{2^{r-1}}$ be a diagonal matrix with rows and columns indexed by the elements of V_{r-1}^\perp and the diagonal entry at position $v \in V_{r-1}^\perp$ be defined as $(-1)^{\langle v, w_r \rangle}$. Let $z = (z_0, z_1)$ where $z_0 \in \mathbb{R}^{2^{r-1}}$ (resp., $z_1 \in \mathbb{R}^{2^{r-1}}$) is the restriction of Z to the entries indexed by V_{r-1}^\perp (resp., $v_r + V_{r-1}^\perp$). Using the above notation, we can derive the recurrence

$$H_r z = (H_{r-1} z_0 + D_r H_{r-1} z_1, H_{r-1} D'_r z_0 + (-1)^{\langle v_r, w_r \rangle} D_r H_{r-1} D'_r z_1).$$

Therefore, after calling the transformation defined by H_{r-1} twice as a subroutine, the product $H_r z$ can be computed using $O(2^r)$ operations on n -bit vectors. Therefore, the recursive procedure can compute the transformation defined by H_r using $O(r2^r)$ operations on n -bit vectors. \square

Using the above tools, we are now ready to finish the proof of Lemma 9. Consider any $t \in [D]$. Let $V \subseteq \mathbb{F}_2^n$ be the kernel of h_t and $N := 2^n$. Let M^t be the $2^r \times N$ submatrix of M consisting of rows corresponding to the fixed choice of t . Our goal is to compute $M^t \cdot x$ for all fixings of t . Without loss of generality, we can assume that h_t is surjective. If not, certain rows of M^t would be all zeros and the submatrix of M^t obtained by removing such rows would correspond to a surjective linear function h'_t whose kernel can be computed in time $n^{O(1)}$.

When h_t is surjective, we have $\dim V = n - r$. Let $W \subseteq \mathbb{F}_2^n$ be the space of coset representatives of V (i.e., $|W| = 2^r$ and $V + W = \mathbb{F}_2^n$). Note that we also have $|V^\perp| = 2^r$, and that a basis for W and V^\perp can be computed in time $n^{O(1)}$ (in fact, V^\perp is generated by the rows of the $r \times N$ transformation matrix defined by h_t , and a generator for W can be computed using Gaussian elimination in time $O(n^3)$).

By standard linear algebra, for each $y \in \mathbb{F}_2^r$ there is an $a(y) \in W$ such that $h_t^{-1}(y) = a(y) + V$ and that $a(y)$ can be computed in time $n^{O(1)}$. Observe that $M^t x$ contains a row for each y , at which the corresponding inner product is the summation $\sum_{i \in h_t^{-1}(y)} x(i) = x(a(y) + V)$. Therefore, the problem reduces to computing the vector $(x(a + V) : a \in W)$ which, according to Lemma 15, can be computed in time $O(r2^r)$ in addition to the $n^{O(1)}$ time required for computing a basis for W and V^\perp . By going over all choices of t , it follows that Mx can be computed as claimed. This concludes the proof of Lemma 9. \square

4 Obtaining nearly optimal reconstruction time

The modular nature of the sparse Hadamard transform algorithm presented in Section 3 reduces the problem to the general ℓ_1/ℓ_1 sparse recovery which is of independent interest. As a result, in order to make the algorithm run in sublinear time it suffices to design a sparse recovery algorithm analogous to the result of Theorem 10 that runs in sublinear time in N . In this section we construct such an algorithm, which is independently interesting for sparse recovery applications.

4.1 Augmentation of the sensing matrix

A technique that has been used in the literature for fast reconstruction of exactly k -sparse vectors is the idea of augmenting the measurement matrix with additional rows that guide the search process (cf. [2]). For our application, one obstacle that is not present in general sparse recovery is that the augmented sketch should be computable *only with access to Hadamard transform queries*. For this reason, crucially we cannot use any general sparse recovery algorithm as black box and have to specifically design an augmentation that is compatible with the restrictive model of Hadamard

transform queries. We thus restrict ourselves to tensor product augmentation with “bit selection” matrices defined as follows, and will later show that such augmentation can be implemented only using queries to the Hadamard coefficients.

Definition 16. The bit selection matrix $B \in \{0, 1\}^{n \times N}$ with n rows and $N = 2^n$ columns is a matrix with columns indexed by the elements of \mathbb{F}_2^n such that the entry of B at the j th row and i th column (where $j \in [n]$ and $i \in \mathbb{F}_2^n$) is the j th bit of i .

Definition 17. Let $A \in \{0, 1\}^m \times \{0, 1\}^N$ and $A' \in \{0, 1\}^{m'} \times \{0, 1\}^N$ be matrices. The tensor product $A \otimes A'$ is an $mm' \times N$ binary matrix with rows indexed by the elements of $[m] \times [m']$ such that for $i \in [m]$ and $i' \in [m']$, the rows of $A \otimes A'$ indexed by (i, i') is the coordinate-wise product of the i th row of A and i' th row of A' .

We will use tensor products of expander-based sensing matrices with bit selection matrix, and extend the result of Lemma 9 to such products.

Lemma 18. Let $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$, where $r \leq n$, be a function computable in time $n^{O(1)}$ and linear in the first argument, and define $N := 2^n$. Let $M \in \{0, 1\}^{D2^r \times N}$ be the adjacency matrix of the bipartite graph associated with h (as in Definition 7) and $M' := M \otimes B$ where $B \in \{0, 1\}^{n \times N}$ is the bit selection matrix with n rows. Then, for any $x \in \mathbb{R}^N$, the product $M'x$ can be computed using only query access to \hat{x} from $O(D2^r n)$ deterministic queries to \hat{x} and in time $D2^r n^{O(1)}$.

Proof. For each $b \in [n]$, define $h^b: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^{r+1}$ to be $h^b(x, z) := (h(x, z), x(b))$. Note that since h is linear over \mathbb{F}_2 , so is h^b for all b . Let $M_b'' \in \{0, 1\}^{D2^{r+1} \times N}$ be the adjacency matrix of the bipartite graph associated with h^b (as in Definition 7) and $M'' \in \{0, 1\}^{Dn2^{r+1} \times N}$ be the matrix resulting from stacking M_1'', \dots, M_n'' on top of each other. One can see that the set of rows of M'' contains the $Dn2^r$ rows of $M' = M \otimes B$.

By Lemma 9 (applied on all choices of h^b for $b \in [n]$), the product $M''x$ (and hence, $M'x$) can be computed using only query access to \hat{x} from $O(Dn2^r)$ deterministic queries to \hat{x} and in time $O(Drn2^r)$. This completes the proof. \square

In order to improve the running time of the algorithm in Theorem 11, we use the following result which is our main technical tool and discussed in Section 4.2.

Theorem 19. There are absolute constants $c > 0$ and $\epsilon > 0$ such that the following holds. Let k, n, L ($k \leq n$ and $\log L = n^{O(1)}$) be positive integer parameters, and suppose there exists a function $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$ (where $r \leq n$) which is an explicit $(\log(4k), \epsilon)$ -lossless condenser. Let M be the adjacency matrix of the bipartite graph associated with h and B be the bit-selection matrix with n rows and $N := 2^n$ columns. Then, there is an algorithm that, given k and vectors Mx and $(M \otimes B)x$ for some $x \in \mathbb{R}^N$ (which is not given to the algorithm and whose entries are $n^{O(1)}$ bits long), computes a k -sparse estimate \tilde{x} satisfying

$$\|\tilde{x} - x\|_1 \leq c \cdot \|x - H_k(x)\|_1.$$

Moreover, the running time of the algorithm is $O(2^r D^2 n^{O(1)})$.

The above result is proved using the algorithm discussed in Section 4.2. By using this result in conjunction with Lemma 18 in the proof of Theorem 11, we obtain our main result as follows.

Theorem 20. (Main) *There are absolute constants $c > 0$ and $\epsilon > 0$ such that the following holds. Let k, n ($k \leq n$) be positive integer parameters, and suppose there exists a function $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$ (where $r \leq n$) which is an explicit $(\log(4k), \epsilon)$ -lossless condenser and is linear in the first argument. Then, there is a deterministic algorithm running in time $2^r D^2 n^{O(1)}$ that, given (non-adaptive) query access to $\hat{x} \in \mathbb{R}^N$ (where $N := 2^n$, and each entry of \hat{x} is $n^{O(1)}$ bits long), outputs $\tilde{x} \in \mathbb{R}^N$ such that*

$$\|\tilde{x} - x\|_1 \leq c \cdot \|x - H_k(x)\|_1.$$

Proof. We closely follow the proof of Theorem 11, but in the proof use Theorem 19 instead of Theorem 10.

Since each entry of \hat{x} is $n^{O(1)}$ bits long and the Hadamard transform matrix (after normalization) only contains ± 1 entries, we see that each entry of $\sqrt{N}x$ is $n^{O(1)}$ bits long as well.

Let M be the adjacency matrix of the bipartite expander graph associated with h , B be the bit selection matrix with n rows, and $M' := M \otimes B$. By the argument of Theorem 11, the product Mx can be computed in time $2^r D n^{O(1)}$ only by non-adaptive query access to \hat{x} . Same is true for the product $M'x$ using a similar argument and using Lemma 18. Once computed, this information can be passed to the algorithm guaranteed by Theorem 19 to compute the desired estimate on x . \square

Finally, by using the condenser of Theorem 12 in the above theorem, we immediately obtain Theorem 1 as a corollary, which is restated below.

Theorem 1 (restated). *For every fixed constant $\alpha > 0$, there is a deterministic algorithm as follows. Let $N = 2^n$ and $k \leq N$ be positive integers. Then, given (non-adaptive) query access to any $\hat{x} \in \mathbb{R}^N$ where each coefficient of \hat{x} is $n^{O(1)}$ bits long, the algorithm runs in time $k^{1+\alpha} n^{O(1)}$ and outputs $\tilde{x} \in \mathbb{R}^N$ that satisfies (2) (where $\hat{x} = \text{DHT}(x)$) for some absolute constant $c > 0$.*

4.2 The sparse recovery algorithm

The claim of Theorem 19 is shown using the algorithm presented in Figure 1. In this algorithm, M' is the $D2^r(n+1) \times N$ formed by stacking M on top of $M \otimes B$ and the algorithm is given $y := M'x$ for a vector $x \in \mathbb{R}^N$ to be approximated. For each $t \in [D]$, we define the $2^r \times N$ matrix M^t to be the adjacency matrix of the bipartite graph G^t associated with h_t (according to Definition 7). For $b \in [n]$ we let $B_b \in \{0, 1\}^{1 \times N}$ be the b th row of B . We assume that the entries of y are indexed by the set $\mathbb{F}_2^r \times [D] \times \{0, \dots, n\}$ where the entry $(a, t, 0)$ corresponds to the inner product defined by the a th row of M^t and the entry (a, t, b) (for $b \neq 0$) corresponds to the a th row of $M^t \otimes B_b$. Since each entry of x is $n^{O(1)}$ bits long, by using appropriate scaling we can without loss of generality

```

SEARCH( $j \in \mathbb{F}_2^r, t \in [D], s \in \mathbb{N}$ )
1  for  $b = 1$  to  $n$ 
2      if  $|y^{s,t,b}(j)| \geq |y^{s,t,0}(j)|/2$ 
3           $u_b = 1.$ 
4      else
5           $u_b = 0.$ 
6  return  $(u_1, \dots, u_n).$ 

ESTIMATE( $t \in [D], s \in \mathbb{N}$ )
1  Initialize  $S \subseteq \mathbb{F}_2^n$  as  $S = \emptyset.$ 
2  Initialize  $\Delta^{s,t} \in \mathbb{R}^N$  as  $\Delta^{s,t} = 0.$ 
3  Let  $T \subseteq \mathbb{F}_2^r$  be the set of coordinate positions corresponding to
   the largest  $2k$  entries of  $y^{s,t,0}.$ 
4  for  $j \in T$ 
5       $u = \text{SEARCH}(j, t, s).$ 
6      if  $h(u, t) \in T$ 
7           $S = S \cup \{u\}.$ 
8           $\Delta^{s,t}(u) = y^{s,t,0}(h(u, t)).$ 
9  return  $\Delta^{s,t}.$ 

RECOVER( $y \in \mathbb{R}^{2^r D n}, s_0 \in \mathbb{N}$ )
1   $s = 0.$ 
2  Let  $B_1, \dots, B_n \in \{0, 1\}^{1 \times N}$  be the rows of the bit selection matrix  $B.$ 
3  Initialize  $x^0 \in \mathbb{R}^N$  as  $x^0 = 0.$ 
4  for  $(t, b, j) \in [D] \times \{0, \dots, n\} \times \mathbb{F}_2^r$ 
5       $y^{0,t,b}(j) = y(j, t, b).$ 
6  repeat
7      for  $t \in [D]$ 
8           $y^{s,t,0} = M^t \cdot (x - x^s) \in \mathbb{R}^{2^r}.$ 
9          for  $b \in [n]$ 
10              $y^{s,t,b} = (M^t \otimes B_b) \cdot (x - x^s) \in \mathbb{R}^{2^r}.$ 
11              $\Delta^{s,t} = \text{ESTIMATE}(t, s).$ 
12             Let  $t_0$  be the choice of  $t \in [D]$  that minimizes  $\|Mx - M(x^s + \Delta^{s,t})\|_1.$ 
13              $x^{s+1} = H_k(x^s + \Delta^{s,t_0}).$ 
14              $s = s + 1.$ 
15 until  $s = s_0.$ 
16 Set  $x^*$  to be the choice of  $x^s$  (for  $s = 0, \dots, s_0$ ) that minimizes  $\|Mx - Mx^s\|_1.$ 
17 return  $x^*.$ 

```

Figure 1: Pseudo-code for the reconstruction algorithm $\text{RECOVER}(y, s_0)$, where y is the sketch $M'x$ and s_0 specifies the desired number of iterations. It suffices to set $s_0 = n^{O(1)}$ according to the bit length of x . Notation is explained in Section 4.2.

assume that x has integer entries in range $[-L, +L]$ for some L such that $\log L = n^{O(1)}$, and the algorithm's output can be rounded to the nearest integer in each coordinate so as to make sure that the final output is integral.

The main ingredient of the analysis is the following lemma which is proved in Appendix 4.5.

Lemma 21. *For every constant $\gamma > 0$, there is an ϵ_0 only depending on γ such that if $\epsilon \leq \epsilon_0$ the following holds. Suppose that for some s ,*

$$\|x - x^s\|_1 > C\|x - H_k(x)\|_1$$

for $C = 1/\epsilon$. Then, there is a $t \in [D]$ such that

$$\|x - (x^s + \Delta^{s,t})\|_1 \leq \gamma\|x - x^s\|_1.$$

The above lemma can be used, in conjunction with the fact that M satisfies RIP-1, to show that if ϵ is a sufficiently small constant, we can ensure exponential progress $\|x - x^{s+1}\|_1 \leq \|x - x^s\|_1/2$ (shown in Corollary 27) until the approximation error $\|x - x^s\|_1$ reaches the desired level of $C\|x - H_k(x)\|_1$ (after the final truncation). Then it easily follows that $s_0 = \log(NL) + O(1) = n^{O(1)}$ iterations would suffice to deduce Theorem 19. Formal proof of Theorem 19 appears in Section 4.4.

4.3 Analysis of the running time

In order to analyze the running time of the procedure RECOVERY, we first observe that all the estimates x^0, \dots, x^{s_0} are k -sparse vectors and can be represented in time $O(k(\log n + \log L))$ by only listing the positions and values of their non-zero entries. In this section we assume that all sparse N -dimensional vectors are represented in such a way. We observe the following.

Proposition 22. *Let $w \in \mathbb{R}^N$ be k -sparse. Then, for any $t \in [D]$, the products $(M^t \otimes B) \cdot w$ and $M^t w$ can be computed in time $n^{O(1)}(k + 2^r)\ell$, assuming each entry of w is represented within ℓ bits of precision.*

Proof. Let $B_1, \dots, B_n \in \{0, 1\}^{1 \times N}$ be the rows of the bit selection matrix B . Observe that each column of M^t is entirely zero except for a single 1 (this is because M^t represents the truth table of the function h_t). The product $M^t \cdot w$ is simply the addition of at most k such 1-sparse vectors, and thus, is itself k -sparse. The nonzero entries of $M^t \cdot w$ along with their values can thus be computed by querying the function h_t in up to k points (corresponding to the support of w) followed by k real additions. Since h_t can be computed in polynomial time in n , we see that $M^t \cdot w$ can be computed in time $n^{O(1)}(k + 2^r)\ell$ (we may assume that the product is represented trivially as an array of length 2^r and thus it takes 2^r additional operations to initialize the result vector). The claim then follows once we observe that for every $b \in [n]$, the matrix $M^t \otimes B_b$ is even more sparse than M^t . \square

Observe that the procedure SEARCH needs $O(n)$ operations. In procedure ESTIMATE, identifying T takes $O(2^r)$ time, and the loop runs for $2k$ iterations, each taking $O(nk)$ time. In procedure RECOVER, we note that for all t , $M^t x$ as well as $(M^t \otimes B_b)x$ for all $b \in [n]$ is given as a part of y at the input. Moreover, all the vectors x^s and $\Delta^{s,t}$ are $O(k)$ -sparse. Thus in light of Proposition 22 and noting that $L = 2^{n^{O(1)}}$ and the fact that h is a lossless condenser (which implies $2^r = \Omega(k)$), we see that computation of each product in Lines 8 and 10 of procedure RECOVER takes time $n^{O(1)}2^r$. Since the for loop runs for D iterations and so is the number of iterations, the running time of the loop is $n^{O(1)}D2^r$. With a similar reasoning, the computation in Line 12 takes time $n^{O(1)}D^22^r$. Similarly, computation of the product in Line 16 of procedure RECOVER takes time $n^{O(1)}D^22^r s_0$. Altogether, recalling that $s_0 = \log(NL) + O(1) = n^{O(1)}$, the total running time is $n^{O(1)}D^22^r$.

4.4 Proof of Theorem 19

Theorem 19 is proved using the algorithm presented in Figure 1 and discussed in Section 4.2. We aim to set up the algorithm so that it outputs a k -sparse estimate $\tilde{x} \in \mathbb{R}^N$ satisfying (2). Instead of achieving this goal, we first consider the following slightly different estimate

$$\|\tilde{x} - x\|_1 \leq C\|x - H_k(x)\|_1 + \nu\|x\|_1, \quad (4)$$

for an absolute constant $C > 0$, where $\nu > 0$ is an arbitrarily small “relative error” parameter. Let us show that this alternative guarantee implies (2), after rounding the estimate obtained by the procedure RECOVER to the nearest integer vector. Recall that without loss of generality (by using appropriate scaling), we can assume that x has integer coordinates in range $[-L, +L]$, for some L satisfying $\log L = n^{O(1)}$.

Proposition 23. *Let $x \in \mathbb{R}^N$ be an integer vector with integer coordinates in range $[-L, +L]$, and $\tilde{x} \in \mathbb{R}^N$ be so that (4) holds for some $\nu \leq 1/(4NL)$. Let \tilde{x}' be the vector obtained by rounding each entry of \tilde{x} to the nearest integer. Then, \tilde{x}' satisfies*

$$\|\tilde{x}' - x\|_1 \leq (3C + 1/2) \cdot \|x - H_k(x)\|_1.$$

Proof. If $x = 0$, there is nothing to show. Thus we consider two cases.

Case 1: $\|x - H_k(x)\|_1 = 0$. In this case, since $\|x\|_1 \leq NL$, we see that $\|\tilde{x} - x\|_1 \leq 1/4$. Therefore, rounding \tilde{x} to the nearest integer vector would exactly recover x .

Case 2: $\|x - H_k(x)\|_1 > 0$. Since x is an integer vector, we have $\|x - H_k(x)\|_1 \geq 1$. Therefore, again noting that $\|x\|_1 \leq NL$, from (4) we see that

$$\|\tilde{x} - x\|_1 \leq (C + 1/4) \cdot \|x - H_k(x)\|_1.$$

Therefore, by an averaging argument, the number of the coordinate positions at which \tilde{x} is different from x by $1/2$ or more is at most $2(C + 1/4) \cdot \|x - H_k(x)\|_1$. Since rounding can only cause error at such positions, and by at most 1 per coordinate, the added error caused by rounding would be at most $2(C + 1/4) \cdot \|x - H_k(x)\|_1$, and the claim follows. \square

In light of Proposition 23 above, in the sequel we focus on achieving (4), for a general ν , and will finally choose $\nu := 1/(4NL)$ so that using Proposition 23 we can attain the original estimate in (2). We remark that Proposition 23 is the only place in the proof that assumes finite precision for x and we do not need such an assumption for achieving (4).

A key ingredient of the analysis is the following result (Lemma 25 below) shown in [2]. Before presenting the result, we define the following notation.

Definition 24. Let $w = (w_1, \dots, w_N) \in \mathbb{R}^N$ be any vector and G be any bipartite graph with left vertex set $[N]$ and edge set E . Then, $\text{First}(G, w)$ denotes the following subset of edges:

$$\text{First}(G, w) := \{e = (i, j) \in E \mid (\forall e' = (i', j) \in E): (|w_i| > |w_{i'}|) \vee (|w_i| = |w_{i'}| \wedge i' > i)\}.$$

Lemma 25. [2] Let G be a (k', ϵ) -unbalanced expander graph with left vertex set $[N]$ and edge set E . Then, for any k' -sparse vector $w = (w_1, \dots, w_N) \in \mathbb{R}^N$, we have

$$\sum_{(i,j) \in E \setminus \text{First}(G,w)} |w_i| \leq \epsilon \sum_{(i,j) \in E} |w_i|.$$

Intuitively, for every right vertex in G , $\text{First}(G, w)$ picks exactly one edge connecting the vertex to the left neighbor at which w has the highest magnitude (with ties broken in a consistent way), and Lemma 25 shows that these edges pick up most of the ℓ_1 mass of w .

We apply Lemma 25 to the graph G that we set to be the graph associated with the function h . Note that this graph is a $(4k, \epsilon)$ -unbalanced expander by Lemma 8. This means that for every $(4k)$ -sparse vector w and letting E denote the edge set of G , we have

$$\sum_{(i,j) \in E \setminus \text{First}(G,w)} |w_i| \leq \epsilon \sum_{(i,j) \in E} |w_i| = \epsilon D \|w\|_1,$$

where the last equality uses the fact that G is D -regular from left. By an averaging argument, and noting that G is obtained by taking the union of the edges of graphs G^1, \dots, G^D (each of which being 1-regular from left), we get that for some $t(G, w) \in [D]$,

$$\sum_{(i,j) \in E^{t(G,w)} \setminus \text{First}(G,w)} |w_i| \leq \epsilon \|w\|_1, \tag{5}$$

where $E^{t(G,w)}$ denotes the edge set of $G^{t(G,w)}$.

Our goal will be to show that the algorithm converges exponentially to the near-optimal solution. In particular, in the following we show that if the algorithm is still “far” from the optimal solution on the s th iteration, it obtains an improved approximation for the next iteration. This is made precise in Lemma 21, which we recall below.

Lemma 21. (restated) For every constant $\gamma > 0$, there is an ϵ_0 only depending on γ such that if $\epsilon \leq \epsilon_0$ the following holds. Suppose that for some s ,

$$\|x - x^s\|_1 > C\|x - H_k(x)\|_1 \quad (6)$$

for $C = 1/\epsilon$. Then, there is a $t \in [D]$ such that

$$\|x - (x^s + \Delta^{s,t})\|_1 \leq \gamma\|x - x^s\|_1. \quad (7)$$

The proof of Lemma 21 is deferred to Section 4.5.

Proposition 26. Suppose $x', x'' \in \mathbb{R}^N$ are $(3k)$ -sparse and satisfy

$$\|M(x - x')\|_1 \leq \|M(x - x'')\|_1.$$

Then,

$$\|x - x'\|_1 \leq \left(1 + \frac{3 + C_0\epsilon}{1 - C_0\epsilon}\right)\|x - H_k(x)\|_1 + \frac{1 + C_0\epsilon}{1 - C_0\epsilon} \cdot \|x - x''\|_1$$

where C_0 is the constant in Theorem 5. In particular when $C_0\epsilon \leq 1/2$, we have

$$\|x - x'\|_1 \leq 8\|x - H_k(x)\|_1 + 3\|x - x''\|_1.$$

Proof.

$$\|x - x'\|_1 \leq \|x - H_k(x)\|_1 + \|H_k(x) - x'\|_1 \quad (8)$$

$$\leq \|x - H_k(x)\|_1 + \frac{\|MH_k(x) - Mx'\|_1}{D(1 - C_0\epsilon)} \quad (9)$$

$$\leq \|x - H_k(x)\|_1 + \frac{\|Mx - Mx'\|_1 + \|M(x - H_k(x))\|_1}{D(1 - C_0\epsilon)} \quad (10)$$

$$\leq \|x - H_k(x)\|_1 + \frac{\|Mx - Mx''\|_1 + \|M(x - H_k(x))\|_1}{D(1 - C_0\epsilon)} \quad (11)$$

$$\leq \|x - H_k(x)\|_1 + \frac{\|MH_k(x) - Mx''\|_1 + 2\|M(x - H_k(x))\|_1}{D(1 - C_0\epsilon)} \quad (12)$$

$$\leq \|x - H_k(x)\|_1 + \frac{(1 + C_0\epsilon)\|H_k(x) - x''\|_1 + 2\|x - H_k(x)\|_1}{(1 - C_0\epsilon)} \quad (13)$$

$$\leq \|x - H_k(x)\|_1 + \frac{(1 + C_0\epsilon)\|x - x''\|_1 + (3 + C_0\epsilon)\|x - H_k(x)\|_1}{(1 - C_0\epsilon)} \quad (14)$$

$$\leq \left(1 + \frac{3 + C_0\epsilon}{1 - C_0\epsilon}\right)\|x - H_k(x)\|_1 + \frac{1 + C_0\epsilon}{1 - C_0\epsilon} \cdot \|x - x''\|_1 \quad (15)$$

In the above, (8), (10), (12), and (14) use the triangle inequality (after adding and subtracting $H_k(x)$, Mx , $MH_k(x)$, and x inside the norms, respectively); (9) and (13) use RIP-1 of the matrix M (seeing that x' , x'' , and $H_k(x)$ are sufficiently sparse); (11) uses the assumption that $\|M(x - x')\|_1 \leq \|M(x - x'')\|_1$; (13) also uses the fact that all columns of M have Hamming weight D and thus the matrix cannot increase the ℓ_1 norm of any vector by more than a factor D . \square

The following corollary is implied by Lemma 21.

Corollary 27. *For every constant $\gamma_0 > 0$, there is an ϵ_0 only depending on γ_0 such that if $\epsilon \leq \epsilon_0$ the following holds. Assume condition (6) of Lemma 21 holds. Then,*

$$\|x - x^{s+1}\|_1 \leq \gamma_0 \|x - x^s\|_1.$$

Proof. Let $t_0 \in [D]$ be the value computed in Line 12 of the procedure RECOVER, and $t \in [D]$ be the value guaranteed to exist by Lemma 21. From the fact that the algorithm picks t_0 to be the minimizer of the quantity $\|Mx - M(x^s + \Delta^{s,t})\|_1$ for all $t \in [D]$, we have that

$$\|Mx - M(x^s + \Delta^{s,t_0})\|_1 \leq \|Mx - M(x^s + \Delta^{s,t})\|_1.$$

Note that x^s is k -sparse and Δ^{s,t_0} and $\Delta^{s,t}$ are $(2k)$ -sparse. Thus we can apply Proposition 26 and deduce that

$$\|x - (x^s + \Delta^{s,t_0})\|_1 \leq \left(1 + \frac{3 + C_0\epsilon}{1 - C_0\epsilon}\right) \|x - H_k(x)\|_1 + \frac{1 + C_0\epsilon}{1 - C_0\epsilon} \cdot \|x - (x^s + \Delta^{s,t})\|_1.$$

Plugging in the bound implied by Lemma 21 and (6) in the above inequality we get

$$\|x - (x^s + \Delta^{s,t_0})\|_1 \leq \gamma' \|x - x^s\|_1, \quad (16)$$

where we have defined

$$\gamma' := \epsilon \left(1 + \frac{3 + C_0\epsilon}{1 - C_0\epsilon}\right) + \frac{\gamma(1 + C_0\epsilon)}{(1 - C_0\epsilon)}.$$

Now, we can write

$$\begin{aligned} \|x - x^{s+1}\|_1 &= \|x - H_k(x^s + \Delta^{t_0,s})\|_1 \\ &\leq \|x - (x^s + \Delta^{t_0,s})\|_1 + \|x^s + \Delta^{t_0,s} - H_k(x^s + \Delta^{t_0,s})\|_1 \end{aligned} \quad (17)$$

$$\leq \|x - (x^s + \Delta^{t_0,s})\|_1 + \|x^s + \Delta^{t_0,s} - H_k(x)\|_1 \quad (18)$$

$$\leq 2\|x - (x^s + \Delta^{t_0,s})\|_1 + \|x - H_k(x)\|_1 \quad (19)$$

$$\leq (2\gamma' + \epsilon)\|x - x^s\|_1. \quad (20)$$

In the above, (17) and (19) use the triangle inequality (after adding and subtracting $x^s + \Delta^{t_0,s}$ inside the norm); (18) uses the fact that $H_k(x)$ and $H_k(x^s + \Delta^{t_0,s})$ are both k -sparse by definition and $H_k(x^s + \Delta^{t_0,s})$ is the best approximator of $x^s + \Delta^{t_0,s}$ among all k -sparse vectors; and (20) uses (6) and (16). Finally, note that we can choose γ and ϵ small enough so that $2\gamma' + \epsilon \leq \gamma_0$. \square

For the rest of the analysis, we set ϵ a small enough constant so that

1. $C_0\epsilon \leq 1/2$, where C_0 is the constant in Theorem 5.
2. $\gamma_0 = 1/2$, where γ_0 is the constant in Corollary 27.

Observe that for the first iteration of the algorithm, the estimation error is $\|x - x^0\|_1 = \|x\|_1$. By repeatedly applying the exponential decrease guaranteed by Corollary 27, we see that as long as $s_0 \geq \log(3/\nu)$, we can ensure that at some stage $s \leq s_0$ we attain

$$\|x - x^s\|_1 \leq C\|x - H_k(x)\|_1 + (\nu/3)\|x\|_1.$$

Let x^* be the estimate computed in the end of procedure RECOVER. Recall that both x^* and x^s are k -sparse vectors. Thus, by Proposition 26 we see that

$$\|x - x^*\|_1 \leq 8\|x - H_k(x)\|_1 + 3\|x - x^s\|_1 \leq (3C + 8) \cdot \|x - H_k(x)\|_1 + \nu\|x\|_1.$$

Finally, as discussed in the beginning of the analysis, by choosing $\nu := 1/(4NL)$ (and thus, $s_0 = \log(NL) + O(1) = n^{O(1)}$) and using Proposition 23, the analysis (and proof of Theorem 19) is complete. \square

4.5 Proof of Lemma 21

We start with some notation. Let U denote the set of the k largest (in magnitude) coefficients of x , and let V be the support of x^s . Furthermore, we set $W = U \cup V$ and $z = x - x^s$. That is, z is the vector representing the current estimation error vector. Note that $|W| \leq 2k$ and that $H_k(x) = x_U$. With a slight abuse of notation, we will use the sets $\{0, 1\}^n$ and $[N]$ interchangeably (for example in order to index coordinate positions of z) and implicitly assume the natural n -bit representation of integers in $[N]$ in doing so.

We first apply the result of Lemma 25 to the vector z_W so as to conclude that, for some $t \in [D]$, according to (5) we have

$$\sum_{\substack{(i,j) \in E^t \setminus \text{First}(G, z_W) \\ i \in W}} |z(i)| \leq \epsilon \|z_W\|_1. \quad (21)$$

We fix one particular such choice of t for the rest of the proof. Define the set

$$D := \{i \in [N] \mid (i, h_t(i)) \in \text{First}(G, z_W)\}.$$

Intuitively, $\text{First}(G, z_W)$ resolves collisions incurred by h_t by picking, for each hash output, only the pre-image with the largest magnitude (according to z_W). In other words, $\text{First}(G, z_W)$ induces a partial function from $[N]$ to \mathbb{F}_2^r that is one-to-one, and D defines the domain of this partial function. Using (21), we thus have

$$\|z_{W \setminus D}\|_1 = \sum_{i \in W \setminus D} |z(i)| \leq \epsilon \|z_W\|_1 \leq \epsilon \|z\|_1. \quad (22)$$

Define, for any $i \in [N]$,

$$d_i := \left\| z_{h_t^{-1}(h_t(i)) \setminus \{i\}} \right\|_1 = \sum_{i' \in h_t^{-1}(h_t(i)) \setminus \{i\}} |z(i')|. \quad (23)$$

Intuitively, with respect to the hash function h_t , the quantity d_i collects all the mass from elsewhere that fall into the same bin as i .

Our aim is to show that $\Delta^{s,t}$ which is the estimate on the error vector produced by the algorithm recovers “most” of the coefficients in z_W , and is therefore “close” to the actual error vector z .

Our analysis will focus on coefficients in z_W that are “good” in the following sense. Formally, we define the set of *good* coefficients \mathcal{G} to contain coefficients i such that:

1. $i \in W \cap D$, and,
2. $d_i < \delta|z(i)|$, for some small parameter $\delta \leq 1/4$ to be determined later.

Intuitively, \mathcal{G} is the set of coefficients i that “dominate” their bucket mass $y(h_t(i))$. Thus applying the binary search on any such bucket (i.e., procedure $\text{SEARCH}(h_t(i), t, s)$) will return the correct value i (note that the above definition implies that for any $i \in \mathcal{G}$, we must have $y^{s,t,0}(h_t(i)) \neq 0$, and thus the binary search would not degenerate). More formally, we have the following.

Proposition 28. *For any $i \in \mathcal{G}$, the procedure $\text{SEARCH}(h_t(i), t, s)$ returns i .*

Proof. Consider the sequence (u_1, \dots, u_n) produced by the procedure SEARCH and any $b \in [n]$. Recall that $y^{s,t,0} = M^t z$ and for each $b \in [n]$, $y^{s,t,b} = (M^t \otimes B_b) \cdot z$. Let $j := h_t(i)$. Since $i \in \mathcal{G}$, we have $d_i < \delta|z(i)| \leq |z(i)|/2$. Therefore,

$$|z(i)|(1 - \delta) < |y^{s,t,0}(j)| < z(i)(1 + \delta). \quad (24)$$

Let $b \in [n]$ and $v \in \{0, 1\}$ be the b th bit in the n -bit representation of i . Let S be the set of those elements in $h_t^{-1}(j) \subseteq \{0, 1\}^n$ whose b th bit is equal to 1. Note that $i \in S$ iff $v = 1$. Recall that

$$y^{s,t,b}(j) = \sum_{i' \in S} z(i').$$

Whenever $i \notin S$, we get

$$|y^{s,t,b}(j)| = \left| \sum_{i' \in S} z(i') \right| \leq \sum_{i' \in h_t^{-1}(j) \setminus \{i\}} |z(i')| = d_i < \delta|z(i)| < \frac{\delta|y^{s,t,0}(j)|}{1 - \delta},$$

according to the definition of d_i and (24). On the other hand, when $i \in S$, we have

$$|y^{s,t,b}(j)| \geq |z(i)| - \left| \sum_{i' \in S \setminus \{i\}} z(i') \right| \geq |z(i)| - d_i > |z(i)|(1 - \delta) > \frac{(1 - \delta)|y^{s,t,0}(j)|}{1 + \delta},$$

again according to the definition of d_i and (24). Thus, the procedure SEARCH will be able to distinguish between the two cases $i \in S$ and $i \notin S$ (equivalently, $v = 1$ and $v = 0$) and correctly set $u_b = v$ provided that

$$\frac{\delta}{1 - \delta} < \frac{1}{2}$$

and

$$\frac{1 - \delta}{1 + \delta} \geq \frac{1}{2}$$

which is true according to the choice $\delta \leq 1/4$. \square

By rewriting assumption (6) of the lemma, we know that

$$\|z\|_1 \geq C\|x_{\overline{U}}\|_1,$$

and thus,

$$\|z_{\overline{W}}\|_1 = \|x_{\overline{W}}\|_1 \leq \|x_{\overline{U}}\|_1 \leq \|z\|_1/C = \epsilon\|z\|_1, \quad (25)$$

where the first equality uses the fact that x and $z = x - x^s$ agree outside $V = \text{supp}(x^s)$ (and thus, outside W) and we also recall that $W \subseteq U$.

Observe that for each $i, i' \in D$ such that $i \neq i'$, we have $h_t(i) \neq h_t(i')$ (since $\text{First}(G, z_W)$ picks exactly one edge adjacent to the right vertex $h_t(i)$, namely $(i, h_t(i))$, and exactly one adjacent to $h_t(i')$, namely $(i', h_t(i'))$). In other words for each $i \in D$, the set $h_t^{-1}(h_t(i))$ cannot contain any element of D other than i . Therefore, we have

$$\sum_{i \in W \cap D} d_i \leq \|z_{\overline{D}}\|_1 \leq \|z_{W \setminus D}\|_1 + \|z_{\overline{W}}\|_1 \leq 2\epsilon\|z\|_1, \quad (26)$$

where for the last inequality we have used (22) and (25).

Now we show that a substantial portion of the ℓ_1 mass of z is collected by the set of good indices \mathcal{G} .

Lemma 29. $\sum_{i \in \mathcal{G}} |z(i)| \geq (1 - 2\epsilon(1 + 1/\delta))\|z\|_1$.

Proof. We will upper bound $\sum_{i \notin \mathcal{G}} |z(i)|$, and in order to do so, decompose this sum into three components bounded as follows:

- $\sum_{i \notin W} |z(i)| \leq \epsilon\|z\|_1$ (according to (25))
- $\sum_{i \in W \setminus D} |z(i)| \leq \epsilon\|z\|_1$ (according to (22))
- $\sum_{(W \cap D) \setminus \mathcal{G}} |z(i)| \leq 2\epsilon/\delta\|z\|_1$. In order to verify this claim, observe that from the definition of \mathcal{G} , every $i \notin \mathcal{G}$ satisfies $|z(i)| \leq d_i/\delta$. Therefore, the left hand side summation is at most $\sum_{i \in W \cap D} |d_i|/\delta$ and the bound follows using (26).

By adding up the above three partial summations, the claim follows. \square

Lemma 29 shows that it suffices to recover most of the coefficients z_i for $i \in \mathcal{G}$ in order to recover most of the ℓ_1 mass in z . This is guaranteed by the following lemma.

Lemma 30. *There is a $\beta > 0$ only depending on ϵ and δ such that $\beta = O_\delta(\epsilon)$ and*

$$\sum_{i \in \mathcal{G}, h_t(i) \in T} |z(i)| \geq (1 - \beta) \|z\|_1,$$

where T is the set define in Line 3 of the procedure ESTIMATE.

Proof. Consider the bin vector $y := y^{s,t,0} = M^t z$. From the choice of T as the set picking the largest $2k$ coefficients of y , it follows that for all $j \in T \setminus h_t(\mathcal{G})$ and $j' \in h_t(\mathcal{G}) \setminus T$ (where $h_t(\mathcal{G})$ denotes the set $\{h_t(i) \mid i \in \mathcal{G}\}$) we have $|y(j)| \geq |y(j')|$. Since $|T| = 2k$ and $|h_t(\mathcal{G})| \leq 2k$ (because $\mathcal{G} \subseteq W$ which is in turn $(2k)$ -sparse), it follows that $|T \setminus h_t(\mathcal{G})| \geq |h_t(\mathcal{G}) \setminus T|$. Therefore,

$$\sum_{j \in h_t(\mathcal{G}) \setminus T} |y(j)| \leq \sum_{j \in T \setminus h_t(\mathcal{G})} |y(j)|.$$

Now, using Lemma 29 we can deduce the following.

$$\sum_{j \in T \setminus h_t(\mathcal{G})} |y(j)| \leq \sum_{i \notin \mathcal{G}} |z(i)| \leq 2\epsilon(1 + 1/\delta) \|z\|_1 \quad (27)$$

where for the first inequality we note that $y(j) = \sum_{i \in h_t^{-1}(j)} z(i)$ and that the sets $h_t^{-1}(j)$ for various j are disjoint and cannot intersect \mathcal{G} unless, by definition, $j \in h_t(\mathcal{G})$.

Recall that for every $i \in \mathcal{G}$, by the definition of \mathcal{G} we have

$$(1 - \delta)|z(i)| < |y(h_t(i))| < (1 + \delta)|z(i)|.$$

Using this, it follows that

$$\begin{aligned} \sum_{i \in \mathcal{G}, h_t(i) \in T} |z(i)| &\geq \frac{1}{1 + \delta} \sum_{j \in h_t(\mathcal{G}) \cap T} |y(j)| \\ &\geq \frac{1}{1 + \delta} \left(\sum_{j \in h_t(\mathcal{G})} |y(j)| - \sum_{j \in h_t(\mathcal{G}) \setminus T} |y(j)| \right) \\ &\geq \frac{1}{1 + \delta} \left((1 - \delta) \sum_{i \in \mathcal{G}} |z(i)| - \sum_{j \in h_t(\mathcal{G}) \setminus T} |y(j)| \right) \\ &\geq \frac{(1 - \delta)(1 - 2\epsilon(1 + 1/\delta)) - 2\epsilon(1 + 1/\delta)}{1 + \delta} \|z\|_1 =: (1 - \beta) \|z\|_1, \end{aligned}$$

where the last step follows from Lemma 29 and (27). \square

We are now ready to conclude the proof of Lemma 21. First, observe using Proposition 28 that for coordinates $i \in \mathcal{G}$ such that $h_t(i) \in T$, we have $\Delta^{s,t}(i) = y(h_t(i))$ and that, since $i \in \mathcal{G}$,

$$z(i)(1 - \delta) \leq z(i) - d_i \leq y(h_t(i)) \leq z(i) + d_i \leq z(i)(1 + \delta). \quad (28)$$

Therefore, for such choices of i , $|\Delta^{s,t}(i) - z(i)| \leq \delta|z(i)|$. Thus we have

$$\|\Delta^{s,t} - z\|_1 = \sum_{i \in \mathcal{G} \cap h_t^{-1}(T)} |\Delta^{s,t}(i) - z(i)| + \sum_{i \notin \mathcal{G} \cap h_t^{-1}(T)} |\Delta^{s,t}(i) - z(i)| \quad (29)$$

$$\leq \delta \|z\|_1 + \sum_{i \notin h_t^{-1}(T)} |\Delta^{s,t}(i) - z(i)| + \sum_{i \in h_t^{-1}(T) \setminus \mathcal{G}} |\Delta^{s,t}(i) - z(i)| \quad (30)$$

$$= \delta \|z\|_1 + \sum_{i \notin h_t^{-1}(T)} |z(i)| + \sum_{i \in h_t^{-1}(T) \setminus \mathcal{G}} |\Delta^{s,t}(i) - z(i)|$$

$$\leq \delta \|z\|_1 + \sum_{i \notin h_t^{-1}(T)} |z(i)| + \sum_{i \in h_t^{-1}(T) \setminus \mathcal{G}} (|\Delta^{s,t}(i)| + |z(i)|)$$

$$= \delta \|z\|_1 + \sum_{i \notin h_t^{-1}(T) \cap \mathcal{G}} |z(i)| + \sum_{i \in h_t^{-1}(T) \setminus \mathcal{G}} (|\Delta^{s,t}(i)| + |z(i)|)$$

$$\leq (\delta + \beta) \|z\|_1 + \sum_{i \in h_t^{-1}(T) \setminus \mathcal{G}} |\Delta^{s,t}(i)| \quad (31)$$

In the above, (30) uses (28) and (31) uses Lemma 30. Now, for each $i \in h_t^{-1}(T) \setminus \mathcal{G}$ such that $|\Delta^{s,t}(i)| \neq 0$, the algorithm by construction sets $\Delta^{s,t}(i) = y^{s,t,0}(h_t(i)) = \sum_{j \in h_t^{-1}(h_t(i))} z(j)$. Observe that in this case, we must have $h_t^{-1}(h_t(i)) \cap \mathcal{G} = \emptyset$. This is because if there is some $i' \in h_t^{-1}(h_t(i)) \cap \mathcal{G}$, the for loop in procedure ESTIMATE upon processing the element $h_t(i) = h_t(i')$ in the set T would call SEARCH($h_t(i'), t, s$) which would return i' rather than i according to Proposition 28 (since $i' \in \mathcal{G}$), making the algorithm estimate the value of $\Delta^{s,t}(i)$ and leave $\Delta^{s,t}(i')$ zero. Therefore,

$$\sum_{i \in h_t^{-1}(T) \setminus \mathcal{G}} |\Delta^{s,t}(i)| \leq \sum_{i \notin \mathcal{G}} |z(i)| \leq \beta \|z\|_1,$$

the last inequality being true according to Lemma 29. Plugging this result back into (30), we get that

$$\|x - (x^s + \Delta^{s,t})\|_1 = \|\Delta^{s,t} - z\|_1 \leq (\delta + 2\beta) \|z\|_1 = (\delta + 2\beta) \|x - x^s\|_1.$$

The proof of Lemma 21 is now complete by choosing δ and ϵ (thus β) small enough constants so that $\delta + 2\beta \leq \gamma$. □

5 Speeding up the algorithm using randomness

Although this work focuses on deterministic algorithms for sparse Hadamard transform, in this section we show that our algorithm in Figure 1 can be significantly sped up by using randomness (yet preserving non-adaptivity).

The main intuition is straightforward: In the **for** loop of Line 7, in fact most choices of t turn out to be equally useful for improving the approximation error of the algorithm. Thus, instead of

trying all possibilities of t , it suffices to just pick one random choice. However, since the error ϵ of the condenser is a constant, the “success probability” of picking a random t has to be amplified. This can be achieved by either 1) Designing the error of condenser small enough to begin with; or, 2) Picking a few independent random choices of t and trying each such choice, and then estimating the choice that leads to the best improvements. It turns out that the former option can be rather wasteful in that it may increase the output length of the condenser (and subsequently, the overall sample complexity and running time) by a substantial factor. In this section, we pursue the second approach which leads to nearly optimal results.

In this section, we consider a revised algorithm that

- Instead of looping over all choices of t in Line 7 of procedure RECOVER, just runs the loop over a few random choices of t .
- In Line 17, instead of minimizing $\|Mx - Mx^s\|_1$, performs the minimization with respect to a randomly sub-sampled submatrix of M obtained from restriction M to a few random and independent choices of t .

The above randomized version of procedure RECOVER is called procedure RECOVER' in the sequel, and is depicted in Figure 2. The algorithm chooses an integer parameter q which determines the needed number of samples for t . In the algorithm, we use the notation $M^{\mathcal{T}}$, where $\mathcal{T} \subseteq [D]$ is a multi-set, to denote the $|\mathcal{T}|2^r \times N$ matrix obtained by stacking matrices M^t for all $t \in \mathcal{T}$ on top of one another. Note that the algorithm repeatedly uses fresh samples of t as it proceeds. This eliminates possible dependencies as the algorithm proceeds and simplifies the analysis.

More formally, our goal in this section is to prove the following randomized analogue of Theorem 19. Since the running time of the randomized algorithm may in general be less than the sketch length ($2^r D(n+1)$), we assume that the randomized algorithm receives the sketch implicitly and has query access to this vector.

Theorem 31. *(Analogue of Theorem 19) There are absolute constants $c > 0$ and $\epsilon' > 0$ such that the following holds. Let k, n ($k \leq n$) be positive integer parameters, and suppose there exists a function $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$ (where $r \leq n$) computable in time $f(n)$ (where $f(n) = \Omega(n)$) which is an explicit $(\log(4k), \epsilon')$ -lossless condenser. Let M be the adjacency matrix of the bipartite graph associated with h and B be the bit-selection matrix with n rows and $N := 2^n$ columns. Then, there is a randomized algorithm that, given k, n , parameters $\eta, \nu > 0$, and query access to the vectors Mx and $(M \otimes B)x$ for some $x \in \mathbb{R}^N$ (which is not given to the algorithm), computes a k -sparse estimate \tilde{x} such that, with probability at least $1 - \eta$ over the random coin tosses of the algorithm,*

$$\|\tilde{x} - x\|_1 \leq c \cdot \|x - H_k(x)\|_1 + \nu \|x\|_1.$$

Moreover, execution of the algorithm takes $O(2^r \cdot \log(\log(1/\nu)/\eta) \cdot \log(1/\nu)f(n))$ arithmetic operations in the worst case.

```

RECOVER'(y, s0, q)
1  s = 0.
2  Let B1, ..., Bn ∈ {0, 1}1×N be the rows of the bit selection matrix B.
3  Initialize x0 ∈ ℝN as x0 = 0.
4  for (t, b, j) ∈ [D] × {0, ..., n} × ℱ2r
5      y0,t,b(j) = y(j, t, b).
6  repeat
7      Let ℱs ⊆ [D] be a multiset of q uniformly and independently random elements.
8      for t ∈ ℱs
9          ys,t,0 = Mt · (x - xs) ∈ ℝ2r.
10         for b ∈ [n]
11             ys,t,b = (Mt ⊗ Bb) · (x - xs) ∈ ℝ2r.
12             Δs,t = ESTIMATE(t, s).
13         Let ℱ's ⊆ [D] be a multiset of q uniformly and independently random elements.
14         Let t0 be the choice of t ∈ ℱ's that minimizes ||Mℱ'sx - Mℱ's(xs + Δs,t)||1.
15         xs+1 = Hk(xs + Δs,t0).
16         s = s + 1.
17 until s = s0.
18 Let ℱ'' ⊆ [D] be a multiset of q uniformly and independently random elements.
19 Set x* to be the choice of xs (for s = 0, ..., s0) that minimizes ||Mℱ''x - Mℱ''xs||1.
20 return x*.

```

Figure 2: Pseudo-code for the randomized version of the algorithm RECOVER. The algorithm receives y implicitly and only queries y at a subset of the positions. The additional integer parameter q is set up by the analysis.

Proof of Theorem 31 is deferred to Section 5.1. In the sequel, we instantiate this theorem for use in sparse Hadamard transform application. Specifically, we consider the additional effect on the running time incurred by the initial sampling stage; that is, computation of the input to the algorithm in Figure 2 from the information provided in $\hat{x} = \text{DHT}(x)$.

First, notice that all the coin tosses of the algorithm in Figure 2 (namely, the sets $\mathcal{T}^0, \dots, \mathcal{T}^{s_0-1}, \mathcal{T}'^0, \dots, \mathcal{T}'^{s_0-1}$, and \mathcal{T}'') can be performed when the algorithm starts, due to the fact that each random sample $t \in [D]$ is distributed uniformly and independently of the algorithm's input and other random choices. Therefore, the sampling stage needs to compute $M^t x$ and $(M^t \otimes B)x$ for all the $(2s_0 + 1)q$ random choice of t made by the algorithm.

For $t \in [D]$, let V_t be the $(n - r)$ -dimensional subspace of \mathbb{F}_2^n which is the kernel of the linear function h_t . Moreover, let V_t^\perp and W_t respectively denote the dual and complement of V_t (as in Lemma 15). As discussed in the proof of Theorem 20, for each $t \in [D]$, we can use Lemma 9 to compute $M^t x$ from query access to $\hat{x} = \text{DHT}(x)$ at $O(2^r r)$ points and using $O(2^r r n)$ arithmetic operations, assuming that a basis for V_t^\perp and W_t is known. Similarly, $(M^t \otimes B)x$ may be computed using $O(2^r r n^2)$ operations and by querying \hat{x} at $O(2^r r n)$ points.

Computation of a basis for V_t^\perp and W_t for a given t can in general be performed⁵ using Gaussian elimination in time $O(n^3)$. Therefore, the additional time for the pre-processing needed for computation of such bases for all choices of t picked by the algorithm is $O(qs_0 n^4)$.

Altogether, we see that the pre-processing stage in total takes

$$O(qs_0(2^r r + n^2)n^2) = O(\log(\log(1/\nu)/\eta) \cdot \log(1/\nu) \cdot (2^r r + n^2)n^2)$$

arithmetic operations.

Finally we instantiate the randomized sparse DHT algorithm using Theorem 31, pre-processing discussed above, and the lossless condensers constructed by the Leftover Hash Lemma (Lemma 41). As for the linear family of hash functions required by the Leftover Hash Lemma, we use the linear family \mathcal{H}_{lin} which is defined in Section 4.3. Informally, a hash function in this family corresponds to an element β of the finite field \mathbb{F}_{2^n} . Given an input x , the function interprets x as an element of \mathbb{F}_{2^n} and then outputs the bit representation of $\beta \cdot x$ truncated to the desired r bits. We remark that the condenser obtained in this way is computable in time $f(n) = O(n \log n)$ using the FFT-based multiplication algorithm over \mathbb{F}_{2^n} . Simplicity of this condenser and mild hidden constants in the asymptotics is particularly appealing for practical applications.

Recall that for the Leftover Hash Lemma, we have $2^r = O(k/\epsilon'^2) = O(k)$, which is asymptotically optimal. Using this in the above running time estimate, we see that the final randomized

⁵ For structured transformations it is possible to do better (see [14]). This is the case for the specific case of Leftover Hash Lemma that we will later use in this section. However, we do not attempt to optimize this computation since it only incurs an additive poly-logarithmic factor in N which affects the asymptotic running time only for very small k .

version of the sparse DHT algorithm performs

$$O(\log(\log(1/\nu)/\eta) \cdot \log(1/\nu) \cdot (k \log k + n^2) \cdot n^2)$$

arithmetic operations in the worst case to succeed with probability at least $1 - \eta$.

Finally, by recalling that an algorithm that computes an estimate satisfying (4) can be transformed into one satisfying (2) using Proposition 23, we conclude the final result of this section (and Theorem 3) that follows from the above discussion combined with Theorem 31.

Corollary 32 (Generalization of Theorem 3). *There is a randomized algorithm that, given integers k, n (where $k \leq n$), parameters $\eta > 0$ and $\nu > 0$, and (non-adaptive) query access to any $\hat{x} \in \mathbb{R}^N$ (where $N := 2^n$), outputs $\tilde{x} \in \mathbb{R}^N$ that, with probability at least $1 - \eta$ over the internal random coin tosses of the algorithm, satisfies*

$$\|\tilde{x} - x\|_1 \leq c\|x - H_k(x)\|_1 + \nu\|x\|_1,$$

for some absolute constant $c > 0$ and $\hat{x} = \text{DHT}(x)$. Moreover, the algorithm performs a worst-case

$$O(\log(\log(1/\nu)/\eta) \cdot \log(1/\nu) \cdot (k \log k + n^2) \cdot n^2)$$

arithmetic operations⁶ to compute \tilde{x} . Finally, when each coefficient of \hat{x} takes $O(n)$ bits to represent, the algorithm can be set up to output \tilde{x} satisfying

$$\|\tilde{x} - x\|_1 \leq c\|x - H_k(x)\|_1,$$

using $O(\log(n/\eta) \cdot (k \log k + n^2) \cdot n^3)$ arithmetic operations in the worst case. In particular, when $\eta = 1/n^{O(1)}$ and $k = \Omega(n^2) = \Omega(\log^2 N)$, the algorithm runs in worst case time $O(kn^3(\log k)(\log n)) = \tilde{O}(k(\log N)^3)$. \square

5.1 Proof of Theorem 31

The proof is quite similar to the proof of Theorem 19, and therefore, in this section we describe the necessary modifications to the proof of Theorem 19 which lead to the conclusion of Theorem 31.

5.1.1 Correctness analysis of the randomized sparse recovery algorithm

Similar to the proof of Theorem 19, our goal is to set up the randomized algorithm so that, given arbitrarily small parameters $\nu, \eta > 0$, it outputs a k -sparse estimate $\tilde{x} \in \mathbb{R}^N$ that at least with probability $1 - \eta$ (over the random coin tosses of the algorithm) satisfies (4), recalled below, for an absolute constant $C > 0$:

$$\|\tilde{x} - x\|_1 \leq C\|x - H_k(x)\|_1 + \nu\|x\|_1,$$

⁶We remark that the running time estimate counts $O(n)$ operations for indexing; that is, looking for $\hat{x}(i)$ for an index $i \in [N]$, and one operation for writing down the result.

As in the proof of Theorem 19 and using Proposition 23, once we have such a guarantee for some $\nu = \Theta(1/(NL))$, assuming that x has integer coordinates in range $[-L, +L]$ and by rounding the final result vector to the nearest integer vector we get the guarantee in (2).

We will also use the following “error amplification” result that can be simply proved using standard concentration results.

Lemma 33. *Suppose $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$ is a (κ, ϵ) -lossless condenser. For any set $S \subseteq \mathbb{F}_2^n$ where $|S| \leq 2^\kappa$ the following holds. Let $q \in \mathbb{N}$ be a parameter and t_1, \dots, t_q be drawn uniformly and independently at random. Let $h': \mathbb{F}_2^n \times [q] \rightarrow \mathbb{F}_2^r$ be defined as $h'(x, j) := h(x, t_j)$, and G be the bipartite graph associated with h' . Let $T \subseteq \mathbb{F}_2^r$ be the neighborhood of the set of left vertices of G defined by S . Then, with probability at least $1 - \exp(-\epsilon^2 q/4)$ (over the randomness of t_1, \dots, t_q), we have $|T| \geq (1 - 2\epsilon)q|S|$.*

Proof. Let G^0 be the bipartite graph associated with h , with N left vertices and $D2^r$ right vertices, and for each $t \in [D]$, denote by G^t the bipartite graph associated with h_t , each having N left vertices and 2^r right vertices. Recall that G^0 contains the union of the edge set of G^1, \dots, G^D (with shared left vertex set $[N]$ and disjoint right vertex sets), and that G contains the union of the edge set of G^{t_1}, \dots, G^{t_q} . Let T^0 be the set of right neighbors of S in G^0 . Similarly, let T^t ($t \in [D]$) be the set of right neighbors of S in G^t .

Since h is a lossless condenser, we know that $|T^0| \geq (1 - \epsilon)D|S|$. For $i \in [q]$, let $X_i \in [0, 1]$ be such that $|T^i| = (1 - X_i)|S|$, and define $X := X_1 + \dots + X_q$. By an averaging argument, we see that $\mathbb{E}[X_i] \leq \epsilon$. Moreover, the random variables X_1, \dots, X_q are independent. Therefore, by a Chernoff bound,

$$\Pr[X > 2\epsilon q] \leq \exp(-\epsilon^2 q/4).$$

The claim follows after observing that $|T| = (q - X)|S|$ (since the graph G is composed of the union of G^1, \dots, G^q with disjoint right vertex sets). \square

Note that the above lemma requires the set S to be determined and fixed *before* the random seeds t_1, \dots, t_q are drawn. Thus the lemma makes no claim about the case where an adversary chooses S based on the outcomes of the random seeds.

In the sequel, we set the error of the randomness condenser (that we shall denote by ϵ') to be $\epsilon' \leq \epsilon/2$, where ϵ is the constant from Theorem 20.

We observe that the result reported in Lemma 25 only uses the expansion property of the underlying bipartite graph with respect to the particular support of the vector w . Thus, assuming that the conclusion of Lemma 33 holds for the set S in the lemma set to be the support of a k' -sparse vector w (where in our case $k' = 4k$), we may use the conclusion of Lemma 25 that, for some $t \in \{t_1, \dots, t_q\}$,

$$\sum_{(i,j) \in E^t \setminus \text{First}(G,w)} |w_i| \leq \epsilon \|w\|_1.$$

Using the above observation, we can deduce an analogue of the result of Lemma 21 for the randomized case by noting that the result in Lemma 21 holds as long as the set W in the proof of this lemma satisfies (21). Since the choice of W only depends on the previous iterations of the algorithm; that is the algorithm's input and random coin tosses determining $\mathcal{T}^0, \dots, \mathcal{T}^{s-1}$, we can use Lemma 33 to ensure that (21) holds with high probability. In other words, we can rephrase Lemma 21 as follows.

Lemma 34. *(Analogue of Lemma 21) For every constant $\gamma > 0$, there is an ϵ_0 and $C > 0$ only depending on γ such that if $\epsilon' \leq \epsilon_0$ the following holds. Suppose that for some s ,*

$$\|x - x^s\|_1 > C\|x - H_k(x)\|_1. \quad (32)$$

Then, with probability at least $1 - \exp(-\epsilon'^2 q/4)$, there is a $t \in \mathcal{T}^s$ such that

$$\|x - (x^s + \Delta^{s,t})\|_1 \leq \gamma\|x - x^s\|_1.$$

□

Declare a *bad event* at stage s if we have the condition $\|x - x^s\|_1 > C\|x - H_k(x)\|_1$ however the conclusion of the lemma does not hold because of unfortunate random coin tosses by the algorithm. By a union bound, we see that the probability that any such bad event happens throughout the algorithm is at most $s_0 \exp(-\epsilon'^2 q/4)$.

Next we show an analogue of Proposition 26 for the randomized algorithm.

Proposition 35. *Let $x', x'' \in \mathbb{R}^N$ be fixed $(3k)$ -sparse vectors and \mathcal{T} be a multi-set of q elements in $[D]$ chosen uniformly and independently at random. Moreover, assume*

$$\|M^{\mathcal{T}}(x - x')\|_1 \leq \|M^{\mathcal{T}}(x - x'')\|_1.$$

Then, with probability at least $1 - 2 \exp(-\epsilon'^2 q/4)$ over the choice of \mathcal{T} , we have

$$\|x - x'\|_1 \leq \left(1 + \frac{3 + C_0\epsilon}{1 - C_0\epsilon}\right)\|x - H_k(x)\|_1 + \frac{1 + C_0\epsilon}{1 - C_0\epsilon} \cdot \|x - x''\|_1$$

where C_0 is the constant in Theorem 5. In particular when $C_0\epsilon \leq 1/2$, we have (with the above-mentioned probability bound)

$$\|x - x'\|_1 \leq 8\|x - H_k(x)\|_1 + 3\|x - x''\|_1.$$

Proof. Proof is the same as the original proof of Proposition 26. The only difference is observing that the argument is valid provided that the RIP-1 condition holds for two particular $(4k)$ -sparse vectors $H_k(x) - x''$ and $H_k(x) - x'$ (as used in (9) and (13)). On the other hand, the proof of Theorem 5 only uses the expansion property of the underlying expander graph for the particular support of the sparse vector being considered, and holds as long as the expansion is satisfied for this particular choice. By applying Lemma 33 twice on the supports of $H_k(x) - x''$ and $H_k(x) - x'$, and taking a union bound, we see that the required expansion is available with probability at least $1 - 2 \exp(-\epsilon'^2 q/4)$, and thus the claim follows. □

Using the above tool, we can now show an analogue of Corollary 27; that is,

Corollary 36. *For every constant $\gamma_0 > 0$, there is an ϵ_0 only depending on γ_0 such that if $\epsilon \leq \epsilon_0$ the following holds. Assume condition (32) of Lemma 34 holds. Then, with probability at least $1 - 2 \exp(-\epsilon'^2 q/4)$ over the choice of \mathcal{T}'^s , we have*

$$\|x - x^{s+1}\|_1 \leq \gamma_0 \|x - x^s\|_1.$$

Proof. The proof is essentially the same as the proof of Corollary 27. The only difference is that instead of $\|Mx - M(x^s + \Delta^{s,t})\|_1$, the quantity $\|M^{\mathcal{T}'^s} x - M^{\mathcal{T}'^s}(x^s + \Delta^{s,t})\|_1$ that is used in the randomized algorithm is considered, and Proposition 35 is used instead of Proposition 26. In order to ensure that we can use Proposition 35, we use the fact that particular choices of the vectors x' and x'' that we instantiate Proposition 35 with (respectively, the vectors $x^s + \Delta^{s,t_0}$ and $x^s + \Delta^{s,t}$ in the proof of Corollary 27) only depend on the algorithm's input and random coin tosses determining $\mathcal{T}^0, \dots, \mathcal{T}^s$ and $\mathcal{T}'^0, \dots, \mathcal{T}'^{s-1}$ and not on \mathcal{T}'^s . \square

Again, declare a *bad event* at stage s if we have the condition $\|x - x^s\|_1 > C\|x - H_k(x)\|_1$ however the conclusion of Corollary 36 does not hold because of unfortunate coin tosses over the choice of \mathcal{T}'^s . Same as before, by a union bound we can see that the probability that any such bad event happens throughout the algorithm is at most $2s_0 \exp(-\epsilon'^2 q/4)$.

Since the initial approximation is $x^0 = 0$ (with error at most $\|x\|_1$), assuming $\gamma_0 \leq 1/2$, we have that for some $s \leq \log(1/\nu)$ the condition (4) is satisfied provided that a bad event does not happen in the first s iterations. By the above union bounds, this is the case with probability at least $1 - 3s_0 \exp(-\epsilon'^2 q/4)$.

Let x^* be the estimate computed in Line 17 of procedure RECOVER'. We can conclude the analysis in a similar way to the proof of Theorem 19 by one final use of Proposition 35 as follows. By Proposition 35, assuming no bad event ever occurs, with probability at least $1 - 2 \exp(-\epsilon'^2 q/4)$ we see that

$$\|x - x^*\|_1 \leq 8\|x - H_k(x)\|_1 + 3\|x - x^s\|_1 \leq C' \cdot \|x - H_k(x)\|_1 + \nu\|x\|_1, \quad (33)$$

where we define $C' := 3C + 8$.

Altogether, by a final union bound we conclude that the desired (33) holds with probability at least $1 - \eta$ for some choice of $q = O(\log(s_0/\eta)/\epsilon'^2) = O(\log(s_0/\eta))$.

5.1.2 Analysis of the running time of the randomized sparse recovery algorithm

The analysis of the running time of procedure RECOVER' in Figure 2 is similar to Section 4.3. As written in Figure 2, the algorithm may not achieve the promised running time since the sketch length may itself be larger than the desired running time. Thus we point out that the sketch is implicitly given to the algorithm as an oracle and the algorithm queries the sketch as needed

throughout its execution. Same holds for the initialization step in Line 4 of procedure RECOVER', which need not be performed explicitly by the algorithm.

In order to optimize time, the algorithm stores vectors in sparse representation; i.e., maintaining support of the vector along with the values at corresponding positions.

As discussed in Section 4.3, each invocation of procedure SEARCH takes $O(n)$ arithmetic operations, and procedure ESTIMATE takes $O(r2^r + kf(n)) = O(2^r f(n))$ operations (using naive sorting to find the largest coefficients and noting that $2^r \geq k$ and $f(n) = \Omega(n) = \Omega(r)$).

We observe that for every k -sparse $w \in \mathbb{R}^N$, and $t \in [D]$, computing the multiplication $M^t \cdot k$ (which itself would be a k -sparse vector) takes $O(kf(n))$ operations (k invocations of the condenser function, once for each nonzero entry of w , each time adding the corresponding entry of w to the correct position in the result vector). Note that the indexing time for updating an entry of the resulting vector is logarithmic in its length, which would be $r \leq n$ and thus the required indexing time is absorbed into the above asymptotic since $f(n) = \Omega(n)$. Moreover, we observe that without an effect in the above running time, we can in fact compute $(M^t \otimes B) \cdot w$; since for each $i \in [N]$ on the support of w , the corresponding $w(i)$ is added to a subset of the copies of M^t depending on the bit representation of i and thus the additional computation per entry on the support of w is $O(n)$, which is absorbed in the time $f(n) = \Omega(n)$ needed to compute the condenser function. Altogether we see that computing $(M^t \otimes B) \cdot k$ can be done with $O(kf(n))$ arithmetic operations.

Since procedure RECOVER' loops q times instead of D times in each of the s_0 iterations, each iteration taking time $O(2^r f(n))$, we see that the algorithm requires $O(2^r q s_0 f(n))$ arithmetic operations in total. Now we can plug in the values of q and s_0 by the analysis in the previous section and upper bound the number of operations performed by the algorithm by

$$O(2^r \cdot \log(\log(1/\nu)/\eta) \cdot \log(1/\nu) f(n)).$$

This completes the running time analysis of the algorithm in Figure 2.

References

- [1] A. Akavia. Deterministic sparse Fourier approximation via fooling arithmetic progressions. In *in Proceedings of COLT 2010*, pages 381–393, 2010.
- [2] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, and M. Strauss. Combining geometry and combinatorics: a unified approach to sparse signal recovery. In *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*, 2008.
- [3] M. Cheraghchi. *Applications of Derandomization Theory in Coding*. PhD thesis, Swiss Federal Institute of Technology, Lausanne, Lausanne, Switzerland, 2010. (available online at http://eccc.hpi-web.de/static/books/Applications_of_Derandomization_Theory_in_Coding/).

- [4] S. Foucart and H. Rauhut. *A Mathematical Introduction to Compressive Sensing*. Springer, 2013.
- [5] A. Gilbert, P. Indyk, M. Iwen, and L. Schmidt. Recent developments in the sparse Fourier transform. *Signal Processing Magazine*, 31:91–100, 2014.
- [6] A. Gilbert, M. Strauss, and J. A. Tropp. A tutorial on fast Fourier sampling. *Signal Processing Magazine*, 2008.
- [7] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.
- [8] V. Guruswami, C. Umans, and S. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. *Journal of the ACM*, 56(4), 2009.
- [9] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly optimal sparse Fourier transform. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, pages 563–578, 2012.
- [10] R. Impagliazzo, L. Levin, and M. Luby. Pseudorandom generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 12–24, 1989.
- [11] P. Indyk. Faster algorithms for sparse Fourier transform. Available online at <http://www.cs.princeton.edu/~ynaamad/misc/fourier-princeton.pdf>, 2013.
- [12] M. Iwen. Improved approximation guarantees for sublinear-time Fourier algorithms. *Applied And Computational Harmonic Analysis*, 34:57–82, 2013.
- [13] M. A. Iwen. Combinatorial sublinear-time Fourier algorithms. *Foundations of Computational Mathematics*, 10:303–338, 2010.
- [14] T. Kailath and A. H. Sayed. *Fast Reliable Algorithms for Matrices with Structure (Advances in Design and Control)*. Society for Industrial and Applied Math (SIAM), 1999.
- [15] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, pages 455–464, 1991.
- [16] R. Scheibler, S. Haghhighatshoar, and M. Vetterli. A fast Hadamard transform for signals with sub-linear sparsity. *CoRR*, abs/1310.1803, 2013.
- [17] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54:435–447, 1990.

[18] A. Ta-Shma, C. Umans, and D. Zuckerman. Lossless condensers, unbalanced expanders, and extractors. In *Proceedings of the 33th STOC*, pages 143–152, 2001.

A Proof of Theorem 12 (construction of the lossless condenser)

In this appendix, we include a proof of Theorem 12 from [3]. The first step is to recall the original framework for construction of lossless condensers in [8] which is depicted in Construction 1. The construction is defined with respect to a prime power alphabet size q and integer parameter $u > 1$.

- *Given:* A random sample $X \sim \mathcal{X}$, where \mathcal{X} is a distribution on \mathbb{F}_q^n with min-entropy at least κ , and a uniformly distributed random seed $Z \sim \mathcal{U}_{\mathbb{F}_q}$ over \mathbb{F}_q .
- *Output:* A vector $C(X, Z)$ of length ℓ over \mathbb{F}_q .
- *Construction:* Take any irreducible univariate polynomial g of degree n over \mathbb{F}_q , and interpret the input X as the coefficient vector of a random univariate polynomial F of degree $n - 1$ over \mathbb{F}_q . Then, for an integer parameter u , the output is given by

$$C(X, Z) := (F(Z), F_1(Z), \dots, F_{\ell-1}(Z)),$$

where we have used the shorthand $F_i := F^{u^i} \bmod g$.

Construction 1: Guruswami-Umans-Vadhan’s Condenser $C: \mathbb{F}_q^n \times \mathbb{F}_q \rightarrow \mathbb{F}_q^\ell$.

The following key result about Construction 1 is proved in [8]:

Theorem 37. [8] *For any $\kappa > 0$, the mapping defined in Construction 1 is a (κ, ϵ) lossless condenser with error $\epsilon := (n - 1)(u - 1)\ell/q$, provided that $\ell \geq \kappa/\log u$.*

By a careful choice of the parameters, the condenser can be made linear as observed by Cheraghchi [3]. We quote this result, which is a restatement of Theorem 12, below.

Corollary 38. [3] *Let p be a fixed prime power and $\alpha > 0$ be an arbitrary constant. Then, for parameters $n \in \mathbb{N}$, $\kappa \leq n \log p$, and $\epsilon > 0$, there is an explicit (κ, ϵ) -lossless condenser $h: \mathbb{F}_p^n \times \{0, 1\}^d \rightarrow \mathbb{F}_p^r$ with $d \leq (1 + 1/\alpha)(\log(n\kappa/\epsilon) + O(1))$ and output length satisfying $r \log p \leq d + (1 + \alpha)\kappa$. Moreover, h is a linear function (over \mathbb{F}_p) for every fixed choice of the second parameter.*

Proof. We set up the parameters of the condenser C given by Construction 1 and apply Theorem 37. The range of the parameters is mostly similar to what chosen in the original result of Guruswami et al. [8].

Letting $u_0 := (2p^2 n \kappa / \epsilon)^{1/\alpha}$, we take u to be an integer power of p in range $[u_0, pu_0]$. Also, let $\ell := \lceil \kappa / \log u \rceil$ so that the condition $\ell \geq \kappa / \log u$ required by Theorem 37 is satisfied. Finally, let $q_0 := n u \ell / \epsilon$ and choose the field size q to be an integer power of p in range $[q_0, pq_0]$.

We choose the input length of the condenser C to be equal to n . Note that C is defined over \mathbb{F}_q , and we need a condenser over \mathbb{F}_p . Since q is a power of p , \mathbb{F}_p is a subfield of \mathbb{F}_q . For $x \in \mathbb{F}_p^n$ and $z \in \{0, 1\}^d$, let $y := C(x, z) \in \mathbb{F}_q^\ell$, where x is regarded as a vector over the extension \mathbb{F}_q of \mathbb{F}_p . We define the output of the condenser $h(x, z)$ to be the vector y regarded as a vector of length $\ell \log_p q$ over \mathbb{F}_p (by expanding each element of \mathbb{F}_q as a vector of length $\log_p q$ over \mathbb{F}_p). Clearly, h is a (κ, ϵ) -condenser if C is.

By Theorem 37, C is a lossless condenser with error upper bounded by

$$\frac{(n-1)(u-1)\ell}{q} \leq \frac{n\ell}{q_0} = \epsilon.$$

It remains to analyze the seed length d and the output length r of the condenser. For the output length of the condenser, we have

$$r \log p = \ell \log q \leq (1 + \kappa / \log u) \log q \leq d + \kappa(\log q) / (\log u),$$

where the last inequality is due to the fact that we have $d = \lceil \log q \rceil$. Thus in order to show the desired upper bound on the output length, it suffices to show that $\log q \leq (1 + \alpha) \log u_0$. We have

$$\log q \leq \log(pq_0) = \log(pn\ell/\epsilon) \leq \log u_0 + \log(p^2 n\ell/\epsilon)$$

and our task is reduced to showing that $p^2 n\ell/\epsilon \leq u_0^\alpha = 2p^2 n\kappa/\epsilon$. But this bound is obviously valid by the choice of $\ell \leq 1 + \kappa / \log u$.

Now, $d = \lceil \log q \rceil$ for which we have

$$\begin{aligned} d &\leq \log q + 1 \leq \log q_0 + O(1) \\ &\leq \log(nu_0\ell/\epsilon) + O(1) \\ &\leq \log(nu_0\kappa/\epsilon) + O(1) \\ &\leq \log(n\kappa/\epsilon) + \frac{1}{\alpha} \log(2p^2 n\kappa/\epsilon) \\ &\leq \left(1 + \frac{1}{\alpha}\right) (\log(n\kappa/\epsilon) + O(1)) \end{aligned}$$

as desired.

Since \mathbb{F}_q has a fixed characteristic, an efficient deterministic algorithm for representation and manipulation of the field elements is available [17] which implies that the condenser is polynomial-time computable and is thus explicit.

Moreover, since u is taken as an integer power of p and \mathbb{F}_q is an extension of \mathbb{F}_p , for any choice of polynomials $F, F', G \in \mathbb{F}_q[X]$, subfield elements $a, b \in \mathbb{F}_p$, and integer $i \geq 0$, we have

$$(aF + bF')^{u^i} \equiv aF^{u^i} + bF'^{u^i} \pmod{G},$$

meaning that raising a polynomial to power u^i is an \mathbb{F}_p -linear operation. Therefore, the mapping C that defines the condenser (Construction 1) is \mathbb{F}_p -linear for every fixed seed. This in turn implies that the final condenser h is linear, as claimed. \square

B The Leftover Hash Lemma

Leftover Hash Lemma (first stated by Impagliazzo, Levin, and Luby [10]) is a basic and classical result in computational complexity which is normally stated in terms of randomness extractors. However, it is easy to observe that the same technique can be used to construct linear lossless condensers with optimal output length (albeit large seed length). In other words, the lemma shows that any universal family of hash functions can be turned into a linear extractor or lossless condenser. For completeness, in this section we include a proof of this fact.

Definition 39. A family of functions $\mathcal{H} = \{h_1, \dots, h_D\}$ where $h_t: \{0, 1\}^n \rightarrow \{0, 1\}^r$ for $t = 1, \dots, D$ is called *universal* if, for every fixed choice of $x, x' \in \{0, 1\}^n$ such that $x \neq x'$ and a uniformly random $t \in [D] := \{1, \dots, D\}$ we have

$$\Pr_t[h_t(x) = h_t(x')] \leq 2^{-r}.$$

One of the basic examples of universal hash families is what we call *the linear family*, defined as follows. Consider an arbitrary isomorphism $\varphi: \mathbb{F}_2^n \rightarrow \mathbb{F}_{2^n}$ between the vector space \mathbb{F}_2^n and the extension field \mathbb{F}_{2^n} , and let $0 < r \leq n$ be an arbitrary integer. The linear family \mathcal{H}_{lin} is the set $\{h_\beta: \beta \in \mathbb{F}_{2^n}\}$ of size 2^n that contains a function for each element of the extension field \mathbb{F}_{2^n} . For each β , the mapping h_β is given by

$$h_\beta(x) := (y_1, \dots, y_r), \text{ where } (y_1, \dots, y_n) := \varphi^{-1}(\beta \cdot \varphi(x)).$$

Observe that each function h_β can be expressed as a linear mapping from \mathbb{F}_2^n to \mathbb{F}_2^r . Below we show that this family is pairwise independent.

Proposition 40. *The linear family \mathcal{H}_{lin} defined above is universal.*

Proof. Let x, x' be different elements of \mathbb{F}_{2^n} . Consider the mapping $f: \mathbb{F}_{2^n} \rightarrow \mathbb{F}_2^r$ defined as

$$f(x) := (y_1, \dots, y_r), \text{ where } (y_1, \dots, y_n) := \varphi^{-1}(x),$$

which truncates the binary representation of a field element from \mathbb{F}_{2^n} to r bits. The probability we are trying to estimate in Definition 39 is, for a uniformly random $\beta \in \mathbb{F}_{2^n}$,

$$\Pr_{\beta \in \mathbb{F}_{2^n}} [f(\beta \cdot x) = f(\beta \cdot x')] = \Pr_{\beta \in \mathbb{F}_{2^n}} [f(\beta \cdot (x - x')) = 0].$$

But note that $x - x'$ is a nonzero element of \mathbb{F}_{2^n} , and thus, for a uniformly random β , the random variable βx is uniformly distributed on \mathbb{F}_{2^n} . It follows that

$$\Pr_{\beta \in \mathbb{F}_{2^n}} [f(\beta \cdot (x - x')) = 0] = 2^{-r},$$

implying that \mathcal{H}_{lin} is a universal family. □

Now we are ready to state and prove the Leftover Hash Lemma (focusing on the special case of lossless condensers).

Theorem 41. (*Leftover Hash Lemma*) Let $\mathcal{H} = \{h_t: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^r \mid t \in \mathbb{F}_2^d\}$ be a universal family of hash functions with D elements, and define the function $h: \mathbb{F}_2^n \times [D] \rightarrow \mathbb{F}_2^r$ as $h(x, t) := h_t(x)$. Then, for every κ, ϵ such that $r \geq \kappa + 2 \log(1/\epsilon)$, the function h is a (κ, ϵ) -lossless condenser. In particular, by choosing $\mathcal{H} = \mathcal{H}_{\text{lin}}$, it is possible to get explicit extractors and lossless condensers with $D = 2^n$.

Proof. Recall that by Definition 6 we need to show that for any distribution \mathcal{X} over \mathbb{F}_2^n and random variable X drawn from \mathcal{X} and independent random variable Z uniformly drawn from $[D]$, respectively, the distribution of $h(X, Z)$ is ϵ -close in statistical distance to a distribution with min-entropy at least κ . By a convexity argument, it suffices to show the claim when \mathcal{X} is the uniform distribution on a set $\text{supp}(\mathcal{X})$ of size $K := 2^\kappa$ (on the other hand, we only use the lemma for such distributions in this paper).

Define $R := 2^r$, $D := 2^d$, and let μ be any distribution uniformly supported on some set $\text{supp}(\mu) \subseteq [D] \times \mathbb{F}_2^r$ such that $[D] \times \text{supp}(\mathcal{X}) \subseteq \text{supp}(\mu)$, and denote by \mathcal{Y} the distribution of $(Z, h(X, Z))$ over $[D] \times \mathbb{F}_2^r$. We will first upper bound the ℓ_2 distance of the two distributions \mathcal{Y} and μ (i.e., the ℓ_2 difference of probability vectors defining the two distributions), that can be expressed as follows (we will use the notation $\mathcal{Y}(x)$ for the probability assigned to x by \mathcal{Y} , and similarly $\mu(x)$):

$$\begin{aligned}
\|\mathcal{Y} - \mu\|_2^2 &= \sum_{x \in [D] \times \mathbb{F}_2^r} (\mathcal{Y}(x) - \mu(x))^2 \\
&= \sum_x \mathcal{Y}(x)^2 + \sum_x \mu(x)^2 - 2 \sum_x \mathcal{Y}(x)\mu(x) \\
&\stackrel{\text{(a)}}{=} \sum_x \mathcal{Y}(x)^2 + \frac{1}{|\text{supp}(\mu)|} - \frac{2}{|\text{supp}(\mu)|} \sum_x \mathcal{Y}(x) \\
&= \sum_x \mathcal{Y}(x)^2 - \frac{1}{|\text{supp}(\mu)|}, \tag{34}
\end{aligned}$$

where (a) uses the fact that μ assigns probability $1/|\text{supp}(\mu)|$ to exactly $|\text{supp}(\mu)|$ elements of $[D] \times \mathbb{F}_2^r$ and zeros elsewhere.

Now observe that $\mathcal{Y}(x)^2$ is the probability that two independent samples drawn from \mathcal{Y} turn out to be equal to x , and thus, $\sum_x \mathcal{Y}(x)^2$ is the *collision probability* of two independent samples from \mathcal{Y} , which can be written as

$$\sum_x \mathcal{Y}(x)^2 = \Pr_{Z, Z', X, X'} [(Z, h(X, Z)) = (Z', h(X', Z'))],$$

where the random variables Z, Z' are uniformly and independently sampled from $[D]$ and X, X' are independently sampled from \mathcal{X} . We can rewrite the collision probability as

$$\sum_x \mathcal{Y}(x)^2 = \Pr[Z = Z'] \cdot \Pr[h(X, Z) = h(X', Z') \mid Z = Z']$$

$$\begin{aligned}
&= \frac{1}{D} \cdot \Pr_{Z, X, X'}[h_Z(X) = h_Z(X')] \\
&= \frac{1}{D} \cdot (\Pr[X = X'] + \frac{1}{K^2} \sum_{\substack{x, x' \in \text{supp}(\mathcal{X}) \\ x \neq x'}} \Pr_Z[h_Z(x) = h_Z(x')]) \\
&\stackrel{\text{(b)}}{\leq} \frac{1}{D} \cdot \left(\frac{1}{K} + \frac{1}{K^2} \sum_{\substack{x, x' \in \text{supp}(\mathcal{X}) \\ x \neq x'}} \frac{1}{R} \right) \leq \frac{1}{DR} \cdot \left(1 + \frac{R}{K} \right),
\end{aligned}$$

where (b) uses the assumption that \mathcal{H} is a universal hash family. Plugging the bound in (34) implies that

$$\|\mathcal{Y} - \mu\|_2 \leq \frac{1}{\sqrt{DR}} \cdot \sqrt{1 - \frac{DR}{|\text{supp}(\mu)|} + \frac{R}{K}}.$$

Observe that both \mathcal{Y} and μ assign zero probabilities to elements of $[D] \times \mathbb{F}_2^r$ outside the support of μ . Thus using Cauchy-Schwarz on a domain of size $|\text{supp}(\mu)|$, the above bound implies that the statistical distance between \mathcal{Y} and μ is at most

$$\frac{1}{2} \|\mathcal{Y} - \mu\|_1 \leq \frac{1}{2} \cdot \sqrt{\frac{|\text{supp}(\mu)|}{DR}} \cdot \sqrt{1 - \frac{DR}{|\text{supp}(\mu)|} + \frac{R}{K}}. \quad (35)$$

Now, we specialize μ to any distribution that is uniformly supported on a set of size DK containing $\text{supp}(\mathcal{Y})$ (note that, since \mathcal{X} is assumed to be uniformly distributed on its support, \mathcal{Y} must have a support of size at most DK). Since $r \geq \kappa + 2 \log(1/\epsilon)$, we have $K = \epsilon^2 R$, and (35) implies that \mathcal{Y} and μ are ϵ -close (in fact, $(\epsilon/2)$ -close) in statistical distance. \square