Traversing the FFT Computation Tree for Dimension-Independent Sparse Fourier Transforms

Karl Bringmann
Saarland Uni. & MPI

Michael Kapralov
EPFL

Mikhail Makarov
EPFL

Vasileios Nakos
Saarland Uni. & MPI

Amir Yagudin
MIPT

Amir Zandieh
MPI

January 24, 2023

Abstract

We are interested in the well-studied Sparse Fourier transform problem, where one aims to quickly recover an approximately Fourier k-sparse domain vector $\widehat{x} \in \mathbb{C}^{n^d}$ from observing its time domain representation x. In the exact k-sparse case the best known dimension-independent algorithm runs in near cubic time in k and it is unclear whether a faster algorithm like in low dimensions is possible. Beyond that, all known approaches either suffer from an exponential dependence of their runtime on the dimension d or can only tolerate a trivial amount of noise. This is in sharp contrast with the classical FFT algorithm of Cooley and Tukey, which is stable and completely insensitive to the dimension of the input vector: its runtime is $O(N \log N)$ in any dimension d for $N = n^d$. Our work aims to address the above issues.

First, we provide a translation/reduction of the exactly k-sparse Sparse FT problem to a concrete tree exploration task which asks to recover k leaves in a full binary tree under certain exploration rules. Subsequently, we provide (a) an almost quadratic in k time algorithm for the latter task, and (b) evidence that obtaining a strongly subquadratic time for Sparse FT via this approach is likely to be impossible. We achieve the latter by proving a conditional quadratic time lower bound on sparse polynomial multipoint evaluation (the classical non-equispaced sparse Fourier transform problem) which is a core routine in the aforementioned translation. Thus, our results combined can be viewed as an almost complete understanding of this approach, which is the only known approach that yields sublinear time dimension-independent Sparse FT algorithms.

Subsequently, we provide a robustification of our algorithm, yielding a robust cubic time algorithm under bounded ℓ_2 noise. This requires proving new structural properties of the recently introduced adaptive aliasing filters combined with a variety of new techniques and ideas. Lastly, we provide a preliminary experimental evaluation comparing the runtime of our algorithm to FFTW and SFFT 2.0.

Contents

1 Introduction.

Computing the largest in magnitude Fourier coefficients of a function without computing all of its Fourier transform, or reconstructing a sparse vector/signal x from partial Fourier measurements are common and well-studied tasks across science and engineering, as they appear in a variety of disciplines. Possibly the earliest work on the topic was by Gaspard de Prony in 1795, who showed that any k-sparse vector can be efficiently reconstructed from its first $2k$ Discrete Fourier transform (DFT) coefficients. These ideas have been re-discovered/used both in the context of decoding BCH codes [Wol67], as well as in the context of computer algebra by Ben-Or and Tiwari [BOT88]. In the context of learning theory, and in particular learning decision trees, Kushilevitz and Mansour [KM93] devised an algorithm that detects the largest Fourier coefficients of a function defined over the Boolean hypercube, building upon [GL89]. The work of [AGS03] uses sparse Fourier transform techniques in cryptography, namely for proving hard-core predicates for one-way functions. In 2002, a sublinear-time efficient algorithm for learning the k largest DFT coefficients was proposed in [GGI+02]; this line of work has resulted in (near-)optimal algorithms [GMS05, HIKP12a, Kap16, Kap17] for the DFT case. In terms of its applications to signal processing and reconstruction, arguably the most prominent is the work of Candes, Donoho, Romberg, and Tao [Don06, CT06, CRT06], which has far-reaching applications in fields such as medical imaging and spectroscopy [LDSP08, KY11], and created the area of *compressed sensing*; the reader may consult the text [FR13] for a thorough view on the topic.

Formally, the Sparse Fourier Transform problem is the following. Given oracle access to a size N d-dimensional vector x, find a vector $\widehat{\chi}$ such that

$$\|\widehat{x} - \widehat{\chi}\|_p \leq C \cdot \min_{k\text{-sparse vectors } \widehat{z}} \|\widehat{x} - \widehat{z}\|_q,$$

where C is the approximation factor, and $\|\cdot\|_p, \|\cdot\|_q$ are norms. The number of oracle accesses to x shall be referred to as *sample complexity*. The most well studied case in the literature is the case where $C = 1 + \epsilon$ (or constant) and $p = q = 2$, referred to as the ℓ_2/ℓ_2 guarantee. Other well-studied cases are the so-called ℓ_∞/ℓ_2 guarantee, where $C = \frac{1}{\sqrt{k}}, p = \infty, q = 2$, as well as the ℓ_2/ℓ_1 guarantee, see [CT06, IK14, NSW19]. Our focus in this paper is the ℓ_2/ℓ_2 guarantee. Frequently, the k largest in magnitude coordinates of $\widehat{x}$ are referred to as the *head* of the signal, while all the other coordinates are referred to as the *tail* of the signal, or as *noise*. With this vocabulary, the ℓ_2/ℓ_2 guarantee asks to recover the head of $\widehat{x}$ with error up to $(1 + \epsilon)$ times the noise level.

The research on the topic, especially over the last fifteen years, has been extensive [KM93, LMN93, BFJ+94, Man94, Man95, GGI+02, GMS05, CT06, IGS07, Iwe10, Aka10, CGV13, HIKP12a, HIKP12b, BCG+12, PR13, IKP14, PR14, Bou14, IK14, OPR15, PS15, JENR15, CKPS16, HR16, Kap16, CKSZ17, Kap17, CI17, MZIC17, KVZ19, AZKK19, NSW19, OHR19, JLS20]. Our understanding of the sample complexity of this problem is quite good: we know that $O(k \operatorname{poly}(\log N))$ samples are sufficient for finding in time near linear in N a vector $\widehat{\chi}$ satisfying any of the aforementioned guarantees [CT06, HR16, NSW19]. Regarding the particularly interesting case of $d = 1$, the research effort of the community has produced time-efficient algorithms as well. The fastest algorithm, due to the celebrated work of Hassanieh, Indyk, Katabi, and Price [HIKP12a], runs in time $O(k \log(N/k) \log N)$ and achieves the same sample complexity as well. We know also how to achieve $O(k \log N)$ sample complexity and $O(k \operatorname{poly}(\log N))$ running time [Kap17]. On the other extreme, when $d = \log N$, i.e. in the case of the Walsh-Hadamard transform, almost optimal running time is known to be achievable, even deterministically [CI17].

Along with the running time, the sample complexity, and the error guarantee, of particular interest is also the sensitivity of the algorithm to the underlying field. When we are concerned

with Fourier transforms over $\mathbb{Z}_n^d$[1], this corresponds to the sensitivity to the dimension d. Indeed, virtually all Sparse Fourier transform algorithms have a running time that suffers from an *exponential* dependence on d (in particular $\log^{\Omega(d)} N$), and the techniques either in dimension $d = 1$ or $d = \log N$ heavily rely on the structure of the corresponding group. At the same time, given that the Cooley-Tukey FFT algorithm itself is completely dimension-independent, a natural question is whether this independence transfers also to the Sparse Fourier transform setting. Concretely, is the curse of dimensionality an inherent problem, or an artifact of previous techniques? A major practical motivation is that a quest for removing the curse of dimensionality can ultimately lead to new insights for designing empirical, efficient algorithms in dimensions $d = 3, 4$, which are mostly relevant in applications in NMR-spectroscopy and MRI imaging. Thus, an algorithm with better dependence on the d and k could thus be of practical importance as well.

A step towards dimension-independence was taken in [KVZ19], by giving a $O(k^3 \cdot \text{poly}(\log N))$-time algorithm which recovers *exactly* k-sparse signals in any dimension. Their approach is based on pruning the FFT computation graph, using a new tool called adaptive aliasing filters. However, the aforementioned algorithm had two disadvantages: i) the time was cubic and there was no evidence whether this was optimal under some reasonable assumption, and ii) was not able to go beyond the barrier of exactly k-sparse signals (or, noise level $\text{poly}(N)$ times smaller than the energy of the head). Somewhat relevant is an algorithm due to Mansour [Man95], which performs breadth-first search on the Cooley-Tukey FFT computation tree, and can get $\text{poly}(k)$ running time for exactly k-sparse signals, but pays an additional multiplicative *signal to noise ratio* factor for general signals [Man95]. We also mention a beautiful $O(k \cdot \text{poly}(\log N))$-time algorithm for exactly k-sparse signals from [GHI$^+$13], which requires a distributional assumption on the support of the input signal in Fourier domain and unfortunately suffers from the restriction $k = O(N^{1/d})$; already in dimension $d = O(\log N / \log \log N)$, this guarantees correctness only for $k \leq \text{poly}(\log N)$.

Our results. First, we translate the exactly k-Sparse FT problem using the machinery developed in [KVZ19] to a tree exploration problem that is accessible without any knowledge on Fourier transform. Our first main result is an almost complete understanding of this line of attack.

- The tree exploration task can be solved in almost quadratic time, and hence the exact k-Sparse FT problem can be solved in almost quadratic time. This shaves off almost a factor of k from the previous best sublinear-time, dimension-independent algorithm of [KVZ19].

- The quadratic time is most likely impenetrable by any explorative algorithm which successively peels off elements. That implies that overcoming this quadratic time barrier will likely require a major paradigm shift in Sparse FFT technology. This is based on a lower bound on sparse polynomial multipoint evaluation and is interesting in its own right as the problem is well-studied under the name of non-equispaced Fourier transform.

In the robust case, we obtain a quadratic sample complexity, sublinear-time, dimension-independent algorithm that recovers the head of the signal under bounded ℓ_2 noise, i.e. when every frequency in the head is larger than the energy of the tail. Even under this seemingly restricted noise model, designing an efficient algorithm turns out to be non-trivial, requiring a constellation of new techniques. Previous algorithms were either i) robust and dimension-independent but not sublinear-time [CT06, IK14, NSW19], ii) sublinear-time and robust but not dimension-independent [GMS05, HIKP12a, Kap16], or iii) sublinear-time and dimension-independent but not

---

[1]This is the case with the groups of interest in the Sparse FT literature. Furthermore, these are the groups on which the FFT algorithm of Cooley and Tukey operates. For general finite groups G, the fastest FT algorithm runs in time almost $|G|^{\omega/2}$ [Uma19], where ω is the matrix multiplication exponent.

Task	Result
⋆ Sparse Fourier Transform in the exact case Input: Integers n, d, k and $N = n^d$, and oracle access to a vector $x \in \mathbb{C}^{n^d}$ satisfying $\|\widehat{x}\|_0 \le k$. Question: Compute $\widehat{x}$.	Theorem 1
⋆ ℓ_2/ℓ_2 Sparse Fourier Transform Input: Integers n, d, k and $N = n^d$, parameter $\epsilon < 1$, and oracle access to a vector $x \in \mathbb{C}^{n^d}$. Question: Compute a vector $\widehat{\chi} \in \mathbb{C}^{n^d}$ such that $\|\widehat{x} - \widehat{\chi}\|_2 \le (1 + \epsilon)\|\widehat{x}_{-k}\|_2$.	Theorem 4
⋆ Non-Equispaced Fourier Transform Input: Integers n, d, parameter $\epsilon < 1$, two sets $F, T \subseteq [n]^d$ with $|F| = |T| = k$, and a vector $x \in \mathbb{C}^{n^d}$ supported on T. Question: Compute additive $\pm\epsilon\|\widehat{x}\|_2$ approximations to each of $\widehat{x}_{\boldsymbol{f}}$, for $\boldsymbol{f} \in F$.	Theorem 2
⋆ Sparse Polynomial Multipoint Evaluation Input: Integers n, k, parameter $\epsilon < 1$, a polynomial p of degree n and sparsity k, i.e. k non-zero coefficients, each of which is of magnitude 1, as well as points $a_1, a_2, \ldots, a_k \in \mathbb{C}^n$ of magnitude 1. Question: Compute additive $\pm\epsilon$ approximations to each of $p(a_i)$, for all $i = 1, 2, \ldots, k$.	Theorem 3
⋆ Orthogonal vectors, $\mathrm{OV}_{k,d}$ Input: $A, B \subseteq \{0, 1\}^d$, with $|A| = |B| = k$ Question: Determine whether there exists $a \in A, b \in B$ such that $\langle a, b \rangle = 0$.	

Figure 1: Computational tasks considered in this paper.

robust to any form of noise [KVZ19]. We also discuss all the barriers we have faced, including the barrier to handling noise of larger magnitude, in Section 5.3.

2   Computational Tasks and Formal Results Statement.

This section contains the computational tasks studied in this paper, our results, and a preparations section for the lower bound, namely Theorem 2. We will be concerned with N-length d-dimensional vectors $x : [n]^d \to \mathbb{C}$, where $N = n^d$ and n is a power of 2. Thus, N, n, d will remain unaltered throughout the paper. We will use the notation $[n]$ to denote the set of integer numbers $\{0, 1, \ldots, n-1\}$. We will use a **non-standard** notation $\widetilde{O}(f) = O(f \, \mathrm{poly}(\log N))$, where f is some parameter and N is the size of our underlying vector x. For a vector x, we denote $\|\widehat{x}\|_0 = \left| \left\{ \boldsymbol{f} \in [n]^d : \widehat{x}_{\boldsymbol{f}} \ne 0 \right\} \right|$, and $\widehat{x}_T$, for a set $T \subseteq [n]^d$, to be the vector that results from zeroing out every coordinate of x outside of T. We let $\widehat{x}_{-k}$ be the vector that occurs after zeroing out the top k coordinates in magnitude, breaking ties arbitrarily. All logarithms are base 2. For the algorithm we present, we shall assume exact arithmetic operations over $\mathbb{C}$ in unit time throughout the paper, although the analysis goes through with $\frac{1}{\mathrm{poly}(N)}$ precision as well.

     We start by summarizing the formal definitions of all relevant computational problems in Figure 1. With these definitions in place we can state our results as follows:

Theorem 1 (Almost-Quadratic Time Exact k-Sparse FFT)**.** [2]   *Given oracle access to $x : [n]^d \to \mathbb{C}$*

---
[2] proved as Theorem 11 in Section 3 and Section 9

3

with $\|\widehat{x}\|_0 \leq k$, we can find $\widehat{x}$ in deterministic time

$$\widetilde{O}\left(k^2 \cdot 2^{8\sqrt{\log k \cdot \log \log N}}\right).$$

We formally show that the exact Sparse FFT problem can be reduced to a tree exploration problem and show how to solve the tree exploration in almost quadratic time, and thus prove the above theorem, in Section 3 and Section 9 as Theorem 11.

Conjecture 1. *(Orthogonal Vectors Hypothesis(OVH) [Wil05, AWW14]) For every $\epsilon > 0$, there exists a constant $c \geq 1$ such that* $\mathrm{OV}_{k,d}$ *(see Figure 1) requires $\Omega(k^{2-\epsilon})$ time whenever $d \geq c \log k$.*

It is known that a collapse of the Orthogonal Vectors Hypothesis would have groundbreaking implications in algorithm design, see [GIKW19] and [ABDN18].

Theorem 2 (Lower Bound for Non-Equispaced Fourier Transform). [3] *Assume that for all $k < n$ and $\epsilon > 0$ there exists an algorithm that solves the Non-Equispaced Fourier Transform in time $O(k^{2-\delta}\operatorname{poly}(\log(n/\epsilon)))$ for some constant $\delta > 0$. Then the Orthogonal Vectors hypothesis fails.*

Proof outline: Given sets of vectors

$$A = \{a_0, a_1, \ldots, a_{k-1}\}, \ B = \{b_0, b_1, \ldots, b_{k-1}\} \subseteq \{0,1\}^d,$$

we build $|A|$ points in time domain and $|B|$ points in frequency domain as follows. We pick sufficiently large M, q, N (for details see Section 10) and define for $j \in [k]$:

$$t_j := \sum_{r \in [d]} a_j(r) \cdot M^{rq}, \qquad f_j := \sum_{r \in [d]} b_j(r) \cdot \frac{N}{M^{rq+1}},$$

Subsequently, we look at the indicator vector of the set $\{t_0, t_1, \ldots, t_{k-1}\}$, let it be x. Asking for the values $\widehat{x}_{f_0}, \widehat{x}_{f_1}, \ldots, \widehat{x}_{f_{k-1}}$ corresponds exactly to the non-equispaced Fourier transform problem. Using the aforementioned evaluations we show that it is possible to extract the values

$$V_{j,h} := \sum_{\ell \in [k]} \langle a_\ell, b_j \rangle^h, \ \text{for } j \in [n], h \in [d].$$

For a fixed j, the values of $V_{j,h}$ can be expresed in terms of $Z_r := |\{\ell \in [k] \mid \langle a_\ell, b_j \rangle = r\}|$, via multiplication by a $d \times d$ Vandermonde matrix. Since the entries involved in this matrix and $V_{j,h}$ have $\operatorname{poly}(d, \log k)$ bits, we can then solve for Z_0 in $\operatorname{poly}(d, \log k)$ time, where Z_0 corresponds to the number of vectors $a \in A$ which are orthogonal to b_j. Repeating this over all $j \in [k]$ yields whether there exists a pair of orthogonal vectors.

Of course, the overview presented above completely ignores how we actually extract the values of $V_{j,h}$ from evaluations of the Fourier transform. This carefully exploits periodicity of complex exponentials – see Section 10 for more details. $\square$

A lower bound for sparse polynomial multipoint evaluation (Figure 1) also follows immediately.

Theorem 3 (Lower bound for Sparse Polynomial Multipoint Evaluation over $\mathbb{C}$). *Assume that for all $k < n$ and ϵ there exists an algorithm for sparse polynomial multipoint evaluation which runs in time $k^{2-\delta}\operatorname{poly}(\log(n/\epsilon))$. Then the Orthogonal Vector Hypothesis fails.*

---

[3]proved as Theorem 12 in Section 10

Significance of our lower bound for computational Fourier Transforms. Non-equispaced Fourier transform falls into a class of Fourier transforms referred to as *non-uniform*. These transforms are an extensively studied topic in signal processing and numerical analysis [GR87, FS03, GL04], with numerous applications in imaging, signal interpolation and solutions of differential equations; the reader may consult the texts [BM96, PST01, BM12].

To present our robust Sparse FFT results, we first quantify the notion of "bounded ℓ_2 noise".

High SNR model. A vector $x : [n]^d \rightarrow \mathbb{C}$ satisfies the k-high SNR assumption, if there exist vectors $w, \eta : [n]^d \rightarrow \mathbb{C}$ such that i) $\widehat{x} = \widehat{w} + \widehat{\eta}$, ii) $\text{supp}(\widehat{w}) \cap \text{supp}(\widehat{\eta}) = \varnothing$, iii) $|\text{supp}(\widehat{w})| \leq k$ and iv) $|\widehat{w}_f| \geq 3 \cdot \|\widehat{\eta}\|_2$[4], for every $f \in \text{supp}(\widehat{w})$.

Theorem 4 (Robust Sparse Fourier Transform with Near-quadratic Sample Complexity). [5] *Given oracle access to $x : [n]^d \rightarrow \mathbb{C}$ in the k-high SNR model and parameter $\epsilon > 0$, we can solve the ℓ_2/ℓ_2 Sparse Fourier Transform problem with high probability in N using*

$$m = \widetilde{O}\left(\frac{k^2}{\epsilon} + k^2 \cdot 2^{\Theta\left(\sqrt{\log k \cdot \log \log N}\right)}\right)$$

samples from x and $\widetilde{O}\left(\frac{k^3}{\epsilon}\right)$ running time.

This theorem is restated as Theorem 13 in Section 12 followed by the proof. We re-iterate that even though the noise model we consider might seem restrictive, it turns out to be quite challenging requiring whole new constellation of ideas. The starting point here is the observation that the adaptive aliasing filters constructed by [KVZ19] in fact form an orthogonal basis (see Lemma 14 in Section 11), and while the norms of individual filters in the family are not the same, the sum of their squares is equal to 1 at every point in time domain (see Lemma 16 in Section 11). The combination of these new facts allows us to argue noise stability of our algorithm in Section 12.

Additionally, in Section 5.3 we explain how we are led to consider this particular notion of high SNR regime, and why handling lower SNR is a hard barrier for algorithms which explore a pruned Cooley-Tukey FFT computation tree (which is also the only known class of algorithms that enables sublinear and dimension-independent recovery). The discrepancy between the running time and sample complexity provided by Theorem 4 is due to the fact that we used non-uniform Sparse Fourier Transform to subtract recovered frequencies from time domain in our algorithm, which requires quadratic time as per Theorem 2.

Experimental Evaluation. Lastly, we present our experimental evaluation in Section 14, where we compare our method to the highly optimized software packages such as FFTW and SFFT 2.0. The source code of our implementation is available at `https://bitbucket.org/michaelkapralov/sfft-experiments`.

3 Technical overview.

In this section we first present (in Section 3.1) a new near-isometry property of adaptive aliasing filters of [KVZ19], which underlies our robust high dimensional Sparse FFT algorithm. We then present (in Section 3.2) an abstract formulation of the Sparse Fourier transform algorithms which

---

[4]The constant 3 is arbitrary, and can be driven down to $(1 + \zeta)$, for any $\zeta > 0$.
[5]proved as Theorem 13 in Section 12

work based on these adaptive aliasing filters as an abstract *Tree Exploration Problem*. Such a formulation allows us to present the key ideas behind our quadratic time dimension-independent Sparse FFT algorithm in a concise way, avoiding unnecessary Fourier analytic formalism. The formal connection the tree exploration problem and the adaptive aliasing filter-based Sparse FFT is presented in Section 9.

3.1 A Near-Isometry Property of Adaptive Aliasing Filters.

Recall that given a signal $x : [n]^d \to \mathbb{C}$, the execution of the FFT algorithm produces a binary tree, referred to as T_N^{full}. The root of T_N^{full} corresponds to the universe $[n]^d$, while the children of the root correspond to $[n/2] \times [n]^{d-1}$; note that FFT recurses by peeling off the least significant bit. Every node v has a *label* $\boldsymbol{f}_v \in \mathbb{Z}_n^d$ associated to it, defined according to the following rules.

1. The root has label $\boldsymbol{f}_{\text{root}} = (\underbrace{0,0,\ldots,0}_{d \text{ entries}})$, and corresponds to the universe $[n]^d$.

2. The children $v_{\text{left}}, v_{\text{right}}$ of a node v which corresponds to the universe $[n/2^l] \times [n]^{d'}$, with $0 \le d' \le d-1, 0 \le l \le \log n - 1$, have the following properties. Both correspond to universe $[n/2^{l+1}] \times [n]^{d'}$, and v_{right} has label $\boldsymbol{f}_{v_{\text{right}}} = \boldsymbol{f}_v$, while v_{left} has label $\boldsymbol{f}_{v_{\text{left}}} = \boldsymbol{f}_v + (\underbrace{0,0,\ldots,0}_{d'}, 2^l, \underbrace{0,0,\ldots,0}_{d-d'-1})$.

3. The children of a node v corresponding to universe $[1] \times [n]^{d'}$ with $d' > 0$, are $v_{\text{left}}, v_{\text{right}}$, corresponding to universe $[n/2] \times [n]^{d'-1}$ and have labels $\boldsymbol{f}_{v_{\text{right}}} = \boldsymbol{f}_v$ and $\boldsymbol{f}_{v_{\text{left}}} = \boldsymbol{f}_v + (\underbrace{0,0,\ldots,0}_{d'-1}, 1, \underbrace{0,0,\ldots,0}_{d-d'})$ respectively.

4. A node v corresponding to universe $[1]$ is called a *leaf* in T_N^{full}.

The above rules create a binary tree of depth $\log N$, which corresponds to the FFT computation tree. The labels of the leaves of T_N^{full} represent the set $[n]^d$ of all possible frequencies of any signal $x : [n]^d \to \mathbb{C}$ in the Fourier domain. We demonstrate T_N^{full} that corresponds to the 2-dimensional FFT computation on universe $[4] \times [4]$ in Figure 3. Subtrees T of T_N^{full} can be defined as usual. For every node $v \in T$, the *level* of v, denoted by $l_T(v)$, is the distance from the root to v. We denote by $\text{LEAVES}(T)$ the set of all leaves of tree T, and for every $v \in \text{LEAVES}(T)$, its *weight* $w_T(v)$ *with respect to* T is the number of ancestors of v in tree T with two children. The levels (distances from the root) on which the aforementioned ancestors lie will be called $\text{Anc}(v, T)$. Furthermore, the sub-path of v with respect to T will be the children of the aforementioned ancestors which are not ancestors of v. Additionally, for a node $v \in T$ we denote the subtree of T rooted at v by T_v.

The following definition will be particularly important for our algorithms.

Definition 1 (Frequency cone of a leaf of T). For every subtree T of T_N^{full} and every node $v \in T$, we define the *frequency cone of v* with respect to T as,

$$\text{FreqCone}_T(v) := \left\{ \boldsymbol{f}_u : \text{ for every leaf } u \text{ in subtree of } T_N^{\text{full}} \text{ rooted at } v \right\}.$$

Furthermore, we define $\text{supp}(T) := \bigcup_{u \in \text{LEAVES}(T)} \text{FreqCone}_T(u)$.

The *splitting tree* of a set $S \subseteq [n]^d$ is the subtree of T_N^{full} that contains all nodes $v \in T_N^{\text{full}}$ such that $S \cap \text{FreqCone}_{T_N^{\text{full}}}(v) \ne \varnothing$.

The main technical innovation of [KVZ19] is the introduction of adaptive aliasing filters, a new class of filters that allow to isolate a given frequency from a given set of k other frequencies using

6

$O(k)$ samples in time domain and in $O(k \log N)$ time – see Section 7 for a more detailed account of this prior work.

Definition 2 ((v,T)-isolating filter, see Definition 11)**.** Consider a subtree T of T_N^{full}, and a leaf v of T. A filter $G : [n]^d \to \mathbb{C}$ is called (v,T)-*isolating* if the following conditions hold:

- For all $\boldsymbol{f} \in \text{FreqCone}_T(v)$, we have $\widehat{G}(\boldsymbol{f}) = 1$.

- For every $\boldsymbol{f}' \in \bigcup_{\substack{u \in \text{LEAVES}(T) \\ u \neq v}} \text{FreqCone}_T(u)$, we have $\widehat{G}(\boldsymbol{f}') = 0$.

As shown in [KVZ19], for a given tree T and a node v one can construct isolating filters G such that $\|G\|_0 = O(2^{w_T(v)})$, and $\widehat{G}(\boldsymbol{f})$ is computable in $\widetilde{O}(1)$ time (see also Lemma 9). The sparsity of G in time domain, i.e. $\|G\|_0$, corresponds to the number of accesses to x needed in order to get our hands on $(\widehat{G} \cdot \widehat{x})_{\boldsymbol{f}}$ for a fixed $\boldsymbol{f}$.

Unfortunately, as we have already pointed out, the algorithm in [KVZ19] works only for exactly k-sparse signals, and also demands cubic time and sample complexity. Our new toolkit shows that all three limitations can be remedied (though not completely simultaneously). The key observation underlying our new techniques is a new near-isometry property of adaptive aliasing filters.

Collectively, adaptive aliasing filters act as near-isometries. Adaptive aliasing filters as used in [KVZ19] are particularly effective for *non-obliviously* isolating elements of the head with respect to each other. However, in standard sparse recovery tasks, one desires control of the tail energy that participates in the measurement. This is a relatively easy (or at least well-understood) task in Sparse Fourier schemes which operate via ℓ_∞-box filters [HIKP12a, HIKP12b, IKP14, IK14, Kap17], but a non-trivial task using adaptive aliasing filters. The reason is that the tail via the latter filtering is hashed in a *non-uniform* way. The hashing depends on the arithmetic structure of the elements used to construct the filters, as well as their arithmetic relationship with the elements in the tail. This non-uniformity is essentially the main driving reason for the "exactly k-sparse" assumption in [KVZ19]. Our starting point is the observation that for every tree $T \subseteq T_N^{\text{full}}$, the (v,T)-isolating filters for $v \in \text{LEAVES}(T)$, satisfy the following orthonormality condition in dimension one, see subsection 11.1.

Lemma 1. *(Gram Matrix of adaptive aliasing filters in $d = 1$) Let $T \subseteq T_n^{\text{full}}$, let G_v be the (v,T)-isolating filter of leaf $v \in \text{LEAVES}(T)$, as per (4). Let v and v' be two distinct leaves of T. Then,* **(1)** $\|\widehat{G}_v\|_2^2 := \sum_{\xi \in [n]} |\widehat{G}_v(\xi)|^2 = n \cdot 2^{-w_T(v)}$ *and* **(2)** *the adaptive aliasing filters corresponding to v and v' are orthogonal, i.e.* $\langle \widehat{G}_v, \widehat{G}_{v'} \rangle := \sum_{\xi \in [n]} \widehat{G}_v(\xi) \cdot \overline{\widehat{G}_{v'}(\xi)} = 0$.

This already postulates that adaptive aliasing filters are relatively well-behaved: for a signal x with tree T all leaves of which have roughly the same weight, it must be the case that $x \mapsto \{\langle \widehat{G}_v, \widehat{x} \rangle\}_{v \in \text{LEAVES}(T)}$ is a near-orthonormal transformation. Of course, this is too much to ask in general. The crucial property that we will make use of is captured in the following Lemma, see Subsection 11.2.

Lemma 2. *(see Lemma 16) Consider a tree $T \subseteq T_N^{full}$. For every leaf v of T we let $\widehat{G}_v$ be a Fourier domain (v,T)-isolating filter. Then for every $\boldsymbol{\xi} \in [n]^d$, $\sum_{v \in \text{LEAVES}(T)} |\widehat{G}_v(\boldsymbol{\xi})|^2 = 1$.*

Using standard arguments, the above gives the following Lemma.

Lemma 3. *For $z : [n]^d \to \mathbb{C}$, let $z^{\to \boldsymbol{a}}$ be the cyclic shift of z by a, i.e. $z^{\to \boldsymbol{a}}(\boldsymbol{f}) := z(\boldsymbol{f} - \boldsymbol{a})$, where the subtraction happens modulo n in every coordinate. For a tree $T \subseteq T_N^{\text{full}}$,*

7

$$\mathbb{E}_{\boldsymbol{a} \sim U_{[n]^d}} \left[\sum_{v \in \text{LEAVES}(T)} |\langle \widehat{G_v}, \widehat{z^{\rightarrow \boldsymbol{a}}} \rangle|^2 \right] = \|z\|_2^2,$$

i.e. on expectation over a random shift the total collection of filters is an isometry.

The above property is the key new observation that underlies our analysis, but several other technical ideas are needed to obtain our robust result – a more detailed overview is given in Section 5. We also note that the our ultimate robust algorithm does not achieve the standard ℓ_2/ℓ_2 sparse recovery guarantees, which allow for recovery of a good approximation to the signal if the energy of the top k coefficients is larger than the energy of the tail. Instead, we show that recovery is possible when every one of the top k coefficients dominates the cumulative energy of the tail of the signal. It is weaker, but one must note that several recent works on Fourier sparse recovery are only known to tolerate inverse polynomial amounts of noise [HK15, Moi15], so our robustness guarantee appears to be a strong first step.

3.2   A Tree Exploration Problem.

We now present an abstract formulation of the Sparse Fourier transform algorithms which work based on the adaptive aliasing filters of [KVZ19] as an abstract Tree Exploration Problem. We will lay down the formal connection between this problem and the adaptive aliasing filter-based Sparse FFT in Section 9. For now, we focus on the tree problem without any reference to Fourier transforms.

The setup.   In this problem we are given a full binary tree T_N^{full} with N nodes where each leaf of this tree has a (potentially complex) number written on it, known as the *value* of the leaf. Suppose that T is an instance of T_N^{full} with at most k of its leaves having non-zero values. The goal of the tree exploration problem is to find those k leaves and estimate their corresponding values. It has been formally shown in [KVZ19] that there is a bijective correspondence between any k-sparse $\widehat{x} : [n]^d \to \mathbb{C}$ ($\|\widehat{x}\|_0 \leq k$) and such tree T with k non-zero valued leaves, so the Sparse FFT problem can be formulated as learning the tree. We will show this formally in Section 9.

Definition 3 (LEAVES and HEAVYLEAVES). For a node $v \in T$ we let $\text{LEAVES}(v)$ be the leaves of T_N^{full} which are the descendants of v (including v itself in case v is a leaf). We let $\text{HEAVYLEAVES}(v)$ denote the leaves in $\text{LEAVES}(v)$ with non-zero values.

For a set of vertices $S \subset T_N^{\text{full}}$ we will denote by $T(S)$ the subtree of T_N^{full} with minimum number of vertices containing S and the root.

Definition 4 (Weight of a vertex). For a binary tree T, and a vertex $v \in T$, the weight $w_T(v)$ of v is equal to the number of ancestors of v in T with two children. For a set $S \subseteq T_N^{\text{full}}$ and $v \in T_N^{\text{full}}$, we denote $w_S(v) := w_{T(S \cup \{v\})}(v)$ for simplicity.

It turns out that this weight function is subadditive for a fixed v.

Lemma 4. *For any two sets $S_1, S_2 \subseteq T_N^{\text{full}}$ and $v \in T_N^{\text{full}}$,*

$$w_{S_1 \cup S_2}(v) \leq w_{S_1}(v) + w_{S_2}(v)$$

8

You can find the proof in Appendix B.

The tree T is unknown to us and we can explore it indirectly only using two primitives; ZEROTEST, which can answer queries about whether HEAVYLEAVES(v) is empty, and ESTIMATE, which can estimate the value of a leaf. In this section we are not concerned about the internal working of these primitives and treat them as oracles. The implementation of these routines in the context of Sparse FFT is given in Section 9. In order to design efficient versions of these primitives to virtually prune the tree T_N^{full} and avoid operating on T_N^{full} as a whole, we need to introduce two extra parameters, Found and Excluded.

The Found and Excluded parameters. Found is an associative array of already recovered leaves with some estimates for their values. We say that the estimates in Found are correct if for all leaves v in Found, Found(v) is equal to the value of v in tree T. We denote by |Found| the number of leaves with non-zero estimates in Found. For two associative arrays, Found_1 and Found_2, $\text{Found}_1 + \text{Found}_2$ denotes their union. In our applications we will never take a union of two arrays with intersecting key sets. Our algorithm will virtually subtract the values in Found from the corresponding leaves of tree T, essentially deleting them if the estimates are correct.

Excluded is a subset of nodes of T_N^{full}. Our algorithm will virtually delete all subtrees with roots in Excluded from the tree. For technical reasons, the procedure only accept the set excluded in the form of a tree $T(\text{Excluded})$. For simplicity, we equate the set Excluded and its tree $T(\text{Excluded})$. One can pictorially see examples of these notions in Figure 2. This motivates the following definition:

Definition 5. A vertex $v \in T_N^{\text{full}}$ is *isolated* by Found and Excluded if

$$\text{LEAVES}(v) \cap \text{LEAVES}(\text{Excluded}) = \varnothing$$

and for all leaves ℓ of T_N^{full} not in $\text{LEAVES}(v)$, either the value of ℓ is zero, the estimate Found(ℓ) for the value of ℓ is correct, or $\ell \in \text{LEAVES}(\text{Excluded})$.

Now we are ready to present the interfaces of ZEROTEST and ESTIMATE as well as their runtime.

Assumption 1. *There exist procedures* ZEROTEST *and* ESTIMATE *with the following properties,*

1. ZEROTEST(Found, Excluded, v, b), *where b is a positive integer, representing budget, and v is a vertex in T_N^{full}. This routine checks if there are any leaves with non-zero value in the subtree of v. More formally, if v is isolated by* Found *and* Excluded *and* $|\text{HEAVYLEAVES}(v)| \leq b$, *then the routine returns True if for every $\ell \in \text{LEAVES}(v)$ either the value of ℓ is zero or the estimate* Found(ℓ) *for the value of ℓ is correct, and returns False otherwise; if v is not isolated, it can return either True or False. The runtime of* ZEROTEST *is* $\widetilde{O}\left(2^{w_{\text{Excluded}}(v)}b + |\text{Found}| \cdot b\right)$.

2. ESTIMATE(Found, Excluded, ℓ), *where ℓ is a leaf of T_N^{full}. This routine estimates the value of ℓ correctly if ℓ is isolated by* Found *and* Excluded. *If not, the estimate is arbitrary.*

   The time complexity of ESTIMATE *is* $\widetilde{O}\left(2^{w_{\text{Excluded}}(\ell)} + |\text{Found}|\right)$.

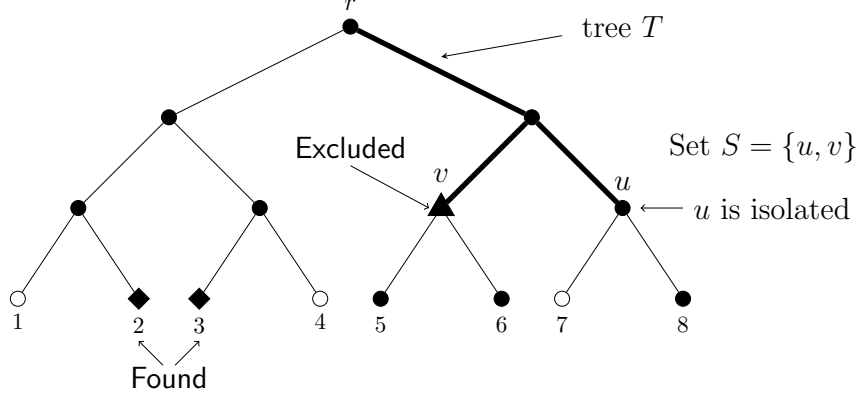*The above running time bounds still hold regardless of the correctness of the inputs.*

Figure 2: Example of a tree T_8^{full}. The leaves are labeled by integers 1 to 8. The unfilled leaves have zero associated value, hence $\text{HEAVYLEAVES}(u) = \{8\}$, $\text{LEAVES}(u) = \{7, 8\}$. Also the diamond-shaped leaves 2 and 3 are assumed to lie in the Found together with their correct estimates. Thus if the triangle-shaped node v is in the set Excluded, then u is isolated by Found and Excluded. Thick edges represent the tree T, and the set of its leaves is $S = \{u, v\}$. If we pick u as the next vertex in the vanilla algorithm, the set Excluded will again contain only the vertex v.

The vanilla algorithm in [KVZ19]. Using the above translation of Sparse FFT into a tree learning problem, the algorithm in [KVZ19] does the following. At all times it maintains a tree $T \subseteq T_N^{\text{full}}$ and a set Found of leaf nodes with their *perfect* estimates such that the following invariants hold:

$$\text{Found} \subseteq \text{HEAVYLEAVES}(\text{root}),$$

$$\text{HEAVYLEAVES}(\text{root}) \setminus \text{Found} \subseteq \text{LEAVES}(T).$$

Initially, T contains only the root of the tree and $\text{Found} = \varnothing$.

While T is not empty, the algorithm picks a leaf $u \in T$ with the smallest weight $w_T(u)$. It now needs to find a set Excluded such that u would be isolated by Found and Excluded. As it turns out, it is enough for Excluded to contain the children of nodes in T on the path from u to the root, except for the nodes that are on this path themselves — see Fig. 2 for an illustration. At every point the algorithm calls $\text{ZEROTEST}(\text{Found}, \text{Excluded}, u, k)$ to test whether the subtree rooted at u contains a non-recovered leaf, in order to avoid exploring empty subtrees. The budget of ZEROTEST is chosen to be k in order to avoid false negatives, i.e. never miss a non-empty subtree. This is the main inefficiency in [KVZ19] which we address here, obtaining a quadratic time algorithm. Before outlining our main algorithmic technique, which allows us to handle frequent false negatives in ZEROTEST by a novel error correction mechanism, we note that quadratic time is a natural barrier for any algorithm that iteratively recovers the input signal. Indeed, suppose that the algorithm has recovered a constant fraction of coefficients and recurses on the rest. The most common approach here is to subtract the recovered elements from time domain samples, reducing to the same problem with a smaller number of coefficients – but this requires nearly quadratic time by our lower bound (see Theorem 2 in Section 10).

3.3 Obtaining Almost Quadratic Runtime via Hierarchical Error Correction.

Our main insight is that not all calls to ZEROTEST need to succeed. Instead, one can try to assign varying budgets to the nodes to be explored and then perform *hierarchical error correction* in order to detect errors in exploration which are caused by the failures of ZEROTEST due to incorrect budget assignments. We explain the main underlying idea next.

Suppose that we optimistically explore a subtree of T_N^{full} with a budget $b \ll k$ for ZeroTest. If the budget assigned to the subtree was correct, i.e. number of heavy leaves in that subtree are no larger than b, then we correctly recover the leaves locally and the resulting speed-up will be a multiplicative k/b factor. On the other hand, if the assigned budget was wrong, exploration could be misguided and estimates could be incorrect. The idea is that this error can be detected at an ancestor of the subtree if we assign that ancestor a large enough budget so that the invocation of ZeroTest does not get fooled. As a sanity check, note that this is definitely the case for the root, where we use a budget of size k. Once the error is detected at an ancestor of the subtree in question, the algorithm will re-explore that subtree with increased budget. Roughly speaking, the algorithm tries to learn the correct budget of each subtree by performing *backtracking*: guess a budget, explore the subtree, determine whether it is wrong upon backtracking to an ancestor and subsequently increase the budget to explore that subtree, so on so forth. In this approach there are two things that need to be carefully balanced. On the one hand, one needs to use the smallest possible budgets, close to the actual sizes of the corresponding subtrees (so that calls to ZeroTest are cheap) and on the other hand one needs to control the amount of backtracking the algorithm performs; a smaller budget leads to a larger number of required backtracking steps. A careful analysis reveals that, roughly speaking, increasing the budget by a factor of $1/\alpha$ for $\alpha := 2^{-2\sqrt{\log k \cdot \log \log N}}$ whenever an adjustment is needed ensures nearly quadratic runtime.

The pseudocode of the recursive recovery algorithm is presented in Algorithm 1. The algorithm is passed budget s, which is assumed to be the sparsity of the subtree of v. If the sparsity is low it defaults to the cubic algorithm of [KVZ19]. Otherwise it starts the inner loop in lines 6 to 29, where it explores the tree. It maintains the tree T containing all of unexplored heavy leaves of v as a set S of its leaves. On each iteration of inner loop, the algorithms picks a vertex z from S with minimum weight in line 7, for which it first checks if there is any heavy leaves in LEAVES(z). If not, it discards the vertex and continues to the next iteration. Otherwise, it tries to guess that the sparsity of each child of z are at most $\alpha \cdot s$ and runs itself recursively on both children of z with a this decreased budget in lines 19 and 20. For each child it then checks using ZeroTest in lines 22 and 23 if the guess was correct and the values were found correctly, they get added to Found$_{out}$, otherwise the corresponding child is added to S. Finally, if the z is the leaf of the T_N^{full}, the algorithm runs ESTIMATE on it instead and then adds the recovered value to Found$_{out}$.

3.4 Analysis of ExactSparseRecovery.

In this section we shall present analysis of Algorithm 1. The key idea is that instead of using a large budget that is sufficient for ZeroTest to succeed every time, we try to explore the subtrees with a lower budget and then check with a larger budget whether the subtrees have been recovered correctly or not.

To do the analysis we will need the correctness guarantee for the base Algorithm 10.

Recall that b is the input parameter of SlowExactSparseRecovery.

Theorem 5 (Correctness of Algorithm 10). *If* |HeavyLeaves(v)| $\leq b$ *and v is isolated by* Found *and* Excluded*, then the procedure* SlowExactSparseRecovery *returns the correct estimates for all* HeavyLeaves(v).

The proof can be found in Appendix A.

Theorem 6 (Correctness of Algorithm 1). *Consider a call to primitive* ExactSparseRecovery*(*Found*,* Excluded*, v, s, k) for any vertex $v \in T_N^{\text{full}}$, budget $s \leq k$ and sets* Found *and* Excluded *such that v is isolated by them (see Definition 5) and* Found $\cap$ Leaves(v) [6]. *If the sparsity of subtree rooted*

---
[6]The algorithm works even without this requirement, however it somewhat simplifies the proof.

Algorithm 1 EXACTSPARSERECOVERY(Found, Excluded, v, s, k)

1: **if** $s \leq 1/\alpha$ **then**   ▷ This is the base case — run the qubic time algorithm
2:      $\mathsf{Found}_{out} \leftarrow$ SLOWEXACTSPARSERECOVERY(Found, Excluded, v, s)
3:      **if** $|\mathsf{Found}_{out}| \leq s$ **return** Found_{out} **else return** $\varnothing$

4: $\mathsf{Found}_{out} \leftarrow \varnothing$, $S \leftarrow \{v\}$, Steps $\leftarrow 1$ ▷ Construct a subtree tree of T_N^{full} rooted at v,
5:   ▷ with S as the set of leaves (so S is initialized as $\{v\}$)
6: **repeat**
7:      $z \leftarrow$ vertex in S with the minimum weight with respect to S
8:      $S = S \setminus \{z\}$, Steps $\leftarrow$ Steps $+ 1$
9:      $\mathsf{Excluded}' \leftarrow \mathsf{Excluded} \cup S$, $\mathsf{Found}' \leftarrow \mathsf{Found} + \mathsf{Found}_{out}$
10:      **if** ZEROTEST($\mathsf{Found}', \mathsf{Excluded}', z, s$) **then continue**
11:                               ▷ No heavy leaves in the subtree of z, so we remove it
12:      **if** z is a leaf in T_N^{full} **then**
13:          $\mathsf{Found}_{out}(z) \leftarrow$ ESTIMATE($\mathsf{Found}', \mathsf{Excluded}', z$)
14:          **continue**
15:
16:      $s_{desc} \leftarrow \min(\alpha \cdot s, k - |\mathsf{Found}'|)$ ▷ s_{desc} is the budget for the children of z
17:      $z_{\text{left}}, z_{\text{right}} \leftarrow$ left and right child of z in T_N^{full} respectively.
18:
   ▷ Try to estimate the values in the subtrees of the children of z with a smaller budget s_{desc}
19:      $\mathsf{Found}_{\text{left}} \leftarrow$ EXACTSPARSERECOVERY($\mathsf{Found}', \mathsf{Excluded}' \cup \{z_{\text{right}}\}, z_{\text{left}}, s_{desc}, k$)
20:      $\mathsf{Found}_{\text{right}} \leftarrow$ EXACTSPARSERECOVERY($\mathsf{Found}', \mathsf{Excluded}' \cup \{z_{\text{left}}\}, z_{\text{right}}, s_{desc}, k$)
21:
   ▷ Check if the values were correctly recovered
22:      $\mathsf{IsZero}_{\text{left}} \leftarrow$ ZEROTEST($\mathsf{Found}' + \mathsf{Found}_{\text{left}}, \mathsf{Excluded}' \cup \{z_{\text{right}}\}, z_{\text{left}}, s$)
23:      $\mathsf{IsZero}_{\text{right}} \leftarrow$ ZEROTEST($\mathsf{Found}' + \mathsf{Found}_{\text{right}}, \mathsf{Excluded}' \cup \{z_{\text{left}}\}, z_{\text{right}}, s$)
24:
25:
   ▷ If the estimates appear correct under current budget, save them
   ▷ Otherwise, we must increase the budget for the child, so we add it to the set S
26:      **If** $\mathsf{IsZero}_{\text{left}}$ **then** $\mathsf{Found}_{out} \leftarrow \mathsf{Found}_{out} + \mathsf{Found}_{\text{left}}$ **else** $S \leftarrow S \cup \{z_{\text{left}}\}$
27:      **If** $\mathsf{IsZero}_{\text{right}}$ **then** $\mathsf{Found}_{out} \leftarrow \mathsf{Found}_{out} + \mathsf{Found}_{\text{right}}$ **else** $S \leftarrow S \cup \{z_{\text{right}}\}$
28:
29: **until** $S = \varnothing$, Steps $> \frac{6 \log N}{\alpha}$ or $|\mathsf{Found}_{out}| > s$
30:
31: **if** $S = \varnothing$, Steps $\leq \frac{6 \log N}{\alpha}$ and $|\mathsf{Found}_{out}| \leq s$ **then**
32:      **return** Found_{out}
33: **else**
34:      **return** $\varnothing$

---

at v is less than the allowed budget, that is $|\text{HEAVYLEAVES}(v)| \leq s$, then EXACTSPARSERECOVERY *returns a correct estimate for every leaf in* HEAVYLEAVES(v). *In particular, if $v =$* root, Excluded $= \varnothing$, Found $= \varnothing$ *and* $|\text{HEAVYLEAVES}(r)| \leq s$, *the procedure correctly recovers the entire tree.*

Proof. We will show correctness by induction on the budget s. The **base case** is provided by $s \leq 1/\alpha$. In that case the procedure calls SLOWEXACTSPARSERECOVERY (see line 1 and line 2),

and thus the output is correct by Theorem 5. We now provide the **inductive step**.

Now, we show by induction on the number of iterations of the **repeat** loop in line 6 that for a fixed s the returned frequencies are correct. The set S is the set of leaves of the tree T, described earlier in Section 3.3. We will show that the following invariants hold:

(1) $\mathsf{Found}_{out} \subseteq \textsc{HeavyLeaves}(v)$ and the estimated values in Found_{out} are all correct

(2) $\textsc{HeavyLeaves}(v) \setminus \mathsf{Found}_{out} \subseteq \textsc{Leaves}(S)$[7]

Then the correctness follows from the fact that $S = \varnothing$ at the end of execution.

Consider an iteration of the **repeat** loop where the above invariants hold and let z be the vertex extracted from S in line 7. By the inductive hypothesis, z is isolated by Found' and $\mathsf{Excluded}'$ (see line 9), and by the theorem assumption, $|\textsc{HeavyLeaves}(v)| \leq s$. Therefore, the calls to $\textsc{ZeroTest}$ and $\textsc{Estimate}$ in lines 10 and 13 return correct answers. Therefore, in case $\textsc{HeavyLeaves}(z)$ is empty the node z can be removed from S, or in case z is a leaf in T_N^{full} $\textsc{Estimate}$ returns a correct estimate for z and it again can be removed from S and the invariants still hold.

Otherwise, the algorithm recursively calls itself on z_{left} and z_{right}. Assume that we virtually add z_{left} and z_{right} to S, and remove z from it. Then the invariants still hold. We will only discuss the correctness of operations with z_{left}, since they are symmetric. z_{left} is isolated by Found' and $\mathsf{Excluded}' \cup \{z_{\text{right}}\}$, since z is isolated by Found' and $\mathsf{Excluded}'$. Therefore, if $|\textsc{HeavyLeaves}(z_{\text{left}})| \leq \alpha s$, $\mathsf{Found}_{\text{left}}$ will contain correct estimates of $\textsc{HeavyLeaves}(z_{\text{left}})$, by the inductive hypothesis. Because the algorithm doesn't know if that is the case, it checks whether the recovered values are correct or not in line 22 by running $\textsc{ZeroTest}$. $\mathsf{IsZero}_{\text{left}}$ would be True only if $\textsc{HeavyLeaves}(z_{\text{left}})$ were correctly recovered. From this fact it follows that if $\textsc{HeavyLeaves}(z_{\text{left}})$ were correctly recovered, we can remove z_{left} from S and update Found by adding $\mathsf{Found}_{\text{left}}$ to it without violating the invariants, and if not, we can just discard $\mathsf{Found}_{\text{left}}$. Notice that under the theorem's assumption on sparsity and by invariant 1 it can never happen that $|\mathsf{Found}_{out}| > s$.

Finally, notice that if $|\textsc{HeavyLeaves}(z)| \leq \alpha \cdot s$, the subtree of z will be completely recovered by a recursive call. Therefore, only nodes with $|\textsc{HeavyLeaves}(z)| \geq \alpha \cdot s$ and their children get added to S. By an averaging argument, the number of such nodes is at most $\frac{\log N}{\alpha}$, thus, the maximum number of nodes that would ever be added to S is $3\frac{\log N}{\alpha}$. Because at each iteration at least one vertex is removed from S, $\text{Steps} \leq \frac{6 \log N}{\alpha}$ and at the end of the loop $S = \varnothing$. $\qquad \square$

We finish this section with the runtime analysis of Algorithm 1. To do it, we will need to use the correctness guarantee for Algorithm 10, proof of which can be found in Appendix A.

Theorem 7 (Running time of Algorithm 10). *If $\textsc{Leaves}(v) \cap \textsc{Leaves}(\mathsf{Excluded}) = \varnothing$, the runtime of $\textsc{SlowExactSparseRecovery}$ is bounded by $\widetilde{O}\left(|\mathsf{Found}| \cdot b^2 + 2^{w_{\mathsf{Excluded}}(v)} \cdot b^3\right)$.*

Theorem 8 (Running time of $\textsc{ExactSparseRecovery}$). *Let $\log N > 6\sqrt{5}$ and $k \leq N$. The running time of $\textsc{ExactSparseRecovery}(\varnothing, \varnothing, \text{root}, k, k)$ is bounded by $\widetilde{O}(k^2 \cdot 2^{8\sqrt{\log k \log \log N}})$.*

Proof. Recall that $\alpha := 2^{-2\sqrt{\log k \cdot \log \log N}}$. First, we make several observations:

- Each call to $\textsc{ExactSparseRecovery}$ (Algorithm 1) returns Found_{out} of size at most s.

- At most $6\frac{\log N}{\alpha}$ vertices are inserted in S in a single invocation of $\textsc{ExactSparseRecovery}$, and there the number of recursive calls to $\textsc{ExactSparseRecovery}$ is bounded by $12\frac{\log N}{\alpha}$.

---

[7] Recall the definition of $\textsc{HeavyLeaves}$ – Definition 3.

- Similarly, at all times during an invocation of EXACTSPARSERECOVERY one has $|\mathsf{Found}_{out}| \leq \alpha s \cdot \frac{12 \log N}{\alpha} = \widetilde{O}(s)$. Also, because $s_{desc} \leq k - |\mathsf{Found}'|$ from line 16, it is guaranteed that $|\mathsf{Found}'| \leq k + \widetilde{O}(\alpha s)$.

- For each recursive call in lines 19 and 20 as well as the call to SLOWEXACTSPARSEFFT, we have $\mathrm{LEAVES}(v) \cap \mathrm{LEAVES}(\mathsf{Excluded}) = \varnothing$. This means that the precondition of Theorem 7 is satisfied and that for each picked z,

$$w_{\mathsf{Excluded} \cup S}(z) \leq w_{\mathsf{Excluded}}(z) + w_S(z) \leq w_{\mathsf{Excluded}}(z) + \log(6 \log N / \alpha).$$

  where the first inequality follows from Lemma 4.

Recall that bound our basic setup an invocation of ZEROTEST takes $\widetilde{O}(2^{w_{\mathsf{Excluded}}(z)} \cdot b + |\mathsf{Found}| \cdot b)$ time, an invocation of ESTIMATE takes $\widetilde{O}(2^{w_{\mathsf{Excluded}}(z)} + |\mathsf{Found}|)$ time and an invocation of SLOWEXACTSPARSEFFT takes $\widetilde{O}(2^{w_{\mathsf{Excluded}}(z)} \cdot b^3 + |\mathsf{Found}| \cdot b^2)$ time respectively. We will separately bound the time dependent on $|\mathsf{Found}|$ and $2^{w_{\mathsf{Excluded}}(v)}$. Formally, we say that there are two runtime pools, first and second, and the aforementioned primitives spend $\widetilde{O}(2^{w_{\mathsf{Excluded}}(v)} \cdot b)$, $\widetilde{O}(2^{w_{\mathsf{Excluded}}(v)})$ and $\widetilde{O}(2^{w_{\mathsf{Excluded}}(v)} \cdot b^3)$ from the first pool, and $\widetilde{O}(|\mathsf{Found}| \cdot b)$, $\widetilde{O}(|\mathsf{Found}|)$ and $\widetilde{O}(|\mathsf{Found}| \cdot b^2)$ from the second one, respectively. We now bound the sizes of both pools. Using the notation $A := \frac{6 \log N}{\alpha}$, we have

The first pool. Let $T_1[s, W, l]$ be the time the procedure spends from the first pool where for convenience of notation $W = 2^{w_{\mathsf{Excluded}}(v)}$ and l is the distance to the node v from the root. Notice that the algorithm makes at most A iterations, since at most A vertices are added to S. That also means that $|S| \leq A$. Also notice that for each picked z, since z is the vertex with the smallest weight in S, by Lemma 4 and by Lemma 10

$$2^{w_{\mathsf{Excluded} \cup S}(z)} \leq 2^{w_{\mathsf{Excluded}}(z)} \cdot 2^{w_S(z)} \leq W \cdot A.$$

Notice that because we only call ESTIMATE in line 13, there always a call to ZEROTEST in line 10 that precedes it. Since they are called with the same set of parameters, by Assumption 1 the time to run ZEROTEST dominates that of ESTIMATE. Similarly, it is easy to see that the time to perform all other operations except for recursive calls is also dominated by ZEROTEST, so there exists an $f = \widetilde{O}(1)$ such that,

- If $l = \log N$, then we are in the case where v is a leaf of T_N^{full}. The algorithm makes only one iteration of the **repeat** loop in line 6 where it executes ESTIMATE and stops at line Line 14. The ZEROTEST call takes time $\widetilde{O}(2^{w_{\mathsf{Excluded}}(v)} s) = \widetilde{O}(sW) \leq sWf$, so

$$T_1[s, W, l] \leq sWf.$$

- Else, if $s \leq 1/\alpha$, then we are in the base case where Algorithm 10 is called in line 2. By Theorem 7 it takes time $\widetilde{O}(s^3 2^{w_{\mathsf{Excluded}}(v)}) \leq s^3 W f$ from the first pool to run it, so

$$T_1[s, W, l] \leq s^3 W f.$$

- Else the algorithm proceeds with the recursive mode of operation. As was discussed before, it makes at most A iterations of the **repeat** loop where it runs ZEROTEST and calls itself recursively in lines 19 and 20. Now ZEROTEST is called with set Excluded being equal to $\mathsf{Excluded} \cup S$, so its runtime from the first pool is bounded by $WAsf$. For each recursive

call, similarly, the set Excluded becomes Excluded $\cup\, S \cup \{z'\}$, where z' is the other child of z (see lines 19 and 20), hence the new W is upper bounded by $2WA$. The distance l also increases by at least 1. Finally, by observing that $T_1[s, W, l]$ is monotonically non-decreasing with respect to its parameters, we get the following formula

$$T_1[s, W, l] \leq A(WAsf + 2T_1[\alpha \cdot s, 2W \cdot A, l + 1]).$$

We can now show by induction that

$$T_1[s, W, l] \leq (5\alpha A^2)^{\frac{\log s}{\log 1/\alpha}} W \cdot s \cdot f/\alpha^2. \tag{1}$$

The base case corresponds to $s \leq 1/\alpha$ or $l = \log N$ for which we get by the above inequalities that the runtimes respectively are $s^3 W f \leq sW f/\alpha^2$ and $sW f$, both of which are not greater than the right hand side of Equation (1).

Suppose now that for $s > 1/\alpha$ and $l < \log N$, the inductive hypothesis holds for smaller values of s or larger values of l. Then $\beta \geq 1$ and

$$T_1[s, W, l] \leq A(WAsf + 2T_1[\alpha s, 2W \cdot A, l + 1]), \tag{2}$$

where we have

$$T_1[\alpha s, 2W \cdot A, l + 1] \leq 2(5\alpha A^2)^{\frac{\log \alpha s}{\log 1/\alpha}} WA \cdot \alpha s \cdot f/\alpha^2$$

$$\leq 2(5\alpha A^2)^{\frac{\log s}{\log 1/\alpha} - 1} WA \cdot s \cdot f/\alpha$$

by the inductive hypothesis. Substituting this bound into (2), we get

$$T_1[\alpha s, 2W \cdot A, l + 1] \leq A(WAsf + 4(5\alpha A^2)^{\frac{\log s}{\log 1/\alpha} - 1} WA \cdot s \cdot f/\alpha)$$

$$= Wsf(A^2 + (5\alpha A^2)^{\frac{\log s}{\log 1/\alpha} - 1} \cdot 4A^2/\alpha)$$

$$= Wsf(\alpha^2 A^2 + (5\alpha A^2)^{\frac{\log s}{\log 1/\alpha} - 1} \cdot 4\alpha A^2)/\alpha^2$$

$$\leq Wsf(\alpha^2 A^2 + (5\alpha A^2)^{\frac{\log s}{\log 1/\alpha} - 1} \cdot 4\alpha A^2)/\alpha^2$$

$$\leq Wsf(\alpha^2 A^2 + (5\alpha A^2)^{\frac{\log s}{\log 1/\alpha} - 1} \cdot 4\alpha A^2)/\alpha^2$$

$$\leq (5\alpha A^2)^{\beta} Wsf/\alpha^2,$$

where in the last transition we used the fact that $(5\alpha A^2)^{\frac{\log s}{\log 1/\alpha} - 1} = (5(6 \log N)^2/\alpha)^{\frac{\log s}{\log 1/\alpha} - 1} \geq 5^{\frac{\log s}{\log 1/\alpha} - 1} \geq 1$ (the latter bound holds since $\frac{\log s}{\log 1/\alpha} - 1 \geq 0$, as $s \geq 1/\alpha$ in the inductive step). This completes the inductive step, establishing (1). Substituting the values for α, W, f and A and simplifying, we get

$$T_1[k, 1, 0] \leq (5\alpha A^2)^{\frac{\log k}{\log 1/\alpha}} W \cdot k \cdot f/\alpha^2$$

$$\leq (5\alpha A^2)^{\frac{\log k}{\log 1/\alpha}} 2^{w_{\text{Excluded}}(v)} \cdot k \cdot f/\alpha^2 \tag{3}$$

$$= \widetilde{O}((5\alpha A^2)^{\frac{\log k}{\log 1/\alpha}} k/\alpha^2),$$

where we used the fact that $2^{w_{\text{Excluded}}(v)} = 1$ when v is the root of T_N^{full}.

It remains to upper bound $(5\alpha A^2)^{\frac{\log k}{\log 1/\alpha}} = (5(6\log N)^2/\alpha)^{\frac{\log k}{\log 1/\alpha}}$. We bound the logarithm of this value:

$$\frac{\log k}{\log 1/\alpha}\log(5(6\log N)^2/\alpha) \leq \frac{\log k}{\log 1/\alpha}(\log(\log^4 N) + \log(1/\alpha))$$

$$\leq 4\frac{\log k \log\log N}{\log 1/\alpha} + \log k$$

$$\leq 2\sqrt{\log k \log\log N} + \log k.$$

Substituting this into (3) and recalling that $\alpha = 2^{-2\sqrt{\log k \cdot \log\log N}}$, we get

$$T_1[k, 1, 0] = \widetilde{O}(k^2 \cdot 2^{8\sqrt{\log k \log\log N}})$$

as required.

The second pool. We bound $|\mathsf{Found}'|$ by $\widetilde{O}(k)$. Let $T_2[s, l]$ denote the upper bound on the runtime from the second pool, where l is the distance from v to root. Again, the runtime is dominated by the call to ZeroTest, so for some $f = \widetilde{O}(1)$ the following relations hold:

- If $l = \log N$, then $T_2[s, l] \leq kf$.

- Else, if $s \leq 1/\alpha$, then $T_2[s, l] \leq ks^2 f \leq kf/\alpha^2$.

- Else, $T_2[s, l] \leq A(ksf + 2T_2[\alpha s, l + 1])$.

Similarly to the first pool, one can show by induction that $T_2[s, l] \leq (3\alpha A)^{\beta} ksf/\alpha^2$, where $\beta = \lfloor\frac{\log s}{\log 1/\alpha}\rfloor$. Using the fact that $(3\alpha A)^{\beta} ksf \leq 2^{2\sqrt{\log k \log\log n}}$ we have $T_2[k, 0] = \widetilde{O}(k^2 \cdot 2^{8\sqrt{\log k \log\log n}})$. Summing up runtimes of both pools yields total runtime of $\widetilde{O}(k^2 \cdot 2^{8\sqrt{\log k \log\log n}})$.

Finally, the time spent on maintaining $\mathsf{Excluded}'$ is negligible, since, similarly to SLOWEX-ACTSPARSERECOVERY, it can be constructed once at the beginning of the algorithm, and on each iteration we modify it by adding and removing a constant number of vertices to or from it. Hence, by the same proof as in Theorem 7 the used time is $\widetilde{O}(s + w_{\mathsf{Excluded}}(v))$.

$\square$

4 Preliminaries and Notations.

Fourier transform basics. We will often identify $[n]^d \to \mathbb{C}$ with $\mathbb{C}^{n^d}$ for convenience and use the two interchangeably depending on the context.

Definition 6 (Fourier transform). For any positive integers d and n, the *Fourier transform* of a signal $x \in \mathbb{C}^{n^d}$ is denoted by $\widehat{x}$, where $\widehat{x}_{\boldsymbol{f}} = \sum_{\boldsymbol{t} \in [n]^d} x_{\boldsymbol{t}} e^{-2\pi i \frac{\boldsymbol{f}^\top \boldsymbol{t}}{n}}$ for any $\boldsymbol{f} \in [n]^d$. Here $\boldsymbol{f}^\top \boldsymbol{t} = \sum_{q=0}^{d-1} f_q t_q$.

Recall that by Parseval's theorem we have $\|\widehat{x}\|_2^2 = n^d \cdot \|x\|_2^2$. Furthermore, recall the convolution-multiplication duality $\widehat{(x \star y)} = \widehat{x} \cdot \widehat{y}$, where $x \star y \in C^{n^d}$ is the convolution of x and y and defined by the formula $(x \star y)_{\boldsymbol{t}} = \sum_{\boldsymbol{\tau} \in [n]^d} x_{\boldsymbol{\tau}} \cdot y_{(\boldsymbol{t}-\boldsymbol{\tau} \mod n)}$ for all $\boldsymbol{t} \in [n]^d$, where the modulus is taken coordinate-wise. We will also need the following well-known theorem on the Fourier subsampled matrices.

Theorem 9. *(Restricted Isometry Property of Subsampled Fourier Matrices, [HR17, Theorem 3.7])* *Let $q = \Theta(s \log^3 N)$. Then with high probability in N, the time domain points $\{x_{\boldsymbol{t}}\}_{\boldsymbol{t} \in Q}$ for a random multiset $Q \subseteq [n]^d$ with q uniform samples are sufficient to $(1 \pm \epsilon)$-approximate the energy of all s-sparse vectors $\widehat{x}$, where $\epsilon > 0$ is some absolute constant. Formally, simultaneously for all s-sparse vectors: $\frac{N^2}{q} \sum_{\boldsymbol{t} \in Q} |x_{\boldsymbol{t}}|^2 \in \left[(1-\epsilon)\|\widehat{x}\|_2^2, (1+\epsilon)\|\widehat{x}\|_2^2\right].$*

4.1 Notation for Manipulating FFT Computation Trees.

Recall that given a signal $x : [n]^d \to \mathbb{C}$, the execution of the FFT algorithm produces a binary tree, referred to as T_N^{full}. The root of T_N^{full} corresponds to the universe $[n]^d$, while the children of the root correspond to $[n/2] \times [n]^{d-1}$; note that FFT recurses by peeling off the least significant bit. Every node v has a *label* $\boldsymbol{f}_v \in \mathbb{Z}_n^d$ associated to it, defined according to the following rules.

1. The root has label $\boldsymbol{f}_{\text{root}} = (\underbrace{0, 0, \ldots, 0}_{d \text{ entries}})$, and corresponds to the universe $[n]^d$.

2. The children $v_{\text{left}}, v_{\text{right}}$ of a node v which corresponds to the universe $[n/2^l] \times [n]^{d'}$, with $0 \leq d' \leq d-1, 0 \leq l \leq \log n - 1$, have the following properties. Both correspond to universe $[n/2^{l+1}] \times [n]^{d'}$, and v_{right} has label $\boldsymbol{f}_{v_{\text{right}}} = \boldsymbol{f}_v$, while v_{left} has label $\boldsymbol{f}_{v_{\text{left}}} = \boldsymbol{f}_v + (\underbrace{0, 0, \ldots, 0}_{d'}, 2^l, \underbrace{0, 0, \ldots, 0}_{d-d'-1})$.

3. The children of a node v corresponding to universe $[1] \times [n]^{d'}$ with $d' > 0$, are $v_{\text{left}}, v_{\text{right}}$, corresponding to universe $[n/2] \times [n]^{d'-1}$ and have labels $\boldsymbol{f}_{v_{\text{right}}} = \boldsymbol{f}_v$ and $\boldsymbol{f}_{v_{\text{left}}} = \boldsymbol{f}_v + (\underbrace{0, 0, \ldots, 0}_{d'-1}, 1, \underbrace{0, 0, \ldots, 0}_{d-d'})$ respectively.

4. A node v corresponding to the universe $[1]$ is called a *leaf* in T_N^{full}.

The above rules create a binary tree of depth $\log N$, which corresponds to the FFT computation tree. The labels of the leaves of T_N^{full} represent the set $[n]^d$ of all possible frequencies of any signal $x : [n]^d \to \mathbb{C}$ in the Fourier domain. We demonstrate T_N^{full} that corresponds to the 2-dimensional FFT computation on universe $[4] \times [4]$ in Figure 3. Subtrees T of T_N^{full} can be defined as usual. For every node $v \in T$, the *level* of v, denoted by $l_T(v)$, is the distance from the root to v. We denote by $\text{LEAVES}(T)$ the set of all leaves of tree T, and for every $v \in \text{LEAVES}(T)$, its *weight* $w_T(v)$ *with respect to* T is the number of ancestors of v in tree T with two children. The levels (distances

17

from the root) on which the aforementioned ancestors lie will be called $\mathrm{Anc}(v, T)$. Furthermore, the sub-path of v with respect to T will be the children of the aforementioned ancestors which are not ancestors of v. Additionally, for a node $v \in T$ we denote the subtree of T rooted at v by T_v.

The following definition will be particularly important for our algorithms.

Definition 7 (Frequency cone of a leaf of T)**.** For every subtree T of T_N^{full} and every node $v \in T$, we define the *frequency cone of v* with respect to T as,

$$\mathrm{FreqCone}_T(v) := \left\{ \boldsymbol{f}_u : \text{ for every leaf } u \text{ in subtree of } T_N^{\mathrm{full}} \text{ rooted at } v \right\}.$$

Furthermore, we define $\mathrm{supp}(T) := \bigcup_{u \in \mathrm{LEAVES}(T)} \mathrm{FreqCone}_T(u)$.

The *splitting tree* of a set $S \subseteq [n]^d$ is the subtree of T_N^{full} that contains all nodes $v \in T_N^{\mathrm{full}}$ such that $S \cap \mathrm{FreqCone}_{T_N^{\mathrm{full}}}(v) \neq \varnothing$.

5 Techniques and Comparison with the Previous Technology.

This section is devoted to highlighting the differences between previous work and our technical contributions.

5.1 Previous Techniques.

Most previous sublinear-time Sparse Fourier transform algorithms [GMS05, HIKP12a, Kap16, Kap17] rely on emulating the hashing of signal $\widehat{x}$ by picking a structured set of samples (in low dimensions, the samples correspond to arithmetic progressions) and processing them with the help of bandpass filters, i.e. functions which approximate the ℓ_∞ box in frequency domain and are simultaneously sparse in time domain. However, while those filters are particularly efficient in low dimensions, their performance deteriorates when the number of dimensions increases: indeed, a d-dimensional ℓ_∞ box has 2^d faces, and hence this approach suffers inevitably from the curse of dimensionality. On the other hand, an unstructured collection of $O(k \cdot \mathrm{poly}(\log N))$ samples [CT06, NSW19] suffice, showing that the sample complexity is dimension-independent; the cost that one needs to pay, however, is $\Omega(N)$ running time.

To (partially) remedy the aforementioned state of affairs, the approach of [KVZ19] departs from both the aforementioned approaches, and performs pruning in the Cooley-Tukey FFT computation graph, in a way that suffices for recovery of *exactly k-sparse* vectors. Recall that as we explained in Section 3, the exact Sparse FFT problem can be translated to a tree exploration problem. What makes the exploration possible and is the main technical innovation of [KVZ19] is the introduction of adaptive aliasing filters, a new class of filters that allow to isolate a given frequency from a given set of k other frequencies using $O(k)$ samples in time domain and in $O(k \log N)$ time. Those filters are revised in Section 7.

Definition 8 ((v, T)-isolating filter, see Definition 11)**.** Consider a subtree T of T_N^{full}, and a leaf v of T. A filter $G : [n]^d \to \mathbb{C}$ is called *(v, T)-isolating* if the following conditions hold:

- For all $\boldsymbol{f} \in \mathrm{FreqCone}_T(v)$, we have $\widehat{G}(\boldsymbol{f}) = 1$.

- For every $\boldsymbol{f}' \in \bigcup_{\substack{u \in \mathrm{LEAVES}(T) \\ u \neq v}} \mathrm{FreqCone}_T(u)$, we have $\widehat{G}_v(\boldsymbol{f}') = 0$.

18

As shown in [KVZ19], for a given tree T and a node v one can construct isolating filters G such that $\|G\|_0 = O(2^{w_T(v)})$, and $\widehat{G}(\boldsymbol{f})$ is computable in $\widetilde{O}(1)$ time (see also Lemma 9). The sparsity of G in time domain, i.e. $\|G\|_0$, corresponds to the number of accesses to x needed in order to get our hands on $(\widehat{G} \cdot \widehat{x})_{\boldsymbol{f}}$ for a fixed $\boldsymbol{f}$.

As was shown in Section 3, the FFT tree exploration proceeds using two primitives that satisfy Assumption 1 and both of these primitives can be efficiently constructed given the above filters. The first one is a primitive for performing a *zero test* on a subtree, i.e., checking whether $\widehat{x}_{\mathrm{FreqCone}(v)} \equiv 0$. This check can be performed efficiently using a (deterministic) collection of $O(k \log^3 N)$ samples which satisfy the Restricted Isometry Property (RIP) of order k; its pseudocode, named ZeroTest, is depicted in Algorithm 2. The sample complexity of ZeroTest is then

$$O(2^{w_T(v)} \cdot k \cdot \mathrm{poly}(\log N)),$$

namely, one needs to multiply the time domain support size of the isolating filter G with the number of samples needed to satisfy RIP of order k. The second primitive is used when ℓ is a leaf in T_N^{full}, i.e. a node at depth $\log N$, in which case the algorithm needs to *estimate* $\widehat{x}_{\boldsymbol{f}_\ell}$ using the (ℓ, T)-isolating filter, see Algorithm 3 for a pseudocode. This requires only $O\left(2^{w_T(\ell)}\right)$ samples.

Unfortunately, as we have already pointed out, the algorithm in [KVZ19] works only for exactly k-sparse signals, and also demands cubic time and sample complexity. Our new toolkit shows that all three limitations can be remedied (though not completely simultaneously).

We also mention that a modified version of [Man95] can be employed to recover exactly k-sparse signals in $\widetilde{O}(k^3)$ time. The algorithm presented in [Man95] performs breadth-first search in the Cooley-Tukey FFT computation graph, rather than exploring by picking the lowest weight leaf. Opposed to [KVZ19], the algorithm in [Man95] uses Dirac comb filters to learn all the non-empty frequency cones in the same level at once. However, the techniques in that paper cannot go beyond cubic time for k-sparse signals, and as can be seen in [Man95, Section 6], extending the result to robust signals pays a multiplicative signal-to-noise ratio factor on top of k^3.

5.2   Our Techniques.

Our first technique is a way to traverse the Cooley-Tukey FFT computation graph in almost quadratic time complexity. This was presented in detail in Section 3. Here we give a quick summary of our FFT tree exploration.

FFT backtracking.   The first crucial observation is that the vanilla FFT traversal algorithm given in [KVZ19] performs a *zero test* with RIP of order k to decide whether a subtree contains a non-zero frequency, and this might be unnecessary. Indeed, if we are at a node v for which $\|\widehat{x}_{\mathrm{FreqCone}(v)}\|_0 = O(1)$, i.e. there are at most $O(1)$ elements in $\mathrm{FreqCone}(v)$, we only need to perform RIP of order $O(1)$. Thus, maybe there is a way to approximately learn $\|\widehat{x}_{\mathrm{FreqCone}(v)}\|_0$, for nodes v explored during the execution of the algorithm, and perform a *low-budget* zero test accordingly?

We have demonstrated that this intuition is correct in Section 3. The idea is to assign varying budgets to the nodes to be explored and then perform the hierarchical error detection in order to detect errors in the exploration which are caused by the failures of ZeroTest due to incorrect budget assignments. The algorithm maintains at all times a subtree T, as well as a vector $\widehat{\chi}$, such that $\mathrm{supp}(\widehat{x} - \widehat{\chi}) \subseteq \cup_{u \in T} \mathrm{FreqCone}(u)$, and $\mathrm{supp}(\widehat{\chi}) \subseteq \mathrm{supp}(\widehat{x})$. The algorithm explores the tree by considering values $b_1, b_2 \ldots$, corresponding to the possible assumptions on the sparsity of $\widehat{x}_{\mathrm{FreqCone}(v)}$, for some node v picked during the execution of the algorithm. For a parameter $\alpha < 1$ we

19

use thresholds $b_0 := k, b_1 := \alpha k, b_2 := \alpha^2 k, \ldots, b_{\frac{\log k}{\log(1/\alpha)}} = O(1)$. Our algorithm recursively explores various subtrees T_v with some budget $b := b_j$, i.e. under the assumption $\|\widehat{x}_{\mathrm{FreqCone}(v)}\|_0 \leq b$. The algorithm maintains a subtree T_v, initialized at $\{v\}$ and proceeds by picking the minimum weight node $z \in T_v$ and considering the two children of z, let them be $z_{\mathrm{left}}, z_{\mathrm{right}}$. Then, it runs itself recursively on $T_{z_{\mathrm{left}}}, T_{z_{\mathrm{right}}}$ with budget $b_{j+1} = \alpha b$. When the recursive calls return, yielding candidate vectors $\widehat{\chi}_{\mathrm{left}}, \widehat{\chi}_{\mathrm{right}}$, it performs a zero test on each of $z_{\mathrm{left}}, z_{\mathrm{right}}$ with RIP of order b, in order to check whether $\widehat{x}_{\mathrm{FreqCone}(z_{\mathrm{left}})} - \widehat{\chi}_{\mathrm{left}}$ is the all zeros vector (similarly for the right child). If the zero test on z_{left} is False, we add z_{left} to T_v; similarly for z_{right}. If both zero tests are True, then we remove z. This continues either until $T_v = \varnothing$ or until the number of nodes that have ever been inserted in T_v becomes too large (in particular if there is $\Omega(b/\alpha)$ leaves). In the first case, the algorithm returns the found vector, otherwise it returns the all zeros vector, since insertion of too many nodes into T_v means that we have underestimated the sparsity of $\widehat{x}_{\mathrm{FreqCone}(v)}$, as we argued in Section 3.

Upon performing a call with arguments a node v and a budget b, it could be the case that $\|\widehat{x}_{\mathrm{FreqCone}(v)}\|_0 \leq b$ does not hold; however, this misassumption is not detected by that call, and a vector which is not equal to $\widehat{x}_{\mathrm{FreqCone}(v)}$ is returned to the above recursion level. Nevertheless, although undetectable at the time, this discrepancy will be detected in some recursion level above, where we make use of higher budget; definitely at the very first level where we perform RIP of order k. We proved the correctness of the above process in Section 3 using induction on the tree.

Robust Algorithm. Our tree exploration technique works well for solving the exact Sparse FFT problem. For designing a robust algorithm we need a collection of new techniques in addition to the FFT backtracking. In what follows we explain the techniques needed for *robustifying* our Sparse FFT algorithm.

First of all, in the robust case we should substitute ZeroTest with an analogous HeavyTest routine. The role of this routine is to determine whether $\|(\widehat{x} - \widehat{\chi})_{\mathrm{FreqCone}(v)}\|_2 \geq \|\widehat{\eta}\|_2$, where v is any node that appears during the execution of the algorithm. If the latter inequality holds, this means that there are elements of the head of $\widehat{x}$ inside $\mathrm{FreqCone}(v)$ that are yet to be recovered. Pseudocode for this routine is presented in Algorithm 4, and the guarantees of this routine are spelled out in Lemma 17. The algorithm is very similar to ZeroTest, with the difference that we now need to take a collection of random samples, since a deterministic collection of samples satisfying RIP does not suffice to control the non-sparse component, i.e. the contribution of the tail under filtering. Furthermore, what is demanded is a control on how a (v, T)-isolating filter $\widehat{G}$ acts on $\widehat{x}_{\cup_{u \in T \setminus \{v\}} \mathrm{FreqCone}(u)}$, i.e. on parts of the signal living inside frequency cones which u is *not* isolated from. In words, one would like to appropriately control the energy of $\left(\widehat{G} \cdot \widehat{x}_{\cup_{u \notin T} \mathrm{FreqCone}(u)}\right)$, where $\cdot$ corresponds to element-wise vector multiplication.

Collectively, adaptive aliasing filters act as near-isometries. Adaptive aliasing filters are particularly effective for *non-obliviously* isolating elements of the head with respect to each other. However, in standard sparse recovery tasks, one desires control of the tail energy that participates in the measurement. This is a relatively easy (or at least well-understood) task in Sparse Fourier schemes which operate via ℓ_∞-box filters [HIKP12a, HIKP12b, IKP14, IK14, Kap17], but a non-trivial task using adaptive aliasing filters. The reason is that the tail via the latter filtering is hashed in a *non-uniform* way. The hashing depends on the arithmetic structure of the elements used to construct the filters, as well as their arithmetic relationship with the elements in the tail. This non-uniformity is essentially the main driving reason for the "exactly k-sparse" assumption

in [KVZ19]. Our starting point is the observation that for every tree $T \subseteq T_N^{\text{full}}$, the (v, T)-isolating filters for $v \in \text{LEAVES}(T)$, satisfy the following orthonormality condition in dimension one, see subsection 11.1.

Lemma 5. *(Gram Matrix of adaptive alliasing filters in $d = 1$) Let $T \subseteq T_n^{\text{full}}$, let G_v be the (v, T)-isolating filter of leaf $v \in \text{LEAVES}(T)$, as per (4). Let v and v' be two distinct leaves of T. Then,*

1.

$$\|\widehat{G}_v\|_2^2 := \sum_{\xi \in [n]} |\widehat{G}_v(\xi)|^2 = \frac{n}{2^{w_T(v)}}.$$

2. *(cross terms) the adaptive aliasing filters corresponding to v and v' are orthogonal, i.e.*

$$\langle \widehat{G}_v, \widehat{G}_{v'} \rangle := \sum_{\xi \in [n]} \widehat{G}_v(\xi) \cdot \overline{\widehat{G}_{v'}(\xi)} = 0.$$

This already postulates that adaptive aliasing filters are relatively well-behaved: for a tree T all leaves of which have roughly the same weight, it must be the case that $x \mapsto \{\langle \widehat{G}_v, \widehat{x} \rangle\}_{v \in \text{LEAVES}(T)}$ is a near-orthonormal transformation. Of course, this is too much to ask in general. The crucial property that we will make use of is captured in the following Lemma, see Subsection 11.2.

Lemma 6. *(see Lemma 16) Consider a tree $T \subseteq T_N^{full}$. For every leaf v of T we let $\widehat{G}_v$ be a Fourier domain (v, T)-isolating filter. Then for every $\boldsymbol{\xi} \in [n]^d$,*

$$\sum_{v \in \text{LEAVES}(T)} |\widehat{G}_v(\boldsymbol{\xi})|^2 = 1.$$

Using standard arguments, the above gives the following Lemma.

Lemma 7. *For $z : [n]^d \to \mathbb{C}$, let $z^{\to \boldsymbol{a}}$ be the cyclic shift of z by a, i.e. $z^{\to \boldsymbol{a}}(\boldsymbol{f}) := z(\boldsymbol{f} - \boldsymbol{a})$, where the subtraction happens modulo n in every coordinate. For a tree $T \subseteq T_N^{\text{full}}$,*

$$\mathbb{E}_{\boldsymbol{a} \sim U_{[n]^d}} \left[\sum_{v \in \text{LEAVES}(T)} |\langle \widehat{G}_v, \widehat{z^{\to \boldsymbol{a}}} \rangle|^2 \right] = \|z\|_2^2,$$

i.e. on expectation over a random shift the total collection of filters is an isometry.

Thus, although the tail is hashed in a way that is dependent on the head of the signal, what we can prove is that in expectation over a random shift the total amount of noise is controllable. Using the last property we can ensure that HEAVYTEST in the high-SNR regime we consider i) does not introduce *false positives*, i.e. does not engage in exploration in subtrees that contain no sufficient amount of energy, and ii) prevents false negatives. Guarantee i) translates to a bound on the running time of the algorithm, while ii) ensures correct execution of the algorithm. Note that due to the explorative nature of algorithm and the fact that missing a heavy element *increases the total noise in the system* (since we stop isolating with respect to it afterwards, it contributes as noise in subsequent measurements), accumulation of false negatives can totally destroy the guarantees of our approach. We note that this phenomenon of the tail not hashed independently of the signal occurs also in one-dimensional continuous Sparse Fourier Transform [PS15], although for a very different reason; in their setting handling such an irregularity is significantly easier, mostly due to the fact that errors do not accumulate as in our explorative algorithm.

Identification and estimation are interleaved. In contrast to more standard sparse recovery tasks where usually identification and estimation can be decoupled, our algorithm needs to have a precise way to perform estimation upon identification of a coordinate. That happens due to the explorative nature of our algorithm, which does not allow us to perform estimation at the very end. This is relatively easy in the exactly k-sparse case, but in the robust case, due to the presence of noise it is much more challenging. Whenever we identify a frequency and isolate it from the other head elements, we can pick $\widetilde{O}(k)$ random samples and estimate it up to $1/\sqrt{k}$ fraction of the tail energy. Although this precision is sufficient for our algorithm to go through, it would lead us to an undesirable *cubic* sample complexity in total. The next two techniques are introduced in order to handle this situation.

Lazy Estimation. One additional crucial difference between the exactly k-sparse case and the robust case is estimation. In the former, when we had a tree T and the minimim-weight leaf $v \in T$ was also a leaf in T_N^{full}, we needed $\widetilde{O}(2^{w_T(v)})$ samples in order to perfectly estimate $\widehat{x}_{\boldsymbol{f}_v}$. However, in the robust case, perfect estimation is impossible, and as is usual in sparse recovery tasks, we should estimate it up to additive error $O\left(\frac{1}{\sqrt{k}}\|\widehat{\eta}\|_2\right)$ (recall that we write $\widehat{x} = \widehat{w} + \widehat{\eta}$, where η is the tail of the signal). One way to achieve this type of guarantee is to take $\widetilde{O}(k)$ random samples from $G_v \star x$, where G_v is the (v, T)-isolating filter. This would yield $\widetilde{O}(k \cdot 2^{w_T(v)})$ samples for estimation, a k factor worse than what is needed in the exactly k-sparse case. In total, the sample complexity (and running time) would be k times more expensive, getting us back to $\widetilde{O}(k^3)$.

Let's see how it is possible to shave the aforementioned multiplicative k factor in the sample complexity. Imagine that upon finding such a leaf v, our algorithm *does not* estimate it immediately, but rather decides to postpone estimation for later. Instead, it marks it as a fully identified frequency, without removing it from T and proceeds in exploring T further. From now on, instead of picking the lowest weight leaf in T at any time, it picks the lowest weight *unmarked* leaf in T. Of course, it could be the case that this rule causes the leaf picked to have weight much more than $\log k$, significantly increasing the cost of filtering. Consider however the following strategy. While the minimum weight unmarked leaf in T has weight at most $\log k + 2$, we pick and it and continue exploring. Whenever the aforementioned condition does not hold, the total Kraft mass[8] occupied by the *marked* leaves in T is at least $1 - k \cdot \frac{1}{2k} = \frac{1}{2}$. When this happens, we show that we can extract a large subset of the marked nodes, see Lemma 11, which can be well-estimated *on average* using only a polylogarithmic number of samples. This suffices for the ℓ_2/ℓ_2 guarantee, and furthermore reduces the number of marked nodes (and hence the Kraft mass occupied by marked nodes) causing our algorithm to proceed without increasing the cost of filtering. A more involved demonstration of this idea appears in section 12.

Multi-scale Estimation. The lazy estimation technique presented above can estimate k heavy frequencies of $\widehat{x}$ up to average additive error of $O\left(\frac{\|\widehat{\eta}\|_2}{\sqrt{k}}\right)$ using quadratic samples only if we use the vanilla tree exploration strategy which always picks the lowest weight unmarked leaf of tree T and explores its children. This exploration strategy ensures that leaves get identified and consequentky *marked* in ascending weight order. Thus, there will be a point where the Kraft mass occupied by marked leaves is sufficiently large (recall that marked leaves have weight bounded by $\log k + 2$). However, as we already mentioned, the tree exploration employed in [KVZ19] results in cubic sample complexity even in the exactly k-sparse case. On the other hand, our new exploration

---

[8]For a tree T and a set $S \subseteq \mathrm{LEAVES}(T)$ we shall refer to the quantity $\sum_{v \in S} 2^{-w_T(v)}$ as the Kraft mass occupied by S in T, or just the Kraft mass of S if it is clear from context.
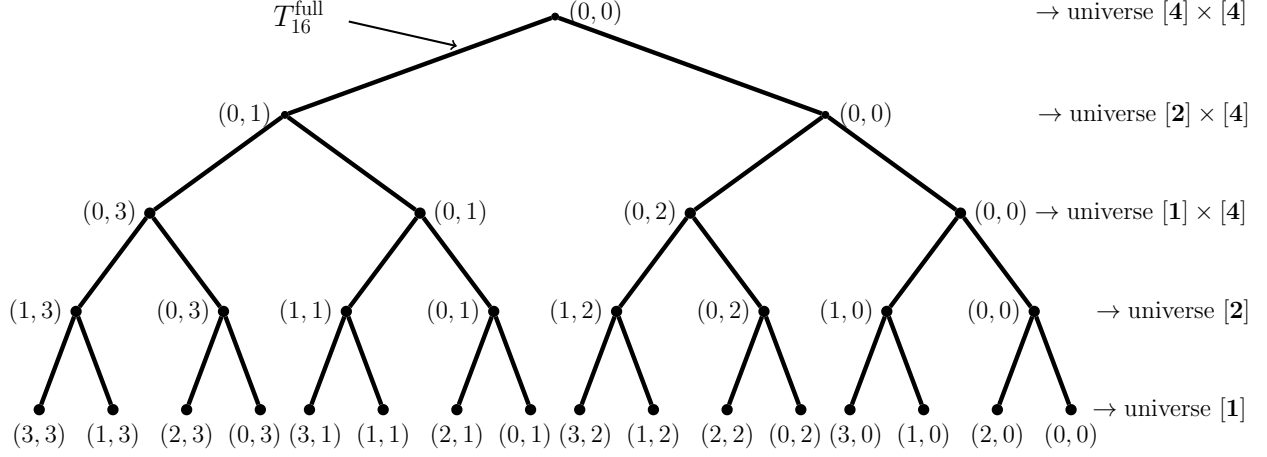
Figure 3: An example of the FFT binary tree T_N^{full} with $n = 4$ and dimension $d = 2$, (thus $N = 16$). The universe corresponding to the nodes at each level of the tree is shown on the right side and the labeles of each node appears next to it.

strategy (FFT backtracking) does not necessarily guarantee that the identified leaves will have large Kraft mass and bounded weight at the same time.

To make both lazy estimation and backtracking tree exploration techniques work together and achieve near quadratic total sample complexity, we devise a multi-scale estimation scheme. Our estimation strategy is to estimate every heavy frequency not once, but multiple times, each time to a different accuracy. More precisely, let's assume we are exploring a node $v \in T$ under the assumption that $\|\widehat{x}_{\text{FreqCone}(v)}\|_0 \leq b$, and this assumption is correct. For every found frequency $\boldsymbol{f}$, we estimate $\widehat{x}(\boldsymbol{f})$, to precision $\frac{\|\widehat{\eta}\|_2}{\sqrt{b}}$ instead of $\frac{\|\widehat{\eta}\|_2}{\sqrt{k}}$, which would be the standard thing to do. However, sticking to this error precision will not give the desired ℓ_2/ℓ_2 guarantee: for small b, it blows up the error by a factor of $\sqrt{\frac{k}{b}}$, and it could be that all $\boldsymbol{f} \in \text{supp}(\widehat{x})$ are estimated in a low-budget subproblem, due to recursion. Nevertheless, we can use these coarse-grained estimates to *only locate* the support of $\widehat{x}$ inside a subtree, and return it to the parent subproblem, i.e. to the above recursion level. The parent subproblem will mark those recovered frequencies, ignore their values, and continue its execution normally (pick the lowest leaf, perform lazy estimation etc). At some point, when the Kraft mass occupied by the parent subproblem is large enough, those frequencies will be estimated up to *higher* precision, i.e. $\frac{\|\widehat{\eta}\|_2}{\sqrt{b/\alpha}}$. When it finishes execution, it will return those elements to the above recursion level, so on so forth. This type of argumentation can be used to glue together lazy estimation and FFT backtracking. An illustration of this idea takes place in Section 13.

5.3 Explanation of the barriers faced.

Discussion on the limits of the explorative approach, or why the quadratic barrier is impenetrable. On a high level, the explorative approach we take maintains a vector $\widehat{\chi}$ such that $\text{supp}(\widehat{\chi}) \subseteq \text{supp}(\widehat{x})$ at all times[9]. Whenever the algorithm reaches a leaf $v \in T_N^{\text{full}}$ (see definitions in the Preliminaries Section), it estimates it and adds it to $\widehat{\chi}$. Subsequently, it proceeds by trying to recover the residual vector $\widehat{x} - \widehat{\chi}$. Now, imagine that we have recovered a constant fraction, say

---

[9]In fact, this is an oversimplification of our approach (as well as slightly inaccurate), but for the sake of discussion let us assume that this is the case.

23

1/10, of $\widehat{x}$, and want to proceed further in order to recover the remaining part of x, i.e. $\widehat{x} - \widehat{\chi}$, which is an $\Omega(k)$-sparse vector. In order even to test whether $\widehat{x} - \widehat{\chi}$ is the zero vector, we need to pick a set of $\Omega(k)$ random samples, satisfying for example the Restricted Isometry Property of order k, from $x - \chi$. In turn, this means that we need to compute the values χ_t for all t in the aforementioned collection of random samples, and subtract them from the corresponding values of x. Since both $\mathrm{supp}(\widehat{\chi})$ and the samples needed for RIP are in principle unstructured sets of size $\Omega(k)$, the computation of the relevant χ_t is exactly the classical non-equispaced Fourier transform, for which no strongly subquadratic algorithm in available. We explain this unavailability by providing a quadratic lower bound on this task based on the well-established Orthogonal Vectors hypothesis, see Theorem 2. This also provides evidence that the quadratic time barrier is the limit of our explorative approach. Indeed, at all times we need to decide whether to explore a subtree or not by testing whether $\widehat{x} - \widehat{\chi}$ is the zero vector projected on that subtree. Since subtracting the effect of $\widehat{\chi}$ from the measurements, i.e. evaluating χ on an unstructured set of samples, cannot be done in strictly subquadratic time unless OVH fails, a subquadratic algorithm for exactly k-sparse FFT by traversing a pruned Cooley-Tukey FFT computation tree would most likely yield a subquadratic algorithm for the Orthogonal Vectors problem.

Discussion on the high-SNR regime. We shall illustrate a potential scenario where we might miss most frequencies in the head of the signal if we run our algorithm on an input signal that is not in the high-SNR regime. Note that throughout the exploration algorithm, we always maintain a set of nodes, such that the union of the frequency cones of those nodes covers the head of the signal. The frequencies which are not covered are essentially treated as *noise*, and we do not isolate with respect to them. Due to the fact that the adaptive aliasing filters hash the noise in a non-uniform way, it could be that our HEAVYTEST primitive misclassifies a subtree as "frequency-inactive", i.e. no head element inside it, although it contains one. In such a scenario, it is natural to abandon exploration inside the subtree. This would cause the noise in the system to increate by the magnitude of the missed head element (since we shall not isolate with respect to it anymore). Subsequently, this can potentially lead to a chain reaction, leading to successively missing head elements, and successively increasing the noise in the system, ending up to not recovering anything. However, our HEAVYTEST primitive is strong and ensures that we never miss a heavy frequency of signals that are not in the high-SNR regime as long as we perform oversampling by a factor k.

On the other hand, note that in order to achieve the ℓ_2/ℓ_2 guarantee on signals that are not in high-SNR regime, we need to set the threshold of HEAVYTEST to $1/k$ fraction of the tail norm as opposed to the tail norm. Hence, another conceivable bad scenario is that, with such low threshold, the tail of the signal can make some frequency-inactive cones to appear heavy, introducing false positives. This can blow up the running time of the algorithm to super-polynomial in k.

Discrepancy between the runtime of our robust algorithm and its sample complexity. The only way we know how to perform dimension-independent estimation is via random sampling, as implemented in the HEAVYTEST routine. If we perform standard (non-lazy estimation) this would yield an additional multiplicative k factor, as claimed in the first paragraph of Techniques III. Remedying this via lazy estimation shaves the multiplicative k factor from the sample complexity, but does not do so in the running time. In particular, we run again into the same issue of subtracting $\widehat{\chi}$ from the buckets (which corresponds to an unstructured set of samples), i.e. the solution of a non-equispaced Fourier transform instance. As we've proven a quadratic time lower bound for the latter problem, this indicates that this discrepancy is most likely unavoidable with this approach.

6   Roadmap.

The roadmap of this paper is the following. We follow an incremental approach, trying to introduce the techniques one by one, to the extent that is possible. In Section 7 we revise adaptive aliasing filters from [KVZ19]. In Section 8 we give the facts related to Kraft's inequality which we are going to use throughout our algorithms. In Section 9 we formally prove that the exact Sparse FFT problem can be translated and reduced to the tree exploration problem and prove our first main result, i.e., Theorem 1. In Section 10 we give the conditional lower bound on non-equispaced Fourier transform. In Section 11, the new structural properties of adaptive aliasing filters are inferred. In section 12 we introduce our first robust Sparse Fourier transform algorithm, illustrating techniques II-III and partly technique I. Lastly, in Section 13 we obtain our final robust Sparse FT algorithm, which uses techniques I-IV. For that reason, the algorithm is presented last.

7   Machinery from Previous work: Adaptive Aliasing Filters.

In this section, we recall the class of adaptive aliasing filters that were introduced in [KVZ19]. These filters form the basis of our sparse recovery algorithm. For simplicity, we begin by introducing the filters in one-dimensional setting and then show how they naturally extend to the multidimensional setting (via tensoring).

7.1   One-dimensional Fourier transform.

Our algorithm extensively relies on binary partitioning the frequency domain. In $d = 1$, the following definitions are the one-dimensional analogues (special cases) of the ones in Section 4.1. We re-iterate them here, for completeness. The following is a re-interpretation of the *splitting tree* of a set in dimension 1.

Definition 9 (Splitting tree). For every $S \subseteq [n]$, the *splitting tree* $T = \text{Tree}(S, n)$ of a set S is a binary tree that is the subtree of T_n^{full} that contains, for every $j \in [\log n]$, all nodes $v \in T_n^{\text{full}}$ at level j such that $\{f \in S : f \equiv f_v \pmod{2^j}\} \neq \varnothing$.

Our Sparse FFT algorithm requires a filter G that satisfies a refined isolating property due to the fact that throughout the execution of the algorithm, the identity of $\text{supp}(\widehat{x})$ is only partially known. The following is a re-interpretation of the frequency cone of a node in dimension 1.

Definition 10 (Frequency cone of a leaf of T). Consider a subtree T of T_n^{full}, and vertex $v \in T$ which is at level $l_T(v)$ from the root, the *frequency cone of v with respect to T* is defined as,

$$\text{FreqCone}_T(v) := \left\{ f_u : \text{ for every leaf } u \text{ in subtree of } T_n^{\text{full}} \text{ rooted at } v \right\}.$$

Note that under this definition, the frequency cone of a vertex v of T corresponds to the subtree rooted at v when T is embedded inside T_n^{full}. Next we present the definition of an isolating filter, introduced in [KVZ19].

Definition 11 ((v, T)-isolating filter). Consider a subtree T of T_n^{full}, and leaf v of T, a filter $G : [n] \to \mathbb{C}^n$ is called (v, T)-*isolating* if the following conditions hold:

- For all $f \in \text{FreqCone}_T(v)$, we have $\widehat{G}(f) = 1$.

- For every $f' \in \bigcup_{\substack{u \in \text{LEAVES}(T) \\ u \neq v}} \text{FreqCone}_T(u)$, we have $\widehat{G}_{v'}(f') = 0$.

Note that in particular, for all signals $x \in \mathbb{C}^n$ with $\text{supp}(\widehat{x}) \subseteq \bigcup_{u \in \text{LEAVES}(T)} \text{FreqCone}_T(u)$ and $t \in [n]$,

$$\sum_{j \in [n]} x(j) G_v(t - j) = \frac{1}{n} \sum_{f \in \text{FreqCone}_T(v)} \widehat{x}_f e^{2\pi i \frac{ft}{n}}.$$

The main technical construction of [KVZ19] is captured by the following Lemma.

Lemma 8 (Filter properties, [KVZ19])**.** *Let n be an integer power of two, T a subtree of T_n^{full}, v a leaf in T. Let $f := f_v$ be the label of node v. Then the filter $G_v : [n] \to \mathbb{C}$ with Fourier Transform*

$$\widehat{G}_v(\xi) = \frac{1}{2^{w_T(v)}} \prod_{\ell \in \text{Anc}(v,T)} \left(1 + e^{2\pi i \frac{(\xi - f)}{2^{\ell+1}}} \right), \tag{4}$$

is a (v, T)-isolating filter. Furthermore,

- *$|\text{supp}(G_v)| = 2^{w_T(v)}$, and the filter G can be constructed in $O(2^{w_T(v)} + \log n)$ time (in the time domain).*

- *Computing $\widehat{G}_v(\xi)$ for $\xi \in [n]$ can be done in $O(\log n)$ time.*

7.2 d-dimensional Fourier transform.

In this subsection, we present the extension of adaptive aliasing filters to higher dimensions (by tensoring). It was shown in [KVZ19] that multidimensional construction of these filters is extremely efficient and incurs no loss in the dimensionality.

Definition 12 (Multidimensional (v, T)-isolating filter)**.** For every subtree T of T_N^{full} and vertex $v \in T$, a filter $G_v \in \mathbb{C}^{n^d}$ is called (v, T)-*isolating* if $\widehat{G}_v(\boldsymbol{f}) = 1$ for every $\boldsymbol{f} \in \text{FreqCone}_T(v)$ and $\widehat{G}_v(\boldsymbol{f}') = 0$ for every $\boldsymbol{f}' \in \text{supp}(T) \setminus \text{FreqCone}_T(v)$.

In particular, for every signal $x \in \mathbb{C}^{n^d}$ with $\text{supp}(\widehat{x}) \subseteq \text{supp}(T)$ and for all $\boldsymbol{t} \in [n]^d$,

$$\sum_{\boldsymbol{j} \in [n]^d} x(\boldsymbol{j}) G_v(\boldsymbol{t} - \boldsymbol{j}) = \frac{1}{N} \sum_{\boldsymbol{f} \in \text{FreqCone}_T(v)} \widehat{x}_{\boldsymbol{f}} e^{2\pi i \frac{\boldsymbol{f}^T \boldsymbol{t}}{n}}.$$

We need the following lemma which is the main result of this section and shows that isolating filters can be constructed efficiently.

Lemma 9 (Construction of a multidimensional isolating filter – Lemma 4.2 of [KVZ19])**.** *Let T of T_N^{full}, and consider $v \in \text{LEAVES}(T)$. There exists a deterministic construction of a (v, T)-isolating filter G_v such that*

1. *$|\text{supp}(G_v)| = 2^{w_T(v)}$.*

2. *G_v can be constructed in time $O\left(2^{w_T(v)} + \log N \right)$.*

3. *For any frequency $\boldsymbol{\xi} \in [n]^d$, $\widehat{G}_v(\boldsymbol{\xi})$, i.e. the Fourier transform of G_v at frequency $\boldsymbol{\xi}$, can be computed in time $O(\log N)$.*

8 Kraft-McMillan inequality and averaging claims.

For our needs, we are going to make use of the following standard claim from coding theory, referred to as Kraft's or Kraft-McMillan inequality. The most general version is an inequality, but in the case of binary trees (complete codes in coding theory vocabulary), it becomes an equality.

Theorem 10 (Kraft's equality). *Let $T \subseteq T_N^{\text{full}}$, it holds that*

$$\sum_{u \in \text{Leaves}(T)} 2^{-w_T(u)} = 1.$$

For a tree $T \subseteq T_N^{\text{full}}$ and a set $S \subseteq \text{Leaves}(T)$, we shall refer to the *Kraft mass* of S with respect to T as the quantity $\sum_{u \in S} 2^{-w_T(u)}$.

We shall frequently use the following straightforward Lemma, which we shall refer to as Kraft averaging. This ideas has appeared in [KVZ19].

Lemma 10 (Kraft averaging). *Let $T \subseteq T_N^{\text{full}}$, with L leaves. Then there exists a $u^* \in \text{Leaves}(T)$ such that $w_T(u^*) \leq \log_2 L$.*

The following fine-grained version of Kraft averaging is an indispensable building block of our lazy estimation technique, and constitutes one of the important departures from the approach in [KVZ19]. The reader may postpone reading it at the moment, since its first usage will be in section 12. Neverthless, we decided to keep all the claims regarding Kraft's inequality in a separate section, for compactness reasons.

Lemma 11 (Fine-grained Kraft Averaging). *Consider a subtree T of T_N^{full} and a positive integer b such that $|\text{Leaves}(T)| \leq b$. Let $S := \left\{ v \in \text{Leaves}(T) : 2^{w_T(v)} \leq 2b \right\}$, i.e. the leaves of T with weight at most $\log_2(2b)$. Then there exists a subset $L \subseteq S$ such that*

$$\frac{\max_{v \in L} 2^{w_T(v)}}{|L|} \leq \frac{1}{\theta},$$

where $\theta \leq \frac{1}{4 + 2\log_2 b}$.

Informally (but somewhat imprecisely), the claim postulates that for any subtree T of T_N^{full} with $|\text{Leaves}(T)| = k$, there exist either 1 node of weight 1, or 2 nodes of weight of 2, or ... at least $2^j / \log k$ nodes of weight j, or ... $k / \log k$ nodes of weight $\log k$. We now proceed with its proof.

Proof. First note that one can show the preconditions of claim imply that $\sum_{u \in S} 2^{-w_T(u)} \geq \frac{1}{2}$. For every $j = 0, 1, \ldots \lceil \log_2(2b) \rceil$, let L_j denote the subset of S defined as $L_j := \{u : u \in S, w_T(u) = j\}$. We can write,

$$\sum_{u \in S} 2^{-w_T(u)} = \sum_{j=0}^{\lceil \log_2(2b) \rceil} \frac{|L_j|}{2^j}$$

Therefore by the assumption of the claim, we have that there must exist a $j \in \{0, 1, \ldots \lceil \log_2(2b) \rceil\}$ such that $\frac{|L_j|}{2^j} \geq \frac{1}{2\lceil \log_2(2b) \rceil}$. Because $\theta \leq \frac{1}{4 + 2\log_2 |S|}$, there must exist a set $L \subseteq S$ such that $|L| \geq \theta \cdot \max_{v \in L} 2^{w_T(v)}$. $\qquad\square$

9 Translation of Exactly k-sparse FFT to Tree Exploration.

This section is devoted to solving the exact Sparse FFT problem through translation and reduction of this problem to the tree exploration problem detailed in Section 3 and invoking Algorithm 1. Recall that in this problem, we try to recover the *values* written on the leaves of a full binary tree T_N^{full} using two procedures, ZEROTEST and ESTIMATE which satisfy the properties given in Assumption 1. For solving the Sparse FFT problem we let the full binary tree T_N^{full} be defined as per Section 4.1 and each of its leaf values be the Fourier coefficients of the input signal x associated with the frequency labels of the corresponding leaves. Given this tree construction we can implement the procedures ZEROTEST and ESTIMATEFREQ in a similar fashion to [KVZ19].

Algorithm 2 ZEROTEST$(T, x, \widehat{\chi}, v, s)$

1: $\boldsymbol{f} := \boldsymbol{f}_v$
2: $G_v \leftarrow$ the (v, T) isolating filter as in Lemma 9
3: $\text{RIP}_s :=$ a set of $O(s \log^3 N)$ samples, which suffice for s-RIP condition, see Theorem 9
4: $h_{\boldsymbol{f}}^{\Delta} \leftarrow \sum_{\boldsymbol{\xi} \in [n]^d} \left(e^{2\pi \frac{\boldsymbol{\xi}^{\top} \Delta}{n}} \cdot \widehat{\chi}(\boldsymbol{\xi}) \cdot \widehat{G}_v(\boldsymbol{\xi}) \right)$, for all $\Delta \in \text{RIP}_s$
5: $H_{\boldsymbol{f}}^{\Delta} \leftarrow \sum_{\boldsymbol{j} \in [n]^d} x(\boldsymbol{j}) G_v(\Delta - \boldsymbol{j}) - h_{\boldsymbol{j}}^{\Delta}$, for all $\Delta \in \text{RIP}_s$
6: **if** $\sum_{\Delta \in \text{RIP}_s} |H_{\boldsymbol{f}}^{\Delta}|^2 = 0$ **then**
7:     **return** True
8: **else**
9:     **return** False

Algorithm 3 ESTIMATEFREQ$(T, x, \widehat{\chi}, v)$

1: $f := \boldsymbol{f}_v$
2: $G_v \leftarrow$ the (v, T) isolating filter as in Lemma 9
3: $h_{\boldsymbol{f}} \leftarrow \sum_{\xi \in [n]^d} \left(\widehat{\chi}(\boldsymbol{\xi}) \cdot \widehat{G}_v(\boldsymbol{\xi}) \right)$
4: Return $N \cdot \sum_{\boldsymbol{j} \in [n]^d} x(\boldsymbol{j}) G_v(-\boldsymbol{j}) - h_{\boldsymbol{j}}$

For detailed proofs we refer the reader to [KVZ19]. First we present the performance guarantee of the procedure ESTIMATEFREQ given in Algorithm 3 which can be proved by the filter isolation properties given in Lemma 9.

Lemma 12. *(Estimation) For any signals $x, \widehat{\chi}$, any tree $T \subseteq T_N^{\text{full}}$ such that $\text{supp}(\widehat{x} - \widehat{\chi}) \subseteq \text{supp}(T)$, and any leaf $\ell \in T$ which is also a leaf of T_N^{full}, the procedure $\text{ESTIMATEFREQ}(T, x, \widehat{\chi}, \ell)$ returns $(\widehat{x} - \widehat{\chi})(\boldsymbol{f}_\ell)$. Furthermore, the routine requires*

- $O\left(2^{w_T(\ell)}\right)$ *sample complexity, and*

- $\widetilde{O}(\|\widehat{\chi}\|_0 + 2^{w_T(\ell)})$ *running time.*

Next we present the performance guarantee of the procedure ZEROTEST given in Algorithm 2 which was also proved in [KVZ19].

Lemma 13. *(Testing whether a subtree is empty, see also [KVZ19, Lemma 7]) For any signals $x, \widehat{\chi}$, any tree $T \subseteq T_N^{\text{full}}$ such that $\text{supp}(\widehat{x} - \widehat{\chi}) \subseteq \text{supp}(T)$, and any leaf $v \in T$, if $\| (\widehat{x} - \widehat{\chi})_{\text{FreqCone}_T(v)} \|_0 \leq s$, then $\text{ZEROTEST}(T, x, \widehat{\chi}, v, s)$ determines correctly whether $\widehat{x}_{\text{FreqCone}_T(v)} = \widehat{\chi}_{\text{FreqCone}_T(v)}$ or not. Iff $\widehat{x}_{\text{FreqCone}_T(v)} = \widehat{\chi}_{\text{FreqCone}_T(v)}$, the primitive returns True. Furthermore, the routine requires*

- $O(2^{w_T(v)} \cdot |\mathrm{RIP}_s|)$ *sample complexity, and*

- $\widetilde{O}\left(\|\widehat{\chi}\|_0 \cdot |\mathrm{RIP}_s| + 2^{w_T(v)} \cdot |\mathrm{RIP}_s|\right)$ *running time* [10].

Recall that RIP_s *is a set of samples satisfying* s-RIP*, see Theorem 9, and* $|RIP_s| = O(s \log^3 N)$.

Given these primitives we can reduce the Sparse FFT problem to the tree exploration problem. More specifically, for any given signal $x \in \mathbb{C}^{n^d}$ we let T_N^{full} be a binary tree wth $N = n^d$ leaves such that the values of each leaf is the Fourier coefficient of the signal x at the frequency which corresponds to the lable of that leaf. The only catch here is that the tree exploration algorithm we developed in Section 3 relies on functions ZEROTEST and ESTIMATE which satisfy the properties given in Assumption 1, i.e., take as inputs Found and Excluded but the primitives in Algorithm 2 and Algorithm 3 operate on a tree T and a signal $\widehat{\chi} \in \mathbb{C}^{n^d}$. We show that in fact any Found and Excluded can be very efficiently translated to a tree T and a signal $\widehat{\chi}$. Using this translation, we can just invoke Algorithm 1 to solve the exact k-Sparse FFT problem in almost quadratic time and thus, prove Theorem 1.

Theorem 11 (Theorem 1, restated)**.** *The sparse Fourier transform problem with an exactly k-(Fourier sparse) signal* $x : [n]^d \to \mathbb{C}$*, i.e.,* $\|\widehat{x}\|_0 \leq k$ *can be solved in*

$$m = \widetilde{O}\left(k^2 \cdot 2^{8\sqrt{\log k \cdot \log \log N}}\right)$$

time, deterministically.

Proof. We prove the theorem by defining an appropriate binary tree T_N^{full} for any given k-Sparse signal $x \in \mathbb{C}^{n^d}$ and then invoking Algorithm 1 on T_N^{full} and then transforming the output to get the Fourier transform $\widehat{x}$. The first part is straightforward because for any given signal $x \in \mathbb{C}^{n^d}$ we let T_N^{full} be a binary tree wth $N = n^d$ leaves such that the values of each leaf $\ell \in \mathrm{LEAVES}(T_N^{\mathrm{full}})$ equals $\widehat{x}(\boldsymbol{f}_v)$. Because, $\|\widehat{x}\|_0 \leq k$, we can readily see that $|\mathrm{HEAVYLEAVES}(T_N^{\mathrm{full}})| \leq k$.

Next we have to show how to invoke Algorithm 1 to learn this tree efficiently. As was assumed in Section 3, for this algorithm to operate correctly it needs to have access to two primitives ZEROTEST and ESTIMATE which satisfy the properties given in Assumption 1. We show that the primitives given in Algorithm 2 and Algorithm 3 can be modified to satisfy the conditions of Assumption 1.

According to Assumption 1, these primitives take as input Found and Excluded, however, Algorithm 2 and Algorithm 3 take as input a tree T and signals $x, \widehat{\chi}$. The signal x is just the input signal in time domain and we can feed it to these procedures without any modifications. The signal $\widehat{\chi}$ is going to be constructed from Found very efficiently in time $O(|\mathsf{Found}|)$ as follows,

$$\widehat{\chi}_{\boldsymbol{f}} := \begin{cases} \mathsf{Found}(\ell) & \text{if } \boldsymbol{f} = \boldsymbol{f}_\ell \text{ for some leaf } \ell \in \mathrm{LEAVES}(T_N^{\mathrm{full}}) \\ 0 & \text{otherwise} \end{cases}.$$

We can also construct the tree T from Excluded efficiently as follows. We consider the path p in T_N^{full} from node v to the root. First we start with $T = p$. Then we iterate over every node $u \in T_N^{\mathrm{full}}$ which is a child of a node that belongs to the path p we check whether $\mathrm{LEAVES}(u) \cap \mathsf{Excluded} \neq \varnothing$ and if so we will add u to tree T. Thus, this tree can be constructed in time $\widetilde{O}(|\mathsf{Excluded}|)$.

---

[10]The $2^{w_T(v)} \cdot |\mathrm{RIP}_s|$ correspond to the number of accesses on x, and $\|\widehat{\chi}\|_0 \cdot |\mathrm{RIP}_s|$ corresponds to the time needed to subtract $\widehat{\chi}$ from the measurements. Lemma 7 in [KVZ19] has an additional third component, which corresponds to the time needed to prepare the isolating filter $\widehat{G}_v$. It is not hard to see that this third component can always be bounded by $O(\log N)$, and hence can be safely ignored.

Now we show that with the above translation of Found and Excluded to $\widehat{\chi}$ and tree T, Algorithm 2 and Algorithm 3 satisfy the conditions of Assumption 1. First note that the tree T that was constructed above is a subtree of $T(\mathsf{Excluded})$ therefore,

$$\mathrm{supp}(T(\mathsf{Excluded})) \subseteq \mathrm{supp}(T).$$

Using the above inequality and the way we defined the signal $\widehat{\chi}$ we have that, if a node v is isolated by Found and Excluded as per Definition 5 we have the following,

$$\mathrm{supp}(\widehat{x} - \widehat{\chi}) = \mathrm{HEAVYLEAVES}(T_N^{\mathrm{full}}) \setminus \mathrm{LEAVES}(\mathsf{Found}) \subseteq \mathrm{supp}(T(\mathsf{Excluded})) \subseteq \mathrm{supp}(T).$$

Furthermore, under the assumption that node v is isolated by Found and Excluded we have

$$\| (\widehat{x} - \widehat{\chi})_{\mathrm{FreqCone}_T(v)} \|_0 \leq |\mathrm{HEAVYLEAVES}(v)|.$$

Thus, using the above two inequalities we can invoke Lemma 13 to conclude that $\mathrm{ZEROTEST}(T, x, \widehat{\chi}, v, b)$ given in Algorithm 2 satisfies the conditions of the first part of Assumption 1. Note that the way we constructed the tree T implies that $w_{\mathsf{Excluded}}(v) = w_T(v)$, and also from the construction of $\widehat{\chi}$ one can easily see $|\mathsf{Found}| = \|\widehat{\chi}\|_0$. Therefore, the runtime of $\mathrm{ZEROTEST}(T, x, \widehat{\chi}, v, b)$ matches the desired runtime in Assumption 1.

The above argument also implies that for a leaf $\ell \in T_N^{\mathrm{full}}$ if ℓ is isolated by Found and Excluded then $T, x, \widehat{\chi}, \ell$ satisfy the preconditions of Lemma 12, thus $\mathrm{ESTIMATEFREQ}(T, x, \widehat{\chi}, \ell)$ given in Algorithm 3 satisfies the conditions of the second part of Assumption 1. Also the runtime of this procedure matches the desired runtime in Assumption 1.

Therefore, Algorithm 1 is applicable with our proposed translations, and by Theorem 6 and Theorem 8, this procedure finds $\widehat{x}$ perfectly and outputs correct estimates of the frequencies in time $\widetilde{O}\left(k^2 \cdot 2^{8\sqrt{\log k \cdot \log \log N}}\right)$. $\qquad\square$

10 Lower Bound on Non-Equispaced Fourier Transform.

The main result of this section is the following theorem.

Theorem 12. *(Detailed version of Theorem 2) For every $c > 0$ larger than an absolute constant and every $\delta > 0$ there exists $c' > 0$ and $\delta' > 0$ such that if for all $\epsilon \in (0, 1/2)$, for all N a power of two and all $k \leq 2^{c'(\log N)^{1/3}}$ there exists an algorithm that solves the 1-dimensional non-equispaced Fourier Transform problem on universe size N, sparsity k in time $k^{2-\delta'} \mathrm{poly}(\log(N/\epsilon))$, then there exists an algorithm which solves $\mathrm{OV}_{k,d}$ with $d = c \log k$ in time $k^{2-\delta}$.*

As also mentioned in the abstract of this paper, this answers one of the subproblems of Problem 21 from IITK Workshop on Algorithms for Data Streams, Kanpur 2006. Additionally, the following proof facilities gives also the lower bound on sparse multipoint evaluation, i.e. Theorem 3.

Proof. Given an Orthogonal Vectors instance, we shall appropriately construct a non-equispaced Fourier transform instance, such that an algorithm for the non-equispaced Fourier transform with strongly subquadratic running time in k implies a strongly subquadratic time algorithm for the Orthogonal Vectors problem.

Let $A = \{a_0, \ldots, a_{k-1}\}, B = \{b_0, \ldots, b_{k-1}\} \subseteq \{0, 1\}^d$ be the input to an $\mathrm{OV}_{k,d}$ instance with $d = c \log k$. We denote by $a_j(r)$ the r-th coordinate of vector $a_j \in A$. We first pick sufficiently large integers N, M, q that are powers of 2 such that $M = kd^{C_1 d}$, $q = C_2 d$, and $N = M^{2dq}$, where C_1, C_2 are sufficiently large absolute constants.

Next, we define for $j \in [k]$:

$$t_j := \sum_{r \in [d]} a_j(r) \cdot M^{rq}, \qquad f_j := \sum_{r \in [d]} b_j(r) \cdot \frac{N}{M^{rq+1}},$$

and set $F = \{f_0, \ldots, f_{k-1}\}, T = \{t_0, \ldots, t_{k-1}\}$. Furthermore, we define vector $x \in \mathbb{C}^N$ such that $x_t = 1$ if $t \in T$, and 0 otherwise, and we pick $\epsilon = \frac{1}{N}$. Thus, to transform our initial $\mathrm{OV}_{k,d}$ instance to an instance of non-equispaced Fourier transform, we show that from additive $\epsilon \|\widehat{x}\|_2$-approximations of $\widehat{x}_{f_0}, \ldots, \widehat{x}_{f_{k-1}}$ we can infer whether (A, B) contains a pair of orthogonal vectors. It then follows that an algorithm for non-equispaced Fourier transform running in time $k^{2-\delta'} \operatorname{poly}(\log(N/\epsilon))$ would imply a strongly subquadratic time algorithm for Orthogonal Vectors.

Our first claim postulates that $\widehat{x}_{f_j}$ corresponds to summing up $\exp\left(-2\pi i \cdot \frac{1}{M}\langle a_\ell, b_j\rangle\right)$ for all $\ell \in [k]$, up to error terms in the exponent.

Claim 1. *For every $j \in [k]$ it holds that*

$$\widehat{x}_{f_j} = \sum_{\ell \in [k]} \exp\left(-2\pi i \cdot \left(\tfrac{1}{M}\langle a_\ell, b_j\rangle + \xi_{\ell,j}\right)\right),$$

for a real number $\xi_{\ell,j}$ satisfying

$$|\xi_{\ell,j}| \leq \binom{d}{2} M^{-q-1}.$$

Proof. Fix $j \in [k]$ and note that

$$\widehat{x}_{f_j} = \sum_{t \in T} \exp\left(-2\pi i \frac{f_j t}{N}\right)$$

$$= \sum_{\ell \in [k]} \exp\left(-\frac{2\pi i}{N} \cdot \left(\sum_{r' \in [d]} a_\ell(r) \cdot M^{rq}\right) \cdot \left(\sum_{r \in [d]} b_j(r') \cdot \frac{N}{M^{r'q+1}}\right)\right)$$

$$= \sum_{\ell \in [k]} \exp\left(-2\pi i \cdot \sum_{(r,r') \in [d] \times [d]} a_\ell(r) b_j(r') \cdot M^{(r-r')q-1}\right)$$

$$= \sum_{\ell \in [k]} \prod_{(r,r') \in [d] \times [d]} \exp\left(-2\pi i \cdot a_\ell(r) b_j(r') \cdot M^{(r-r')q-1}\right)$$

We now investigate the exponents of the complex exponentials, namely $a_\ell(r) b_j(r') \cdot M^{(r-r')q-1}$ for $\ell \in [k]$ and $(r, r') \in [d] \times [d]$. In particular, we find that:

1. For any pair (r, r') with $r > r'$, we have $(r - r')q - 1 \geq 0$, meaning that the corresponding exponent is an integer multiple of $2\pi i$. In turn, the corresponding term in the product contributes 1, so it can be ignored.

2. For any pair (r, r') with $r < r'$ we have $(r - r')q - 1 \leq -q - 1$. For a fixed ℓ, there are $\binom{d}{2}$ such products, and hence their total contribution to the exponent of the ℓ-th summand is at most $\binom{d}{2} M^{-q-1}$ (in absolute value).

3. The pairs (r, r') with $r = r'$ contribute to the exponent of the ℓ-th summand the term $-2\pi i \cdot M^{-1} \sum_{r \in [d]} a_\ell(r) b_j(r) = -2\pi i \cdot M^{-1}\langle a_\ell, b_j\rangle$.

31

Putting everything together we arrive at the proof of the claim. □

In the remainder of this proof we write

$$V_{j,h} := \sum_{\ell \in [k]} \langle a_\ell, b_j \rangle^h.$$

Next, we perform a series expansion and error analysis on the exponential function to obtain:

Claim 2. *For every $j \in [k]$ it holds that*

$$\widehat{x}_{f_j} = \xi'_j + \sum_{h \geq 0} \left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot V_{j,h},$$

for a complex number ξ'_j satisfying

$$|\xi'_j| \leq M^{-q}.$$

Proof. Let a, b be real numbers. Starting from the basic fact $|\exp(-2\pi ib) - 1| \leq 2\pi|b|$, we obtain $\exp(-2\pi i(a + b)) = \exp(-2\pi ia) + \exp(-2\pi ia)(\exp(-2\pi ib) - 1) = \exp(-2\pi ia) + \xi'_{a,b}$ with $|\xi'_{a,b}| \leq 2\pi|b|$. In particular, with notation as in Claim 1, we have

$$\exp\left(-2\pi i \cdot \left(\tfrac{1}{M} \langle a_\ell, b_j \rangle + \xi_{\ell,j} \right) \right) = \exp\left(-2\pi i \cdot \tfrac{1}{M} \langle a_\ell, b_j \rangle \right) + \xi'_{\ell,j},$$

with $|\xi'_{\ell,j}| \leq 2\pi|\xi_{\ell,j}| \leq 2\pi \binom{d}{2} M^{-q-1} \leq M^{-q}$.
Summing over all $\ell \in [k]$ now yields

$$\widehat{x}_{f_j} = \sum_{\ell \in [k]} \exp\left(-2\pi i \cdot \left(\tfrac{1}{M} \langle a_\ell, b_j \rangle + \xi_{\ell,j} \right) \right) = \xi'_j + \sum_{\ell \in [k]} \exp\left(-2\pi i \cdot \tfrac{1}{M} \langle a_\ell, b_j \rangle \right),$$

with $|\xi'_j| \leq 2\pi k \binom{d}{2} M^{-q-1}$. Using that $M = kd^{C_1 d}$ for a sufficiently large constant $C_1 > 0$, we obtain $|\xi'_j| \leq M^{-q}$.
Finally, we use the series expansion of $\exp(.)$ to obtain

$$\widehat{x}_{f_j} = \xi'_j + \sum_{h \geq 0} \left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot \sum_{\ell \in [k]} \langle a_\ell, b_j \rangle^h.$$

□

We now show that in our expression for $\widehat{x}_{f_j}$ the summands $\left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot V_{j,h}$ lie sufficiently far apart, so that each summand can be reconstructed from an approximation of $\widehat{x}_{f_j}$.

Claim 3. *Let $j \in [k]$, $H \in [d]$, and let $\widetilde{x}_{f_j}$ be an additive $\epsilon\|\widehat{x}\|_2$ approximation of $\widehat{x}_{f_j}$. Then*

$$\widetilde{x}_{f_j} - \sum_{h=0}^{H-1} \left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot V_{j,h} = \left(-\tfrac{2\pi i}{M} \right)^H \tfrac{1}{H!} \cdot \left(V_{j,H} + \xi''_{j,H} \right),$$

for a complex number $\xi''_{j,H}$ satisfying

$$|\xi''_{j,H}| < 1/3.$$

Proof. Note that by Parseval's identity, we have $\|\widehat{x}\|_2 = \sqrt{N} \cdot \|x\|_2 = \sqrt{N \cdot k}$. Therefore, $|\widetilde{x}_{f_j} - \widehat{x}_{f_j}| \leq \epsilon \|\widehat{x}\|_2 \leq \epsilon \sqrt{N \cdot k} \leq \sqrt{k/N}$ as $\epsilon = 1/N$. Since $N = M^{2dq}$ and $M \geq k$, we obtain $|\widetilde{x}_{f_j} - \widehat{x}_{f_j}| \leq M^{-(q-1)}$.

Note that

$$\left| \sum_{h>H} \left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot V_{j,h} \right| \leq \sum_{h>H} \left(\tfrac{2\pi}{M} \right)^h \tfrac{1}{h!} \cdot \sum_{\ell \in [k]} \langle a_\ell, b_j \rangle^h$$

$$\leq \left(\tfrac{2\pi}{M} \right)^H \tfrac{1}{H!} \cdot \sum_{h>H} \left(\tfrac{2\pi}{M} \right)^{h-H} \cdot k \cdot d^h$$

$$= \left(\tfrac{2\pi}{M} \right)^H \tfrac{1}{H!} \cdot kd^H \cdot \sum_{h>H} \left(\tfrac{2\pi d}{M} \right)^{h-H}.$$

Since M is sufficiently larger than d, the latter sum can be bounded by $\frac{4\pi d}{M}$, and hence

$$\left| \sum_{h>H} \left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot V_{j,h} \right| \leq \left(\tfrac{2\pi}{M} \right)^H \tfrac{1}{H!} \cdot \tfrac{4\pi k d^{H+1}}{M} \leq \tfrac{1}{10} \cdot \left(\tfrac{2\pi}{M} \right)^H \tfrac{1}{H!}, \tag{5}$$

using the fact that $M = kd^{C_1 d}$ for a sufficiently large constant $C_1 > 0$ and $H \in [d]$.

We now, using Claim 2, decompose:

$$\widetilde{x}_{f_j} - \sum_{h=0}^{H-1} \left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot V_{j,h}$$

$$= \left(\widetilde{x}_{f_j} - \widehat{x}_{f_j} \right) + \left(\widehat{x}_{f_j} - \sum_{h=0}^{H-1} \left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot V_{j,h} \right)$$

$$= \left(\widetilde{x}_{f_j} - \widehat{x}_{f_j} \right) + \xi_j' + \left(-\tfrac{2\pi i}{M} \right)^H \tfrac{1}{H!} \cdot V_{j,H} + \sum_{h>H} \left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot V_{j,h}.$$

Recall that $|\widetilde{x}_{f_j} - \widehat{x}_{f_j}| \leq M^{-(q-1)}$ and $|\xi_j'| \leq M^{-q}$. We use $H \in [d]$ and our choice of $M = kd^{C_1 d}$ and $q = C_2 d$ for sufficiently large constants $C_1, C_2 > 0$ to conclude that $M^{-(q-1)} \leq \tfrac{1}{10} \cdot \left(\tfrac{2\pi}{M} \right)^H \tfrac{1}{H!}$. Together with inequality (5), this gives

$$\widetilde{x}_{f_j} - \sum_{h=0}^{H-1} \left(-\tfrac{2\pi i}{M} \right)^h \tfrac{1}{h!} \cdot V_{j,h} = \left(-\tfrac{2\pi i}{M} \right)^H \tfrac{1}{H!} \cdot \left(V_{j,H} + \xi_{j,H}'' \right),$$

for a complex number $\xi_{j,H}''$ with $|\xi_{j,H}''| < 1/3$. $\square$

Repeatedly applying the above claim allows us to reconstruct the numbers $V_{j,0}, \ldots, V_{j,d}$:

Claim 4. *Fix $j \in [k]$. Let $\epsilon = \tfrac{1}{N}$. Given an additive $\epsilon \|\widehat{x}\|_2 = \sqrt{k/N}$ approximation to $\widehat{x}_{f_j}$ we can infer the exact values of*

$$V_{j,h} := \sum_{\ell \in [k]} \langle a_\ell, b_j \rangle^h,$$

for any $h \in [d]$, in time $\mathrm{poly}(d, \log k)$.

Proof. Suppose that we have already computed the sums $V_{j,h}$ for all $0 \leq h < H$. Then we know the left hand side of Claim 3. Since $|\xi_{j,H}''| < 1/3$, there is a unique integer $V_{j,H} = \sum_{\ell \in [k]} \langle a_\ell, b_j \rangle^H$ that

33

satisfies the equation in Claim 3. Hence, we can infer $V_{j,H}$. Therefore, we can iteratively compute $V_{j,0}, V_{j,1}, \ldots, V_{j,d-1}$.

Note that when evaluating expressions of the form $\left(-\frac{2\pi i}{M}\right)^h \frac{1}{h!}$, we can compute them up to precision ϵ in time $\mathrm{poly}(d, \log k)$, since it suffices to perform arithmetic on numbers with $\mathrm{poly}(d, \log k)$ digits. This yields another additive error in the same order of magnitude as in the proof of Claim 3. The same error analysis therefore shows that this precision is sufficient to compute the exact integers $V_{j,h}$. $\qquad\square$

The above claim postulates that we can infer the values $V_{j,h}$ for $h \in [d]$. We next show that these values allow us to determine whether there exists a pair of orthogonal vectors.

Claim 5. *Given the values $V_h := V_{j,h}$ for all $h \in [d]$ and some fixed j, we can find out whether there exists an ℓ such that $\langle a_\ell, b_j \rangle = 0$, in time $\mathrm{poly}(d, \log k)$.*

Proof. This relies on the observation that we can write V_h as

$$V_h = \sum_{r=0}^{d-1} Z_r \cdot r^h$$

for

$$Z_r := |\{\ell \in [k] \mid \langle a_\ell, b_j \rangle = r\}|.$$

In other words, the values V_h are obtained from the values Z_r by multiplication with a Vandermonde matrix. Since this $d \times d$ matrix is invertible and all elements of this matrix and V_h are of value at most $k \cdot d^d$, we can infer the values Z_r from the values V_h in $\mathrm{poly}(d, \log k)$ time. Indeed, we can compute the inverse of this Vandermonde matrix multiplied by its determinant (so that the resulting matrix contains integer entries) using $\mathrm{poly}(d)$ operations on integers with $\mathrm{poly}(d, \log k)$ digits (each such operation takes $\mathrm{poly}(d, \log k)$ time). Multiplying the vector of V_h's by this matrix yields Z_r's multiplied by the determinant of the Vandermonde matrix, which can be computed and canceled using $\mathrm{poly}(d, \log k)$ operations by manipulating large integers with $\mathrm{poly}(d, \log k)$ number of digits. This yields the value $Z_0 = |\{\ell \mid \langle a_\ell, b_j \rangle = 0\}|$ and thus allows us to decide whether $b_j \in B$ is orthogonal to some vector in A. $\qquad\square$

Using Claims 4 and 5 over all Fourier evaluations $\{\widehat{x}_f\}_{f \in F}$ we can determine in time $k \cdot \mathrm{poly}(d, \log k)$ whether whether (A, B) contains an orthogonal pair. Thus, for $\delta \in (0, 1/2)$ an algorithm for non-equispaced Fourier transform running in time $k^{2-\delta'} \mathrm{poly}(\log(N/\epsilon))$ for $\epsilon = 1/N$, would imply the existence of a $k^{2-\delta'} \mathrm{poly}(d, \log k)$ time algorithm for $\mathrm{OV}_{k,d}$, since $\log(N/\epsilon) = 2 \log N = O(d^2) \log M = \mathrm{poly}(d, \log k)$ for any choice of constants $C_1, C_2 > 0$. For any constant $c > 0$, if dimension $d = c \log k$, this running time can be bounded by $O(k^{2-\delta})$ as long as $\delta' \geq 2\delta$, contradicting the Orthogonal Vectors Hypothesis (Conjecture 1). Finally, it remains to note that since $d = c \log k$ and

$$N = M^{2dq} = (kd^{C_1 d})^{2dq} = (c \log k)^{C_1 C_2 c^3 \log^3 k},$$

we have that $2^{c'(\log N / \log \log N)^{1/3}} \leq k \leq 2^{c''(\log N)^{1/3}}$ as long as c' is sufficiently small as a function of c, C_1, C_2, and c'' is sufficiently large as required.

$\qquad\square$

34

11 Robust analysis of adaptive aliasing filters.

This section is devoted to our technical innovation regarding adaptive aliasing filters. This a delicate analysis of how the filters act on an arbitrary vector. Such a robustification will be useful in order to control the amount of energy a measurement receives from the elements outside of the head. The absence of the properties derived in this section constitutes the restriction that has driven the "exactly k-sparse" assumption in [KVZ19].

11.1 One-dimensional case.

We first develop the appropriate machinery for the one-dimensional case. Generalizing the idea to higher dimensions can be done using tensoring, as we shall show in the next subsection. We first present a standalone computation of the Gram matrix of adaptive aliasing filters corresponding to a specific tree $T \subseteq T_n^{\text{full}}$.

Lemma 14. *(Gram Matrix of adaptive aliasing filters) Consider a tree $T \subseteq T_n^{\text{full}}$, and two distinct leaves v, v' of T. Let G_v (resp. $G_{v'}$) be the (v, T)-isolating (resp. (v', T)-isolating) filter, as per (4). Then,*

1. *(diagonal terms) the energy of the filter corresponding to v is proportional to $2^{-w_T(v)}$. In particular,*

$$\|\widehat{G}_v\|_2^2 := \sum_{\xi \in [n]} |\widehat{G}_v(\xi)|^2 = \frac{n}{2^{w_T(v)}}.$$

2. *(cross terms) the adaptive aliasing filters corresponding to v and v' are orthogonal, i.e.*

$$\langle \widehat{G}_v, \widehat{G}_{v'} \rangle := \sum_{\xi \in [n]} \widehat{G}_v(\xi) \cdot \overline{\widehat{G}_{v'}(\xi)} = 0.$$

Proof. We prove each bullet separately. Both bullets follow by symmetry considerations: cancellations that occur either by the fact that roots of unity cancel across a poset of a group, or by the sign change happening to specific complex exponentials at branching points of the tree T. The first one uses Kraft's equality.

Proof of Bullet 1. Let $f := f_v$ and $f' := f_{v'}$ denote the labels of v and v', respectively. By (4), we have

$$|\widehat{G}_v(\xi)|^2 = 4^{-w_T(v)} \cdot \prod_{\ell \in \text{Anc}(v,T)} \left(1 + e^{2\pi i \frac{\xi - f}{2^{\ell+1}}}\right) \cdot \left(1 + e^{-2\pi i \frac{\xi - f}{2^{\ell+1}}}\right)$$

$$= 4^{-w_T(v)} \cdot \prod_{\ell \in \text{Anc}(v,T)} \left(2 + e^{2\pi i \frac{\xi - f}{2^{\ell+1}}} + e^{-2\pi i \frac{\xi - f}{2^{\ell+1}}}\right)$$

$$= 4^{-w_T(v)} \cdot \sum_{\substack{S,T \subseteq \text{Anc}(v,T) \\ S \cap T = \varnothing}} 2^{|\text{Anc}(v,T)| - |S \cup T|} \cdot e^{2\pi i (\xi - f) \cdot \left(\sum_{\ell \in S} \frac{1}{2^{\ell+1}} - \sum_{\ell \in T} \frac{1}{2^{\ell+1}}\right)}$$

$$= 4^{-w_T(v)} \cdot \left(2^{w_T(v)} + \sum_{\substack{S,T \subseteq \text{Anc}(v,T) \\ S \cap T = \varnothing, S \cup T \neq \varnothing}} 2^{w_T(v) - |S \cup T|} \cdot e^{2\pi i (\xi - f) \cdot \left(\sum_{\ell \in S} \frac{1}{2^{\ell+1}} - \sum_{\ell \in T} \frac{1}{2^{\ell+1}}\right)}\right)$$

Note that the expression $\exprs_{S,T} = \sum_{\ell \in S} \frac{1}{2^{\ell+1}} - \sum_{\ell \in T} \frac{1}{2^{\ell+1}}$ inside the complex exponential can be 0 if and only if $S = T$, which is precluded by the fact that $S \cap T = \varnothing, S \cup T \neq \varnothing$. Thus, this gives rise to the exponential $e^{2\pi i(\xi - f) \cdot \exprs_{S,T}}$, which cancels out when summing over all ξ. Hence, we obtain that

$$\sum_{\xi \in [n]} |\widehat{G}_v(\xi)|^2 = \sum_{\xi \in [n]} 4^{-w_T(v)} \cdot \left(2^{w_T(v)} + 0\right) = \frac{n}{2^{w_T(v)}}.$$

Proof of Bullet 2. By (4), we have that

$$\langle \widehat{G}_v, \widehat{G}_{v'} \rangle = \sum_{\xi \in [n]} \widehat{G}_v(\xi) \cdot \overline{\widehat{G}_{v'}(\xi)}$$

$$= \sum_{\xi \in [n]} \left(\frac{1}{2^{w_T(v)}} \prod_{\ell \in \mathrm{Anc}(v,T)} \left(1 + e^{2\pi i \frac{\xi - f}{2^{\ell+1}}}\right) \right) \cdot \left(\frac{1}{2^{w_T(v')}} \prod_{\ell \in \mathrm{Anc}(v',T)} \left(1 + e^{-2\pi i \frac{\xi - f'}{2^{\ell+1}}}\right) \right)$$

$$= 2^{-w_T(v)-w_T(v')} \cdot \sum_{\substack{\xi \in [n] \\ S \subseteq \mathrm{Anc}(v,T) \\ S' \subseteq \mathrm{Anc}(v',T)}} \sum e^{2\pi i(\xi-f)\cdot\sum_{\ell \in S}\frac{1}{2^{\ell+1}} - 2\pi i(\xi-f')\cdot\sum_{\ell \in S'}\frac{1}{2^{\ell+1}}}$$

$$= 2^{-w_T(v)-w_T(v')} \cdot \sum_{\substack{S \subseteq \mathrm{Anc}(v,T) \\ S' \subseteq \mathrm{Anc}(v',T)}} \sum_{\xi \in [n]} e^{2\pi i(\xi-f)\cdot\sum_{\ell \in S}\frac{1}{2^{\ell+1}} - 2\pi i(\xi-f')\cdot\sum_{\ell \in S'}\frac{1}{2^{\ell+1}}}$$

$$:= 2^{-w_T(v)-w_T(v')} \cdot (A + B),$$

where A is sum of the terms that satisfy $S \neq S'$, and B is sum of terms satisfying $S = S'$. We will show that $A = B = 0$ separately. The equality $A = 0$ holds by a summation over all ξ and the fact that roots of unity cancel across a poset of a subgroup, whereas the equality $B = 0$ by a symmetry argument which exploits the sign change in the lowest common ancestor of v and v'.

Computing A. We will prove that if $S \neq S'$ then

$$\sum_{\xi \in [n]} e^{2\pi i(\xi-f)\cdot\sum_{\ell \in S}\frac{1}{2^{\ell+1}} - 2\pi i(\xi-f')\cdot\sum_{\ell \in S'}\frac{1}{2^{\ell+1}}} = 0,$$

which suffices to establish $A = 0$. Note that

$$e^{2\pi i(\xi-f)\cdot\sum_{\ell \in S}\frac{1}{2^{\ell+1}} - 2\pi i(\xi-f')\cdot\sum_{\ell \in S'}\frac{1}{2^{\ell+1}}} =$$

$$e^{2\pi i\xi\cdot\left(\sum_{\ell \in S}\frac{1}{2^{\ell+1}} - \sum_{\ell \in S'}\frac{1}{2^{\ell+1}}\right)} \cdot g,$$

where $g = e^{2\pi i f'\cdot\sum_{\ell \in S'}\frac{1}{2^{\ell+1}} - 2\pi i f\cdot\sum_{\ell \in S}\frac{1}{2^{\ell+1}}}$ does not depend on ξ. Summing over all $\xi \in [n]$ and taking into account that $\sum_{\ell \in S}\frac{1}{2^{\ell+1}} - \sum_{\ell \in S'}\frac{1}{2^{\ell+1}} \neq 0$ by the fact that $S \neq S'$, yields the desired result (the summation can also be viewed a summation of the roots of unity over $\frac{n}{2^{\max\{S \triangle S'\}}}$ copies of a poset of an additive subgroup of size $2^{\max\{S \triangle S'\}}$, where $\triangle$ denotes symmetric difference of sets).

36

Computing B. This quantity contains only terms corresponding to $S = S'$. Note that in this case $S \subseteq \text{Anc}(v, T) \cap \text{Anc}(v', T)$, and we have

$$B = \sum_{S \subseteq \text{Anc}(v,T) \cap \text{Anc}(v',T)} \sum_{\xi \in [n]} e^{2\pi i (f' - f) \cdot \sum_{\ell \in S} \frac{1}{2^{\ell+1}}} =$$

$$n \cdot \sum_{S \subseteq \text{Anc}(v,T) \cap \text{Anc}(v',T)} e^{2\pi i (f' - f) \cdot \sum_{\ell \in S} \frac{1}{2^{\ell+1}}}.$$

Let u be the lowest common ancestor of v, v' in tree T, i.e. the node on which the paths from the root to those two nodes split. Partition the powerset of $\text{Anc}(v, T) \cap \text{Anc}(v', T)$ to pair $(S, S \cup \{l_T(u)\})$, where $l_T(u) \notin S$. We shall prove that

$$e^{2\pi i (f' - f) \cdot \sum_{\ell \in S} \frac{1}{2^{\ell+1}}} + e^{2\pi i (f' - f) \cdot \sum_{\ell \in S \cup \{l_T(u)\}} \frac{1}{2^{\ell+1}}} = 0.$$

Indeed, by definition of u we have that $(f' - f) \equiv 2^{l_T(u)} \mod 2^{l_T(u)+1}$, which in turn gives that $e^{2\pi i (f' - f) \cdot \frac{1}{2^{l_T(u)+1}}} = e^{2\pi i \frac{2^{l_T(u)}}{2^{l_T(u)+1}}} = e^{\pi i} = -1$. This gives

$$e^{2\pi i (f' - f) \cdot \sum_{\ell \in S} \frac{1}{2^{\ell+1}}} + e^{2\pi i (f' - f) \cdot \sum_{\ell \in S \cup \{l_T(u)\}} \frac{1}{2^{\ell+1}}} =$$

$$e^{2\pi i (f' - f) \cdot \sum_{\ell \in S} \frac{1}{2^{\ell+1}}} \cdot \left(1 + e^{2\pi i (f' - f) \cdot \frac{1}{2^{l_T(u)+1}}} \right) = 0.$$

Thus, we conclude that $B = 0$, which finishes the proof of this Lemma. $\qquad\square$

The next lemma proves that for any tree T, the sum of squared values of adaptive aliasing filters corresponding to all leaves of T is equal to 1 at every frequency. The (v, T)-isolating filters for different leaves v of T can have very different behaviors and shapes in the Fourier domain, nevertheless, these filters collectively act as an isometry in the sense that the sum of their squared values is 1 everywhere in the Fourier domain.

Lemma 15. *(Total contribution of adaptive aliasing filters to one frequency) Consider a tree $T \subseteq T_n^{\text{full}}$. For every leaf v of T, let G_v denote the (v, T)-isolating filter as per (4), then it holds that*

$$\forall \xi \in [n]: \sum_{v \in \text{LEAVES}(T)} |G_v(\xi)|_2^2 = 1.$$

Proof. Fix $\xi \in [n]$. By (4), we have

$$\sum_{v \in \text{LEAVES}(T)} |\widehat{G_v}(\xi)|^2 =$$

$$\sum_{v \in \text{LEAVES}(T)} 4^{-w_T(v)} \cdot \prod_{\ell \in \text{Anc}(v,T)} \left| 1 + e^{2\pi i (\xi - f_v)/2^{\ell+1}} \right|^2 =$$

$$\sum_{v \in \text{LEAVES}(T)} 4^{-w_T(v)} \cdot \prod_{\ell \in \text{Anc}(v,T)} \left(2 + e^{2\pi i (\xi - f_v)/2^{\ell+1}} + e^{-2\pi i (\xi - f_v)/2^{\ell+1}} \right) =$$

$$\sum_{v \in \text{LEAVES}(T)} 2^{-w_T(v)} \cdot \prod_{\ell \in \text{Anc}(v,T)} \left(1 + \cos\left(2\pi (\xi - f_v)/2^{\ell+1} \right) \right) =$$

$$\sum_{v \in \text{LEAVES}(T)} 2^{-w_T(v)} \sum_{S \subseteq \text{Anc}(v,T)} \prod_{\ell \in S} \cos\left(2\pi \frac{\xi - f_v}{2^{\ell+1}} \right).$$

37

Thus, it suffices to prove that for all $\xi \in [n]$

$$\sum_{v \in \text{LEAVES}(T)} 2^{-w_T(v)} \sum_{S \subseteq \text{Anc}(v,T)} \prod_{\ell \in S} \cos\left(2\pi \frac{\xi - f_v}{2^{\ell+1}}\right) = 1. \tag{6}$$

We will implicitly interchange the summation between v and S in (6) and carefully group terms together so that most of them cancel out, due to the sign change in each branching point. In particular, fix a branching point, i.e. a node $u \in T$ with two children. We will estimate the contribution of all sets S such that $\max(S) = l_T(u)$ in (6). Let u_l be the left child of u in T, and let u_r be the right child of u in T. Note that,

$$\forall f \in \text{FreqCone}_T(u_l), f' \in \text{FreqCone}_T(u_r) : f - f' \equiv 2^{l_T(u)} \mod 2^{l_T(u)+1}.$$

In turn, this implies that for any $\xi \in [n]$ and any two f, f' as above we have: $(\xi - f) \equiv (\xi - f') + 2^{l_T(u)}$ mod $2^{l_T(u)+1}$, which gives the desired change in the branching point:

$$\cos\left(2\pi \frac{\xi - f}{2^{l_T(u)+1}}\right) = -\cos\left(2\pi \frac{\xi - f'}{2^{l_T(u)+1}}\right).$$

Thus, if we let T_r and T_l denote the subtrees of T rooted at u_r and u_l, respectively, then the total contribution of a set S that satisfies $\max(S) = l_T(u)$ and $S \subseteq \text{Anc}(v,T)$ for some leaf v of T to (6) can be expressed as

$$\prod_{\ell \in S \setminus \{l_T(u)\}} \cos\left(2\pi \frac{\xi - f_u}{2^{\ell+1}}\right) \cdot \left(\sum_{v \in T_r} \frac{1}{2^{w_T(v)}} - \sum_{v \in T_l} \frac{1}{2^{w_T(v)}}\right)$$

$$= \prod_{\ell \in S \setminus \{\text{br}\}} \cos\left(2\pi \frac{\xi - f_u}{2^{\ell+1}}\right) \cdot 2^{-w_T(u)} \left(\sum_{v \in T_r} \frac{1}{2^{w_{T_r}(v)}} - \sum_{v \in T_l} \frac{1}{2^{w_{T_l}(v)}}\right) = 0.$$

The latter holds since $\sum_{v \in T_r} \frac{1}{2^{w_{T_r}(v)}} = 1$ by Kraft's equality; similarly $\sum_{v \in T_l} \frac{1}{2^{w_{T_l}(v)}} = 1$.

Thus, we will get cancellation of the contribution of all non-empty sets S by summing over all branching points. On the other hand, the contribution of the empty set $S = \varnothing$ is exactly $2^{-w_T(v)}$, for each leaf v. The sum of all those contributions is 1, again by Kraft's equality, giving the lemma. $\square$

11.2 Extension to d dimensions.

We are now ready to proceed with the generalization of the robustness properties of the adaptive aliasing filters given in Section11.1 to high dimensions. The following lemma states that the isolating filters constructed in Lemma 9, collectively for all leaves, preserve (in particular, do not increase) the energy of a signal.

Lemma 16. *Consider a tree $T \subseteq T_N^{full}$. If for every leaf v of T we let $\widehat{G}_v$ be the Fourier domain (v, T)-isolating filter constructed in Lemma 9, then for every $\boldsymbol{\xi} \in [n]^d$,*

$$\sum_{v \in \text{LEAVES}(T)} |\widehat{G}_v(\boldsymbol{\xi})|^2 = 1.$$

Proof. The proof is by induction on the dimension d.

Base of induction: Lemma 15 precisely proves the inductive claim for $d = 1$.

Inductive step: Suppose that the inductive hypothesis holds for $d - 1$ dimensional isolating filters. Given this inductive hypothesis, we want to prove that the inductive claim holds for d dimensional filters. Let T be a subtree of T_N^{full}, where $N = n^d$. For every leaf v of tree T, let $v_0, v_1, \cdots v_l$ denote the path from root to v where v_0 is the root and $v_l = v$. We let p_v denote a vertex in T, defined as

$$p_v := \begin{cases} v_{\log_2 n} & \text{if } l_T(v) \geq \log_2 n \\ v & \text{otherwise} \end{cases}.$$

Now, we construct the tree T^* by making a copy of the tree T and then removing every node which is at distance more than $\log_2 n$ from the root. Let the nodes of T^* be labeled by projecting the labels of T to their first coordinate as follows,

for every node $u \in T^* : f_u = f_1$, where $(f_1, f_2, \cdots f_d)$ is the label of u in T.

One can easily verify that the set $P := \{p_v : v \in \text{LEAVES}(T)\}$ specifies the set $\text{LEAVES}(T^*)$. For every $u \in P$ let H_u be a (u, T^*)-isolating filter, constructed as in Lemma 9.

Moreover, for every leaf $u \in P$ we define T_u to be a copy of the subtree of T which is rooted at u. We label the nodes of the tree T_u by projecting the labels of T to their last $d - 1$ coordintates as follows,

for every node $z \in T_u : \boldsymbol{f}_z = (f_2, f_3, \cdots f_d)$, where $(f_1, f_2, \cdots f_d)$ is the label of u in T.

For every leaf v of T, let $\widehat{Q}_v$ be the Fourier domain (v, T_{p_v})-isolating filter constructed in Lemma 9. Note that in case $p_v = v$, the tree T_{p_v} will be empty and by convention we define our (v, T_{p_v})-isolating filter to be $\widehat{Q}_v \equiv 1$. Therefore, using these definitions, for every leaf $v \in \text{LEAVES}(T)$, the (v, T)-isolating filter $\widehat{G}_v$ constructed in Lemma 9 satisfies

$$\widehat{G}_v(\boldsymbol{\xi}) \equiv H_{p_v}(\xi_1) \cdot Q_v(\xi_2, \xi_3, \ldots \xi_d),$$

for every $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_d) \in [n]^d$. Hence, we can write

$$\sum_{v \in \text{LEAVES}(T)} \left| \widehat{G}_v(\boldsymbol{\xi}) \right|^2 = \sum_{v \in \text{LEAVES}(T)} |H_{p_v}(\xi_1) \cdot Q_v(\xi_2, \xi_3, \ldots \xi_d)|^2$$

$$= \sum_{u \in P} \sum_{\substack{v \in \text{LEAVES}(T) \\ \text{s.t. } p_v = u}} |H_u(\xi_1)|^2 \cdot |Q_v(\xi_2, \xi_3, \ldots \xi_d)|^2$$

$$= \sum_{u \in P} |H_u(\xi_1)|^2 \sum_{\substack{v \in \text{LEAVES}(T) \\ \text{s.t. } p_v = u}} |Q_v(\xi_2, \xi_3, \ldots \xi_d)|^2.$$

We proceed by proving that for every $u \in P$, $\sum_{\substack{v \in \text{LEAVES}(T) \\ \text{s.t. } p_v = u}} |Q_v(\xi_2, \xi_3, \ldots \xi_d)|^2 = 1$. Recall that for every leaf $v \in \text{LEAVES}(T)$, Q_v is a (v, T_{p_v})-isolating filter, constructed in Lemma 9. Therefore, for every leaf v of T such that $p_v = u$, Q_v is indeed a (v, T_u)-isolating filter as per the construction of Lemma 9. Hence,

$$\sum_{\substack{v \in \text{LEAVES}(T) \\ \text{s.t. } p_v = u}} |Q_v(\xi_2, \xi_3, \ldots \xi_d)|^2 = \sum_{v \in \text{LEAVES}(T_u)} |Q_v(\xi_2, \xi_3, \ldots \xi_d)|^2.$$

Now we can invoke the inductive hypothesis because T_u is a subtree of $T_{N'}^{full}$ where $N' = n^{d-1}$. therefore,

$$\sum_{\substack{v \in \text{LEAVES}(T) \\ \text{s.t. } p_v = u}} |Q_v(\xi_2, \xi_3, \ldots \xi_d)|^2 = \sum_{v \in \text{LEAVES}(T_u)} |Q_v(\xi_2, \xi_3, \ldots \xi_d)|^2 = 1.$$

Consequently, we have,

$$\sum_{v \in \text{LEAVES}(T)} \left| \widehat{G}_v(\boldsymbol{\xi}) \right|^2 = \sum_{u \in P} |H_u(\xi_1)|^2 = \sum_{u \in \text{LEAVES}(T^*)} |H_u(\xi_1)|^2 = 1,$$

where the last equality follows because H_u is a (u, T^*)-isolating filter as per the construction of Lemma 8 and hence by Lemma 15, $\sum_{u \in \text{LEAVES}(T^*)} |H_u(\xi_1)|^2 = 1$. This completes the inductive proof and ergo the Lemma. $\qquad\square$

We readily find that the following corollary of the above lemma holds,

Corollary 1. *The Fourier domain isolating filter $\widehat{G}$ constructed in Lemma 9 satisfies $\|\widehat{G}\|_\infty \leq 1$.*

12 Robust Sparse Fourier Transform I.

The section is devoted to proving our first result on robust Sparse Fourier transforms, which illustrates techniques II to IV and partially technique I. We first remind the reader about the high SNR regime we consider.

k-High SNR Regime. A vector $x : [n]^d \to \mathbb{C}$ satisfies the k-high SNR assumption, if there exists vectors $w, \eta : [n]^d \to \mathbb{C}$ such that i) $\widehat{x} = \widehat{w} + \widehat{\eta}$, ii) $\mathrm{supp}(\widehat{w}) \cap \mathrm{supp}(\widehat{\eta}) = \varnothing$, iii) $|\mathrm{supp}(\widehat{w})| \leq k$ and iv) $|\widehat{w}_f| \geq 3 \cdot \|\widehat{\eta}\|_2$, for every $f \in \mathrm{supp}(\widehat{w})$. In the rest of this section we prove the following main theorem.

Theorem 13 (Robust Sparse Fourier Transform). *Given oracle access to $x : [n]^d \to \mathbb{C}$ with $x = w + \eta$ in k-high SNR model and parameter $\epsilon > 0$, we can find using*

$$m = \widetilde{O}\left(k^{7/3} + \frac{k^2}{\epsilon}\right)$$

samples from x and in $\widetilde{O}\left(\frac{k^3}{\epsilon}\right)$ time a signal $\widehat{\chi}$ such that

$$\|\widehat{\chi} - \widehat{x}\|_2^2 \leq (1 + \epsilon) \cdot \|\widehat{\eta}\|_2^2,$$

with high probability in N.

For every tree T and node $v \in T$, we let $\widehat{x}_v$ be the vector $\widehat{x}_{\mathrm{FreqCone}(v)}$, i.e. signal $\widehat{x}$ supported on frequencies in the frequency cone of v and zeroed out everywhere else. At all times, for every $v \in T$, our algorithm maintains a signal $\widehat{\chi}_v : [n]^d \to \mathbb{C}$ that is supported on $\mathrm{FreqCone}_T(v)$. This signal will serve as our estimate for $\widehat{w}_v$. Initially, all these vectors are going to be $\{0\}^{n^d}$. The execution of our algorithm ensures that we can always keep sparse representations of those vectors. Parameters and variables n, d and $N = n^d$ are treated as global.

Furthermore, for any signal $y : [n]^d \to \mathbb{C}$ and parameter $\mu \geq 0$ we define

$$\mathrm{HEAD}_\mu(y) := \left\{ \boldsymbol{j} \in [n]^d : \ |y_{\boldsymbol{j}}| \geq 3\mu \right\}. \tag{7}$$

Under this notation, we are interested in recovering the set $\mathrm{HEAD}_{\|\widehat{\eta}\|_2}(\widehat{x})$, as well as obtain accurate estimations for the values of $\widehat{x}$ on frequencies in set $\mathrm{HEAD}_{\|\widehat{\eta}\|_2}(\widehat{x})$. Using the notion of $\mathrm{HEAD}_\mu(y)$, one can see that a signal x is in the k-high SNR regime iff there exists a $\mu > 0$ such that $|\mathrm{HEAD}_\mu(\widehat{x})| \leq k$ and $\mu \geq \left\|\widehat{x} - \widehat{x}_{\mathrm{HEAD}_\mu(\widehat{x})}\right\|_2$.

At all times, we keep a set Est, corresponding to the coordinates in $\mathrm{supp}(\widehat{w})$ that we have estimated. We define $L_v := \mathrm{FreqCone}_T(v) \cap (\mathrm{supp}(\widehat{w}) \setminus \mathrm{Est})$, which corresponds to the *unestimated* coordinates in the support of w that lie in the frequency cone of v.

Our main algorithm consists of an outer loop that we call ROBUSTSPARSEFFT and an inner loop that we call ROBUSTPROMISESFT. Our algorithm also makes use of an auxiliary primitive for estimating the values of located frequencies as well as a primitive for testing whether a signal is "heavy" (meaning that it contains a head element). In the rest of this section we first give the primitives ESTIMATE and HEAVYTEST together with the guarantee on their performance. Then we present the main algorithm and prove its performance. The HEAVYTEST routine is analogous to ZEROTEST from Section 6. However, the RIP property alone does not suffice (and hence we cannot pick a deterministic collection of samples). Instead, we use a random collection of samples, which suffices for upper bounding the contribution of the tail while simultaneously satisfying RIP.

12.1 Computational Primitives for the Robust Setting.

In this subsection we give some of the primitives that will be used in our algorithms. The proof of correctness of these primitives is postponed to subsection 12.3.

The very first primitive we present is HEAVYTEST, see Algorithm 4. This primitive performs a test on the signal to detect whether a given frequency cone contains heavy elements or not.

Algorithm 4 Test whether v is a frequency-active node, i.e. $\|\widehat{(x-\chi)}_v\|_2 > 2\|\widehat{\eta}\|_2$

1: **procedure** HEAVYTEST$(x, \widehat{\chi}, T, v, m, \theta)$
2:     $\boldsymbol{f} \leftarrow \boldsymbol{f}_v$
3:     $(G_v, \widehat{G}_v) \leftarrow$ MULTIDIMFILTER(T, v, n)
4:     $//(v,T)$-isolating filters as per Lemma 9
5:     **for** $z = 1$ to $32\log N$ **do**
6:         $\mathrm{RIP}^z_m \leftarrow$ Multiset of m i.i.d. uniform samples from $[n]^d$
7:
8:         $h^z_\Delta \leftarrow \sum_{\boldsymbol{\xi}\in[n]^d}\left(e^{2\pi i\frac{\boldsymbol{\xi}^\top\Delta}{n}}\cdot\widehat{\chi}(\boldsymbol{\xi})\cdot\widehat{G}_v(\boldsymbol{\xi})\right)$ for every $\Delta \in \mathrm{RIP}^z_m$
9:         $H^z \leftarrow \frac{1}{|\mathrm{RIP}^z_m|}\sum_{\Delta\in\mathrm{RIP}^z_m}\left|N\cdot\sum_{\boldsymbol{j}\in[n]^d}G_v(\Delta-\boldsymbol{j})\cdot x(\boldsymbol{j})-h^z_\Delta\right|^2$
10:     **if** MEDIAN$_{z\in[32\log N]}\{H^z\} \leq \theta$ **then**
11:         $//\theta = 5\|\widehat{\eta}\|^2_2$.
12:         **return** False
13:     **else**
14:         **return** True

Lemma 17 (HEAVYTEST guarantee). *Consider signals $x, \widehat{\chi} : [n]^d \to \mathbb{C}$ and an arbitrary subtree T of T^{full}_N. For an arbitrary leaf v of T, let $\widehat{y} := (\widehat{x} - \widehat{\chi}) \cdot \widehat{G}_v$, where $\widehat{G}_v$ be the Fourier domain (v,T)-isolating filter constructed in Lemma 9. Then the following statements hold, for any $\theta > 0$:*

- *If there exists a set $S \subseteq [n]^d$ such that $\|\widehat{y}_S\|^2_2 > \frac{11\theta}{10}$, then HEAVYTEST$(x, \widehat{\chi}, T, v, m, \theta)$ (Algorithm 4) outputs True with probability $1 - \frac{1}{N^{16}}$, provided that m is a large enough integer satisfying*

$$m = \Omega\left(|S|\cdot\frac{\|\widehat{y}\|^2_2}{\|\widehat{y}_S\|^2_2}\cdot\log^2|S|\log N\right).$$

- *If $\|\widehat{y}\|^2_2 \leq \theta/5$, then HEAVYTEST outputs False with probability $1 - \frac{1}{N^5}$.*

- *The sample complexity of this procedure is $\widetilde{O}\left(2^{w_T(v)}\cdot m\right)$.*

- *The runtime of the HEAVYTEST procedure is $\widetilde{O}\left(\|\widehat{\chi}\|_0\cdot m + 2^{w_T(v)}\cdot m\right)$.*

Next, we present the second auxiliary primitive ESTIMATE in Algorithm 4.

Lemma 18 (ESTIMATE guarantee). *Consider signals signals $x, \widehat{\chi} : [n]^d \to \mathbb{C}$, a subtree T of T^{full}_N, and an integer parameter m. For a subset $S \subseteq$ LEAVES(T), the procedure ESTIMATE$(x, \widehat{\chi}, T, S, m)$ (see Algorithm 5) outputs $\left\{\widehat{H}_v\right\}_{v\in S}$ such that*

$$\Pr\left[\sum_{v\in S}\left|\widehat{H}_v - \widehat{(x-\chi)}(\boldsymbol{f}_v)\right|^2 \leq \frac{16}{m}\sum_{\boldsymbol{\xi}\in[n]^d\setminus\mathrm{supp}(T)}\left|\widehat{(x-\chi)}(\boldsymbol{\xi})\right|^2\right] \geq 1 - \frac{|S|}{N^8}.$$

Algorithm 5 For $S \subseteq T$, estimates $(\widehat{x} - \widehat{\chi})_S$ by isolating S from every node in T.

1: **procedure** ESTIMATE$(x, \widehat{\chi}, T, S, m)$
2:      **for** $v \in S$ **do**
3:          $\boldsymbol{f} \leftarrow \boldsymbol{f}_v$
4:          $(G_v, \widehat{G}_v) \leftarrow$ MULTIDIMFILTER(T, v, n) $\triangleright$ (v, T)-isolating filters as per Lemma 9
5:          **for** $z = 1$ to $16 \log N$ **do**
6:              $\text{RIP}_m^z \leftarrow$ Multiset of B i.i.d. uniform samples from $[n]^d$
7:              $h_v^z \leftarrow \sum_{\Delta \in \text{RIP}_m^z} e^{-2\pi i \frac{\boldsymbol{f}^\top \Delta}{n}} \sum_{\boldsymbol{\xi} \in [n]^d} e^{2\pi i \frac{\boldsymbol{\xi}^\top \Delta}{n}} \cdot \widehat{\chi}(\boldsymbol{\xi}) \cdot \widehat{G}_v(\boldsymbol{\xi})$
8:              $H_v^z \leftarrow \frac{1}{|\text{RIP}_m^z|} \left(N \cdot \sum_{\Delta \in \text{RIP}_m^z} \left(e^{-2\pi i \frac{\boldsymbol{f}^\top \Delta}{n}} \sum_{\boldsymbol{j} \in [n]^d} G_v(\Delta - \boldsymbol{j}) \cdot x(\boldsymbol{j}) \right) - h_v^z \right)$
9:          $\widehat{H}_v \leftarrow$ MEDIAN$_{z \in [16 \log N]} \{ H_v^z \}$ $\triangleright$ Median of real and imaginary parts separately
10:      **return** $\left\{ \widehat{H}_v \right\}_{v \in S}$

The sample complexity of this procedure is $\widetilde{O}\left(m \cdot \sum_{v \in S} 2^{w_T(v)} \right)$ and the runtime of the procedure is $\widetilde{O}\left(m \cdot \sum_{v \in S} 2^{w_T(v)} + |S| \cdot m \cdot \|\widehat{\chi}\|_0 \right)$.

Lastly, we need the following primitive whose objective is to find a subset of identified leaves that are cheap to estimate *on average*.

Claim 6 (EXTRACTCHEAPSUBSET guarantee). *For every subtree T of T_N^{full} and every subset $S \subseteq$ LEAVES(T) that satisfies $\sum_{u \in S} 2^{-w_T(u)} \geq \frac{1}{2}$, the primitive EXTRACTCHEAPSUBSET(T, S) (see bottom of Algorithm 7) outputs a non-empty subset $L \subseteq S$ such that*

$$|L| \cdot (8 + 4 \log |S|) \geq \max_{v \in L} 2^{w_T(v)}.$$

12.2   Main Algorithm.

In this subsection we present our main sparse FFT algorithm. The algorithms consists of an outer loop and an inner loop. The outer loop, called ROBUSTSPARSEFT, always maintains a vector $\widehat{\chi}$ a tree FRONTIER such that

$$\text{HEAD}_\mu(\widehat{x} - \widehat{\chi}) \subseteq \cup_{u \in \text{FRONTIER}} \text{FreqCone}(u).$$

At every point in time, we explore the frequency cones of the low-weight FRONTIER by running the ROBUSTPROMISESFT algorithm. For the pseudocodes of the routines ROBUSTPROMISESFT and ROBUSTSPARSEFT, see Algorithms 6 and 7, respectively.

Overview of RobustPromiseSFT (Algorithm6):   Consider an invocation of ROBUSTPROMIS-ESFT$(x, \widehat{\chi}_{in}, \text{SIDETREE}, v, b, k, \mu)$. Suppose that $\widehat{y} := \widehat{x} - \widehat{\chi}_{in}$ is a signal in the k-high SNR regime, i.e., $\widehat{y}$ has k heavy frequencies and the value of each such heavy frequency is at least 3 times higher than the tail's norm. More formally, let HEAD $\subseteq [n]^d$ denote the set of heavy (head) frequencies of $\widehat{y}$ and suppose that $|\text{HEAD}| \leq k$, and the tail norm of $\widehat{y}$ satisfies $\|\widehat{y} - \widehat{y}_{\text{HEAD}}\|_2 \leq \mu$ and additionally suppose that $|\widehat{y}(\boldsymbol{f})| \geq 3\mu$ for every $\boldsymbol{f} \in$ HEAD. If SIDETREE fully captures the heavy frequencies of $\widehat{y}$, i.e., HEAD $\subseteq$ supp(SIDETREE), and the number of heavy frequencies in frequency cone of node v is bounded by b, i.e., $|\text{HEAD} \cap \text{FreqCone}_{\text{SIDETREE}}(v)| \leq b$, then ROBUSTPROMISESFT finds a signal $\widehat{\chi}_v$ such that supp$(\widehat{\chi}_v) =$ HEAD $\cap \text{FreqCone}_{\text{SIDETREE}}(v) := S$ and $\|\widehat{y}_S - \widehat{\chi}_v\|_2^2 \leq \frac{\mu^2}{20}$. An example of
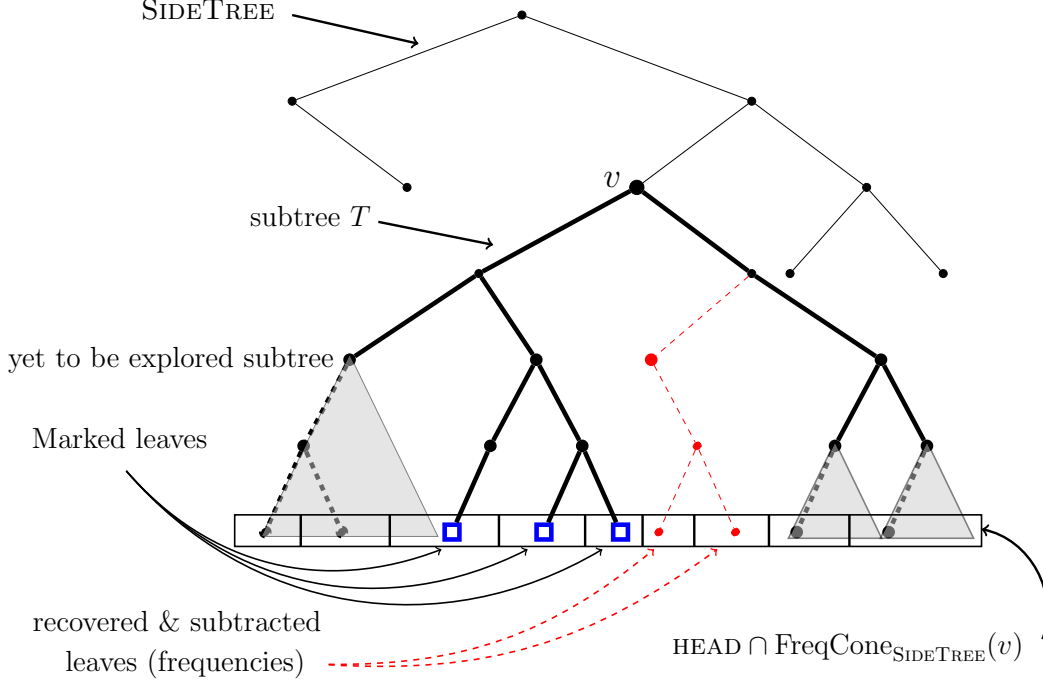
Figure 4: Illustration of an instance of RobustPromiseSFT (Algorithm 6). This procedure takes in a tree SideTree (shown with thin edges) together with a leaf $v \in$ Leaves(SideTree) and adaptively explores/constructs the subtree T rooted at v to find all heavy frequencies that lie in FreqCone$_{\text{SideTree}}(v)$. If head denotes the set of heavy frequencies, then the algorithm finds head $\cap$ FreqCone$_{\text{SideTree}}(v)$ by exploring T. Once the identity of a leaf is fully revealed, the algorithm adds that leaf to the set Marked. When the number of marked leaves grows to the point where marked frequencies can be estimated cheaply, our algorithm estimates them all in a batch, subtracts off the estimated signal, and removes all corresponding leaves from T.

the input tree SideTree is illustrated in Figure 4 with thin solid black edges. Additionally, one can see node v which is a leaf of SideTree in this figure.

Algorithm 6 recovers heavy frequencies in the subree of v, i.e., $S = $ head $\cap$ FreqCone$_{\text{SideTree}}(v)$, by iteratively exploring the subtree of SideTree rooted at v, which we denote by T, and simultaneously updating $\widehat{\chi}_v$. We show an example of subtree T at some iteration of our algorithm in Figure 4 with thick solid edges. Our algorithm, in all iterations, maintains a subtree T such that the frequency cone of each of its leaves contain at least one head element, i.e.,

$$\text{for every } u \in \text{Leaves}(T) : \text{FreqCone}_{\text{SideTree} \cup T}(u) \cap \text{head} \neq \varnothing. \tag{8}$$

We demonstrate, in Figure 4, the leaves that correspond to set $S = $ head $\cap$ FreqCone$_{\text{SideTree}}(v)$ via leaves at bottom level of the subtree rooted at v. One can easily verify (8) in this figure by noting that the frequency cone of each leaf of T contains at least one element from the set head. Additionally, at every iteration of the algorithm, the union of all frequency cones of subtree T captures all heavy frequencies that are not recovered yet, i.e.,

$$S \setminus \text{supp}(\widehat{\chi}_v) \subseteq \text{supp}(\text{SideTree} \cup T). \tag{9}$$

In Figure 4, we show the set of fully recovered leaves (frequencies), i.e., supp($\widehat{\chi}_v$), using red thin dashed subtrees. These frequencies are subtracted from the residual signal $\widehat{y} - \widehat{\chi}_v$ and their corresponding leaves are removed from subtree T, as well. One can verify that condition 9 holds in the

example depicted in Figure 4. Moreover, the estimated value of every frequency that is recovered so far, is accurate up to an average error of $\frac{\mu}{\sqrt{20b}}$. More precisely, in every iteration of the algorithm the following property is maintained,

$$\frac{\sum_{\boldsymbol{f} \in \text{supp}(\widehat{\chi}_v)} |\widehat{y}(\boldsymbol{f}) - \widehat{\chi}_v(\boldsymbol{f})|^2}{|\text{supp}(\widehat{\chi}_v)|} \leq \frac{\mu^2}{20b}. \tag{10}$$

At the start of the procedure, subtree T is initialized to be the leaf v, i.e., $T = \{v\}$. Moreover, we initialize $\widehat{\chi}_v \equiv 0$. Trivially, these initial values satisfy (8), (9), and (10). The algorithm also keeps a subset of leaves denoted by Marked that contains the leaves of T that are fully identified, that is the set of leaves that are at the bottom level and hence there is no ambiguity in their frequency content. Initially Marked is empty. We show the set of marked leaves in Figure 4 using blue squares. The algorithm operates by picking the *unmarked* leaf of T that has the smallest weight. Then the algorithm explores the children of this node by running HEAVYTEST on them to detect if any heavy frequencies lie in their frequency cone. If a child passes the HEAVYTEST the algorithm updates tree T by adding that child to T. As soon as a leaf of T gets to the bottom level and becomes a leaf of T_N^{full}, the algorithm marks it, i.e., adds that leaf to the Marked set. It can be seen in Figure 4 that all marked leaves are at the bottom level of the tree. The marked leaves need not be explored any further because they are at the bottom level and their frequency content is fully identified. These operations ensure that the invariants (8), (9), and (10) are maintained.

Once the size of set Marked grows sufficiently, the algorithm estimates the values of the marked frequencies. More precisely, at some point, the size of Marked will be comparable to the maximum weight of the leaves it contains, and when this happens, the values of all marked frequencies can be estimated cheaply. Hence, when Marked is a cheap to estimate set of leaves, our algorithm esimates those frequencies in a batch up to an average error of $\frac{\mu}{20b}$, updates $\widehat{\chi}_v$ accordingly and removes all estimated (Marked) leaves from T. This ensures that invariants (8), (9), and (10) are maintained. The estimated leaves are illustrated in Figure 4 using red thin dashed subtrees. We also demontrate the subtrees of T that contain HEAD element and are yet to be explored by our algorithm using gray cones and dashed edges in Figure 4. The gray cone means that there are heavy elements in that frequency cone that need to be identified as that node has not reached the bottom level yet.

Finally, the algorithm keeps tabs on the runtime it spends and ensures that even if the input signal does not satisfy the preconditions for successful recovery, in particular if $|\text{HEAD} \cap \text{FreqCone}_{\text{SIDETREE}}(v)| > b$, the runtime stays bounded. Additionally, the algorithm performs a quality control by running a HEAVYTEST on the residual and if the recovered signal is not correct due to violation of some preconditions, it reflects this in its output.

Overview of Algorithm 7: Consider an invocation of ROBUSTSPARSEFT(x, k, ϵ, μ). Suppose that $\widehat{x}$ is a signal in the k-high SNR regime, i.e., $\widehat{x}$ has k heavy frequencies and the value of each such heavy frequency is at least 3 times higher than the tail's norm. More formally, let HEAD $\subseteq [n]^d$ denote the set of heavy (head) frequencies of $\widehat{x}$ and suppose that $|\text{HEAD}| \leq k$, and the tail norm of $\widehat{x}$ satisfies $\|\widehat{x} - \widehat{x}_{\text{HEAD}}\|_2 \leq \mu$ and additionally suppose that $|\widehat{x}(\boldsymbol{f})| \geq 3\mu$ for every $\boldsymbol{f} \in$ HEAD. The primitive ROBUSTSPARSEFT finds a signal $\widehat{\chi}$ such that $\|\widehat{x} - \widehat{\chi}\|_2^2 \leq (1 + \epsilon)\mu^2$.

Algorithm 7 recovers heavy frequencies of the input signal $\widehat{x}$, i.e., HEAD, by iteratively exploring the tree that captures the heavy frequencies, which we denote by FRONTIER, and simultaneously updating the proxy signal $\widehat{\chi}$. At the begining of the procedure, tree FRONTIER only consists of a root and will be dynamically changing throughout the execution of our algorithm. Moreover, $\widehat{\chi}$ is initially zero. The algorithm also maintains a subset of leaves denoted by Marked that contains the leaves of FRONTIER that are fully identified, that is the set of leaves that are at the bottom

---

Algorithm 6 The Inner Loop of Sparse FFT Algorithm

1: **procedure** ROBUSTPROMISESFT$(x, \widehat{\chi}_{in}, \text{SIDETREE}, v, b, k, \mu)$

2: $\triangleright$ μ: upper bound on tail norm $\|\eta\|_2$

3:     $\widehat{\chi}_{out} \leftarrow \{0\}^{n^d}$ $\triangleright$ Sparse vector to approximate $(\widehat{x} - \widehat{\chi}_{in})_{\text{FreqCone}_{\text{SIDETREE}}(v)}$

4:     Marked $\leftarrow \varnothing$ $\triangleright$ Set of marked nodes to be estimated later

5:     Let T denote the subtree of SIDETREE rooted at v – i.e., $T \leftarrow \{v\}$

6:     **repeat**

7:         **if** $|\text{LEAVES}(T)| + \|\widehat{\chi}_v\|_0 > b$ **then**

8:             **return** $\left(\text{False}, \{0\}^{n^d}\right)$ $\triangleright$ Exit because budget of v is wrong

9:         **if** Marked $\neq \varnothing$ and $\frac{|\text{Marked}|}{\max_{u \in \text{Marked}} 2^{w_T(u)}} \geq \frac{1}{4 + 2\log b}$ **then**

10:                             $\triangleright$ The set of marked frequencies that are cheap to estimate on average

11:             $\left\{\widehat{H}_u\right\}_{u \in \text{Marked}} \leftarrow \text{ESTIMATE}\left(x, \widehat{\chi}_{in} + \widehat{\chi}_{out}, \text{SIDETREE} \cup T, \text{Marked}, \frac{368b}{|\text{Marked}|}\right)$

12:             **for** $u \in$ Marked **do**

13:                 $\widehat{\chi}_{out}(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$

14:                 Remove node u from T

15:             Marked $\leftarrow \varnothing$

16:             **continue**

17:         $z \leftarrow \text{argmin}_{u \in \text{LEAVES}(T) \setminus \text{Marked}} w_T(u)$ $\triangleright$ Find the minimum weight unmarked leaf in T

18:         **if** $z \in \text{LEAVES}(T_N^{\text{full}})$ **then** $\triangleright$ Frequency $\boldsymbol{f}_z$ and leaf z are fully identified

19:             Marked $\leftarrow$ Marked $\cup \{z\}$

20:         **else**

21:             $z_{\text{left}} :=$ left child of z and $z_{\text{right}} :=$ right child of z

22:             $T' \leftarrow T \cup \{z_{\text{left}}, z_{\text{right}}\}$ $\triangleright$ Explore children of z

23:             $\text{Heavy}_\ell \leftarrow \text{HEAVYTEST}\left(x, \widehat{\chi}_{in} + \widehat{\chi}_v, \text{SIDETREE} \cup T', z_{\text{left}}, O(b\log^3 N), 6\mu^2\right)$

24:             $\text{Heavy}_r \leftarrow \text{HEAVYTEST}\left(x, \widehat{\chi}_{in} + \widehat{\chi}_v, \text{SIDETREE} \cup T', z_{\text{right}}, O(b\log^3 N), 6\mu^2\right)$

25:             **if** Heavy_ℓ **then**

26:                 Add z_{left} as the left child of z to tree T

27:             **if** Heavy_r **then**

28:                 Add z_{right} as the right child of z to tree T

29:             **if** $z \neq v$ and both Heavy_ℓ and Heavy_r are False **then**

30:                 **return** $\left(\text{False}, \{0\}^{n^d}\right)$ $\triangleright$ Exit because budget of v is wrong

31:     **until** T has no leaves besides v

32:     **if** $\text{HEAVYTEST}\left(x, \widehat{\chi}_{in} + \widehat{\chi}_v, \text{SIDETREE}, v, O(k\log^3 N), 6\mu^2\right)$ **then**

33:                  $\triangleright$ The number of heavy coordinates in $\text{FreqCone}_{\text{SIDETREE}}(v)$ is more than b

34:         **return** $\left(\text{False}, \{0\}^{n^d}\right)$

35:     **else**

36:         **return** $(\text{True}, \widehat{\chi}_{out})$

---

level and hence there is no ambiguity in their frequency content (there is exactly one element in frequency cone of marked leaves). Tree FRONTIER, in all iterations of our algorithm, maintains the invariant that the frequency cone of each of its leaves contain at least one head element and furthermore the frequency cone of each of its *unmarked* leaves contain at least $b + 1$ head element,

where $b = k^{1/3}$, i.e.,

$$| \text{FreqCone}_{\text{FRONTIER}}(v) \cap \text{HEAD}| \geq \begin{cases} 1 & \text{for every } v \in \text{Marked} \\ b+1 & \text{for every } v \in \text{LEAVES(FRONTIER)} \setminus \text{Marked} \end{cases}. \tag{11}$$

Additionally, at every iteration of the algorithm, the union of all frequency cones of tree FRONTIER captures all heavy frequencies that are not recovered yet, i.e.,

$$\text{HEAD} \setminus \text{supp}(\widehat{\chi}) \subseteq \text{supp}(\text{FRONTIER}). \tag{12}$$

The set of fully recovered leaves (frequencies), i.e., $\text{supp}(\widehat{\chi}_v)$, are subtracted from the residual signal $\widehat{x} - \widehat{\chi}$ by our algorithm and their corresponding leaves get removed from FRONTIER, as well. Moreover, the estimated value of every frequency that is recovered so far, is accurate up to an average error of $\sqrt{\frac{\epsilon}{k}} \cdot \mu$. More precisely, in every iteration of the algorithm the following property is maintained,

$$\frac{\sum_{\boldsymbol{f} \in \text{supp}(\widehat{\chi})} |\widehat{x}(\boldsymbol{f}) - \widehat{\chi}(\boldsymbol{f})|^2}{|\text{supp}(\widehat{\chi})|} \leq \frac{\epsilon}{k} \cdot \mu^2. \tag{13}$$

At the start of the procedure, FRONTIER is initialized to only contain a root, i.e., FRONTIER = {root}. Moreover, we initialize $\widehat{\chi} \equiv 0$. Trivially, these initial values satisfy (11), (12), and (13). Also the set of fully identified leaves Marked is initially empty. The algorithm explores FRONTIER by picking the *unmarked* leaf that has the smallest weight, let us call it v. Then the algorithm explores the children of this node by running ROBUSTPROMISESFT on them to recover the heavy frequencies that lie in their frequency cone. We denote by v_{left} and v_{right} the left and right children of v. Let us consider exploration of the left child v_{left}, the right child is exactly the same. If the number of heavy frequencies in the frequency cone of v_{left} is bounded by $b = k^{1/3}$, i.e., $|\text{HEAD} \cap \text{FreqCone}_{\text{FRONTIER} \cup \{v_{\text{left}}, v_{\text{right}}\}}(v_{\text{left}})| \leq b$, then ROBUSTPROMISESFT recovers every frequency in the set $\text{HEAD} \cap \text{FreqCone}_{\text{FRONTIER} \cup \{v_{\text{left}}, v_{\text{right}}\}}(v_{\text{left}})$ up to average error $\frac{\mu}{\sqrt{20b}}$. Note that this everage estimation error is not sufficient for achieving the invariant (13), hence, instead of directly using the values that ROBUSTPROMISESFT recovered and update $\widehat{\chi}$ at the newly recovered heavy frequencies, our algorithm adds the leaves corresponding to the recovered set of frequencies, i.e., $\text{HEAD} \cap \text{FreqCone}_{\text{FRONTIER} \cup \{v_{\text{left}}, v_{\text{right}}\}}(v_{\text{left}})$, at the bottom level of FRONTIER and marks them as fully identified (adds them to Marked). For achieving maximum efficinecy we employ a new *lazy estimation* scheme, that is, the estimation of values of marked leaves is delayed until there is a large number of marked leaves and thus there exists a subset of them that is cheap to estimate. On the other hand, if the number of head elements in frequency cone of v_{left} is more than b then ROBUST-PROMISESFT detects this and notifies our algorithms about it and our algorithm adds node v_{left} to FRONTIER. These operations ensure that the invariants (11), (12), and (13) are maintained.

Once the size of set Marked grows sufficiently such that it contains a subset that is cheap to estimate, our algorithm estimates the values of the cheap frequencies. More precisely, at some point, Marked will contains a non-empty subset Cheap such that the values of all frequencies in Cheap can be estimated cheaply and subsequently, our algorithm esimates those frequencies in a batch up to an average error of $\sqrt{\frac{\epsilon}{k}} \cdot \mu$, updates $\widehat{\chi}$ accordingly and removes all estimated (Cheap) leaves from FRONTIER and Marked. This ensures that invariants (11), (12), and (13) are maintained.

Analysis of RobustPromiseSFT. First we analyze the runtime and sample complexity of primitive ROBUSTPROMISESFT in the following lemma.

Lemma 19 (ROBUSTPROMISESFT – Time and Sample Complexity). *Consider an invocation of* ROBUSTPROMISESFT $(x, \widehat{\chi}_{in}, \text{SIDETREE}, v, b, \mu)$, *where* SIDETREE *is a subtree of* T_N^{full}, v *is some leaf of T, k and b are integers with $k > b$, $\mu \geq 0$, and $x, \widehat{\chi}_{in} : [n]^d \to \mathbb{C}$. Then*

Algorithm 7 Robust High-dimensional Sparse FFT Algorithm

1: **procedure** RobustSparseFT(x, k, ϵ, μ) ▷ μ is an upper bound on tail norm $\|\eta\|_2$

2:     Frontier $\leftarrow \{\text{root}\}$, $\boldsymbol{f}_{\text{root}} \leftarrow 0$

3:     $b \leftarrow \lceil k^{1/3} \rceil$

4:     $\widehat{\chi} \leftarrow \{0\}^{n^d}$

5:     Marked $\leftarrow \varnothing$ ▷ Set of fully identified leaves (frequencies)

6:     **repeat**

7:        **if** $\sum_{u \in \text{Marked}} 2^{-w_{\text{Frontier}}(u)} \geq \frac{1}{2}$ **then**

8:           Cheap $\leftarrow$ ExtractCheapSubset (Frontier, Marked)

9:              ▷ Lazy estimation: We extract from the batch of marked leaves a subset that is cheap to estimate on average

10:           $\left\{\widehat{H}_u\right\}_{u \in \text{Cheap}} \leftarrow$ Estimate $\left(x, \widehat{\chi}, \text{Frontier}, \text{Cheap}, \frac{32k}{\epsilon \cdot |\text{Cheap}|}\right)$

11:           **for** $u \in$ Cheap **do**

12:              $\widehat{\chi}(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$

13:              Remove node u from tree Frontier

14:           Marked $\leftarrow$ Marked $\setminus$ Cheap

15:           **continue**

16:        $v \leftarrow \text{argmin}_{u \in \text{Leaves}(\text{Frontier}) \setminus \text{Marked}} w_{\text{Frontier}}(u)$

17:              ▷ pick the minimum weight leaf in Frontier which is not in Marked

18:        $v_{\text{left}} \leftarrow$ left child of v and $v_{\text{right}} \leftarrow$ right child of v

19:        $T \leftarrow$ Frontier $\cup \{v_{\text{left}}, v_{\text{right}}\}$

20:        $(\text{IsCorr}_{\text{left}}, \widehat{\chi}_{\text{left}}) \leftarrow$ RobustPromiseSFT $(x, \widehat{\chi}, T, v_{\text{left}}, b, k, \mu)$

21:        $(\text{IsCorr}_{\text{right}}, \widehat{\chi}_{\text{right}}) \leftarrow$ RobustPromiseSFT $(x, \widehat{\chi}, T, v_{\text{right}}, b, k, \mu)$

22:        **if** IsCorr$_{\text{left}}$ **then**

23:           $\forall \boldsymbol{f} \in \text{supp}(\widehat{\chi}_{\text{left}})$, add the unique leaf corresponding to $\boldsymbol{f}$ to Frontier and Marked

24:        **else**

25:           Add v_{left} to Frontier

26:        **if** IsCorr$_{\text{right}}$ **then**

27:           $\forall \boldsymbol{f} \in \text{supp}(\widehat{\chi}_{\text{right}})$, add the unique leaf corresponding to $\boldsymbol{f}$ to Frontier and Marked

28:        **else**

29:           Add v_{right} to Frontier

30:        **if** IsCorr$_{\text{left}}$ and IsCorr$_{\text{right}}$ **then**

31:           Remove v from Frontier

32:     **until** Frontier has no leaves besides root

33:     **return** $\widehat{\chi}$

34: **procedure** ExtractCheapSubset(T, S)

35:     $L \leftarrow \varnothing$

36:     **while** $|L| \cdot (8 + 4 \log |S|) < \max_{v \in L} 2^{w_T(v)}$ **do**

37:        $L \leftarrow L \cup \left\{\text{argmin}_{u \in S \setminus L} w_T(u)\right\}$

38:     Return L

- *The running time of primitive is bounded by*

$$\widetilde{O}\left(\|\widehat{\chi}_{in}\|_0 \cdot \left(b^2 + k\right) + bk + 2^{w_{\text{SideTree}}(v)} \cdot \left(b^3 + k\right)\right).$$

48

- *The number of accesses it makes on x is always bounded by*

$$\widetilde{O}\left(2^{w_{\text{SIDETREE}}(v)} \cdot \left(b^3 + k\right)\right).$$

Furthermore, the output signal $\widehat{\chi}_v$ always satisfies $\|\widehat{\chi}_v\|_0 \leq b$ and $\text{supp}(\widehat{\chi}_v) \subseteq \text{FreqCone}_{\text{SIDETREE}}(v)$.

Proof. First we prove that Algorithm 6 terminates after a bounded number of iterations. In order to bound the number of iterations of ROBUSTPROMISESFT, we use a potential function argument. Let $\widehat{\chi}_v^{(t)}$ denote the signal $\widehat{\chi}_v$ at the end of iteration t of the algorithm. Furthermore, let $T^{(t)}$ denote the subtree T at the end of tth iteration. Additionally, let $\text{Marked}^{(t)}$ and $\text{Identified}^{(t)}$ denote the set Marked (defined in Algorithm 6) at the end of iteration t.

We prove that the algorithm always terminates after $O(b \cdot \log N)$ iterations. We prove this by contradiction. For any integer t, define the following potential function

$$\phi_t := \left|\text{Marked}^{(t)}\right| + 2\log N \cdot \left\|\widehat{\chi}_v^{(t)}\right\|_0 + \sum_{u \in \text{LEAVES}\left(T^{(t)}\right)} l_{T^{(t)}}(u).$$

Towards contradiction, suppose that Algorithm 6 does not terminate after $4b \log N$ iterations. We show that the above potential function increases by at least 1 at every iteration $2 \leq t \leq 4b \log N$, i.e., $\phi_t \geq \phi_{t-1} + 1$. This is enough to conclude the termination of the algorithm because the if-statement in line 7 ensures that $\left|\text{Marked}^{(t)}\right| \leq \left|\text{LEAVES}\left(T^{(t)}\right)\right| \leq b$ and also $\left\|\widehat{\chi}_v^{(t)}\right\|_0 \leq b$, thus, $\phi_t = O(b \log N)$ for any t, which proves that algorithm terminates after $O(b \log N)$ iterations.

At any given iteration t of the algorithm, there are 3 possibilities that can happen. We show that if any of these possibilities happen, then the potential function ϕ_t increases by at least 1.

Case 1 – the if-statement in line 9 of Algorithm 6 is True. In this case, the algorithm constructs $T^{(t)}$ by removing all leaves that are in the set $\text{Marked}^{(t-1)}$ from tree $T^{(t-1)}$ and leaving the rest of the tree unchanged. Furthermore, the algorithm sets $\text{Marked}^{(t)} \leftarrow \varnothing$. By construction, the level of the leaves that are in $\text{Marked}^{(t-1)}$ is at most $\log N$, thus

$$\sum_{u \in \text{LEAVES}\left(T^{(t)}\right)} l_{T^{(t)}}(u) \geq \sum_{u \in \text{LEAVES}\left(T^{(t-1)}\right)} l_{T^{(t-1)}}(u) - \log N \cdot \left|\text{Marked}^{(t-1)}\right|$$

Additionally, in this case, the algorithm computes $\{\widehat{H}_u\}_{u \in \text{Marked}^{(t-1)}}$ by running the procedure ESTIMATE in line 11 and then updates $\widehat{\chi}_v^{(t)}(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$ for every $u \in \text{Marked}^{(t-1)}$ and $\widehat{\chi}_v^{(t)}(\boldsymbol{\xi}) = \widehat{\chi}_v^{(t-1)}(\boldsymbol{\xi})$ at every other frequency $\boldsymbol{\xi}$. Therefore, $\left\|\widehat{\chi}_v^{(t)}\right\|_0 = \left\|\widehat{\chi}_v^{(t)}\right\|_0 + \left|\text{Marked}^{(t-1)}\right|$. Also, $\left|\text{Marked}^{(t)}\right| = 0$. Hence,

$$\phi_t - \phi_{t-1} \geq (\log N - 1) \cdot \left|\text{Marked}^{(t-1)}\right| \geq 1,$$

where the inequality above holds because the if-statement in line 9 of the algorithm is True, ensuring that $\text{Marked}^{(t-1)} \neq \varnothing$.

Case 2 – the if-statement in line 9 is False and if-statement in line 18 is True. In this case, in line 19, the algorithm updates Marked by adding the leaf z to this set, i.e., $\text{Marked}^{(t)} \leftarrow \text{Marked}^{(t-1)} \cup \{z\}$. Additionally, tree T and signal $\widehat{\chi}_v$ stay unchanged, i.e., $\widehat{\chi}_v^{(t)} = \widehat{\chi}_v^{(t-1)}$ and $T^{(t)} = T^{(t-1)}$. Therefore, in this case, $\phi_{t+1} - \phi_t = 1$.

Case 3 – both if-statements in lines 9 and 18 are False.   In this case, either the algorithm terminates by the if-statement in line 29, which is exactly what we have assumed towards a contradiction that did not happen, or $\sum_{u \in \text{LEAVES}(T^{(t)})} l_{T^{(t)}}(u) \geq \sum_{u \in \text{LEAVES}(T^{(t-1)})} l_{T^{(t-1)}}(u) + 1$, while $\left|\text{Marked}^{(t)}\right| = \left|\text{Marked}^{(t-1)}\right|$ and $\left\|\widehat{\chi}_v^{(t)}\right\|_0 = \left\|\widehat{\chi}_v^{(t-1)}\right\|_0$ (since we assumed $t \geq 2$ and hence $z \neq v$). Thus, $\phi_{t+1} - \phi_t \geq 1$.

So far we have showed that at every iteration, under the **cases 1**, **2**, and **3**, the potential function ϕ_t increases by at least one. Now we show that, at every iteration, exactly one of these three cases happens and hence the algorithm never stalls. For the sake of contradiction suppose that at iteration t, the algorithm stalls. For this to happen, we must have that all leaves of $T^{(t-1)}$ are in the set $\text{Marked}^{(t-1)}$. By the if-statement in line 7 of Algorithm 6, we are guaranteed that $|\text{Marked}^{(t-1)}| \leq b$. Therefore, by Lemma 11, there must exist a subset $\varnothing \neq L \subset \text{Marked}^{(t-1)}$ such that $|L| \geq \frac{1}{4 + 2\log b} \cdot \max_{u \in L} 2^{w_{T^{(t-1)}}(u)}$. Hence, it follows from the way our algorithm explores the nodes of the tree in an increasing order of weights, that there must exist some $t' < t$ such that $\varnothing \neq \text{Marked}^{(t'-1)} \subseteq \text{Marked}^{(t-1)}$ such that the if-statement in line 9 becomes True on $\text{Marked}^{(t'-1)}$. Therefore, **case 1** must have happened at iteration t', resulting in emptying the set of identified frequencies, i.e., $\text{Marked}^{(t')} \leftarrow \varnothing$. This would have resulted in $\text{Marked}^{(t'-1)} \nsubseteq \text{Marked}^{(t-1)}$ which is the contradiction we wanted. Therefore the algorithm never stalls and always exactly one of **case 1**, **2**, and **3** happen.

We proved that ϕ_t must increase by at least 1 at every iteration. Since $\phi_1 \geq 0$ and we assumed that the algorithm did not terminate after $q = 4b \log N$ iterations, this potential will have a value of at least $4b \log N - 1$:

$$\phi_q \geq 4b \log N - 1, \text{ where } q = 4b \log N.$$

On the other hand, since the if-statement in line 7 ensures that the number of leaves of $T^{(t)}$ is always bounded by $b - \left\|\widehat{\chi}_v^{(t)}\right\|_0$, the sum $\sum_{u \in \text{LEAVES}(T^{(t)})} l_{T^{(t)}}(u)$ is always bounded by $\left(b - \left\|\widehat{\chi}_v^{(t)}\right\|_0\right) \cdot \log N$. Also, the size of the set $\text{Marked}^{(t)}$, which is a subset of $\text{LEAVES}(T^{(t)})$, is always bounded by $b - \left\|\widehat{\chi}_v^{(t)}\right\|_0$. This means that we must have $\phi_q \leq b \cdot (\log N + 1) + (\log N - 1) \cdot \left\|\widehat{\chi}_v^{(q)}\right\|_0$. The if-statement in line 7 also ensures that $\left\|\widehat{\chi}_v^{(q)}\right\|_0 \leq b$ which implies that $\phi_q \leq 2b \cdot \log N$ which contradicts $\phi_q \geq 4b \cdot \log N - 1$. This proves that the number of iterations of the algorithm must be bounded by $O(b \cdot \log N)$, guaranteeing termination of RobustSparseFT. The termination quarantee along with the way our algorithm constructs $\widehat{\chi}_v$ and the if-staement in line 7, imply that the output signal $\widehat{\chi}_v$ always satisfies $\|\widehat{\chi}_v\|_0 \leq b$ and $\text{supp}(\widehat{\chi}_v) \subseteq \text{FreqCone}_{\text{SIDETREE}}(v)$. Now we bound the running time and sample complexity of the algorithm.

Sample Complexity and Runtime: First recall that we proved $\left\|\widehat{\chi}_v^{(t)}\right\|_0 \leq b$ for every iteration t. Additionally, the weight of the node z at every iteration of the algorithm is bounded by $w_{T^{(t)}}(z) \leq \log(2b)$. To see this, note that if at some iteration t, the set of identified frequencies (or leaves) that our algorithm keeps, $\text{Marked}^{(t)}$, is such that there exists a leaf $u \in \text{Marked}^{(t)}$ with $w_{T^{(t)}}(u) > \log(2b)$, then by Lemma 11, $\text{Marked}^{(t)}$ contains a non-empty subset that is cheap to estimate. Thus, at some iteration $t' < t$, where $\varnothing \neq \text{Marked}^{(t')} \subset \text{Marked}^{(t)}$ holds, it must have been the case that the if-statement in line 9 became True on $\text{Marked}^{(t')}$. If this happened, our algorithm would have estimated $\text{Marked}^{(t')}$ at iteration t' and so we would have $\text{Marked}^{(t')} \cap \text{Marked}^{(t)} = \varnothing$ which is a contradiction.

Given the above inequalities, by Lemma 17, time and sample complexities of every invocation of HeavyTest in lines 23 and 24 of Algorithm 6 are bounded by $\widetilde{O}\left(\|\widehat{\chi}_{in}\|_0 \cdot b + 2^{w_{\text{SIDETREE}}(v)} \cdot b^2\right)$

and $\widetilde{O}\left(2^{w_{\text{SIDETREE}}(v)} \cdot b^2\right)$, respectively. Also, since $\|\widehat{\chi}_v\|_0 \leq b$, the runtime and sample complexity of the HEAVYTEST in line 32 of the algorithm are bounded by $\widetilde{O}\left(\|\widehat{\chi}_{in}\|_0 \cdot k + bk + 2^{w_{\text{SIDETREE}}(v)} \cdot k\right)$ and $\widetilde{O}\left(2^{w_{\text{SIDETREE}}(v)} \cdot k\right)$, respectively. Thus, total sample and time complexity of all invocations of HEAVYTEST throughout the execution of our algorithm are bounded by $\widetilde{O}\left(2^{w_{\text{SIDETREE}}(v)} \cdot (b^3 + k)\right)$ and $\widetilde{O}\left(\|\widehat{\chi}_{in}\|_0 \cdot (b^2 + k) + bk + 2^{w_{\text{SIDETREE}}(v)} \cdot (b^3 + k)\right)$, respectively

Additionally, by Lemma 18, the sample and time complexity of every invocation of ESTI-MATE in line 11 of our algorithm are bounded by $\widetilde{O}\left(\frac{b \cdot 2^{w_{\text{SIDETREE}}(v)}}{\left|\text{Marked}^{(t-1)}\right|} \cdot \sum_{u \in \text{Marked}^{(t-1)}} 2^{w_{T^{(t-1)}}(u)}\right)$ and

$\widetilde{O}\left(\frac{b \cdot 2^{w_{\text{SIDETREE}}(v)}}{\left|\text{Marked}^{(t-1)}\right|} \cdot \sum_{u \in \text{Marked}^{(t-1)}} 2^{w_{T^{(t-1)}}(u)} + b \cdot \|\widehat{\chi}\|_0\right)$, respectively. Because we run ESTIMATE only when the if-statement in line 9 holds true, the runtime and sample complexity of ESTIMATE can be further upper bounded by $\widetilde{O}\left(\left|\text{Marked}^{(t-1)}\right| \cdot b \cdot 2^{w_{\text{SIDETREE}}(v)} + b \cdot \|\widehat{\chi}\|_0\right)$ and $\widetilde{O}\left(\left|\text{Marked}^{(t-1)}\right| \cdot b \cdot 2^{w_{\text{SIDETREE}}(v)}\right)$, respectively. Using the fact that

$$\sum_{t: \text{ if-statement in line 9 is True}} \left|\text{Marked}^{(t-1)}\right| = \|\widehat{\chi}_v\|_0 \leq b,$$

the total runtime and sample complexity of all invocations of ESTIMATE in all iterations can be upper bounded by $\widetilde{O}\left(2^{w_{\text{SIDETREE}}(v)} \cdot b^2 + b^2 \cdot \|\widehat{\chi}\|_0\right)$ and $\widetilde{O}\left(2^{w_{\text{SIDETREE}}(v)} \cdot b^2\right)$, respectively. Therefore, by adding up the above contributions we can upper bound the total runtime and sample complexity by $\widetilde{O}\left(\|\widehat{\chi}_{in}\|_0 \cdot (b^2 + k) + bk + 2^{w_{\text{SIDETREE}}(v)} \cdot (b^3 + k)\right)$ and $\widetilde{O}\left(2^{w_{\text{SIDETREE}}(v)} \cdot (b^3 + k)\right)$ which completes the proof of the lemma.

$\square$

We are now in a position to present the main invariant of primitive ROBUSTPROMISESFT.

Lemma 20 (ROBUSTPROMISESFT - Invariants). *Consider the preconditions of Lemma 19. Let $\widehat{y} := \widehat{x} - \widehat{\chi}_{in}$ and $S := \text{FreqCone}_{\text{SIDETREE}}(v) \cap \text{HEAD}_\mu(\widehat{y})$, where $\text{HEAD}_\mu(\cdot)$ is defined as per (7). If i) $\text{HEAD}_\mu(\widehat{y}) \subseteq \text{supp}(\text{SIDETREE})$, ii) $\|\widehat{y} - \widehat{y}_{\text{HEAD}_\mu(\widehat{y})}\|_2^2 \leq \frac{11\mu^2}{10}$, and iii) $|S| \leq k$, then with probability at least $1 - \frac{1}{N^4}$, the output $(\text{Budget}, \widehat{\chi}_v)$ of Algorithm 6 satisfies the following,*

1. *If $|S| \leq b$ then $\text{Budget} = \text{True}$, $\text{supp}(\widehat{\chi}_v) \subseteq S$, and $\|\widehat{y}_S - \widehat{\chi}_v\|_2^2 \leq \frac{\mu^2}{20}$;*

2. *If $|S| > b$ then $\text{Budget} = \text{False}$ and $\widehat{\chi}_v \equiv \{0\}^{n^d}$.*

Proof. We first analyze the algorithm under the assumption that the primitives HEAVYTEST and ESTIMATE are replaced with more powerful primitives that succeeds deterministically. Hence, we assume that HEAVYTEST correctly tests the "heavy" hypothesis on its input signal with probability 1 and also ESTIMATE achieves the estimation guarantee of Lemma 18 deterministrically. With these assumptions in place, we prove that the lemma holds deterministically (with probability 1). We then establish a coupling between this idealized execution and the actual execution of our algorithm, leading to our result.

We prove the first statement of lemma by induction on the *Repeat-Until loop* of the algorithm. Let $\widehat{\chi}_v^{(t)}$ denote the signal $\widehat{\chi}_v$ at the end of iteration t of the algorithm. Furthermore, let $T^{(t)}$ denote the subtree T at the end of tth iteration. Additionally, let $\text{Marked}^{(t)}$ denote the set Marked (defined in Algorithm 6) at the end of iteration t. We prove that if the precondition of statement 1 (that is $|S| \leq b$) together with i, ii and iii hold, then at every iteration $t = 0, 1, 2, \ldots$ of Algorithm 6, the following properties are maintained,

$P_1(t)$ $S \setminus \mathrm{supp}\left(\widehat{\chi}_v^{(t)}\right) \subseteq \mathrm{supp}\left(T^{(t)}\right) := \bigcup_{u \in \mathrm{LEAVES}\left(T^{(t)}\right)} \mathrm{FreqCone}_{\mathrm{SIDETREE} \cup T^{(t)}}(u);$

$P_2(t)$ For every leaf $u \neq v$ of subtree $T^{(t)}$, $\mathrm{HEAD}_\mu(\widehat{y}) \cap \mathrm{FreqCone}_{\mathrm{SIDETREE} \cup T^{(t)}}(u) \neq \varnothing;$

$P_3(t)$ $\left\| \widehat{y}_{S^{(t)}} - \widehat{\chi}_v^{(t)} \right\|_2^2 \leq \frac{|S^{(t)}|}{20b} \cdot \mu^2$, where $S^{(t)} := \mathrm{supp}\left(\widehat{\chi}_v^{(t)}\right);$

$P_4(t)$ $S^{(t)} \subseteq S$ and $S^{(t)} \cap \left(\bigcup_{\substack{u \in \mathrm{LEAVES}\left(T^{(t)}\right) \\ u \neq v}} \mathrm{FreqCone}_{\mathrm{SIDETREE} \cup T^{(t)}}(u) \right) = \varnothing;$

The **base of induction** corresponds to the zeroth iteration $(t = 0)$, at which point $T^{(0)} = \{v\}$ is a subtree of SIDETREE that solely consists of node v. Moreover, $\widehat{\chi}_v^{(0)} \equiv 0$. Thus, statement $P_1(0)$ trivially holds by definition of set S. The statement $P_2(0)$ holds since there exists no leaf $u \neq v$ in $T^{(0)}$. Statements $P_3(0)$ and $P_4(0)$ hold because of the fact that $\widehat{\chi}_v^{(0)} \equiv 0$.

We now prove the **inductive step** by assuming that the inductive hypothesis, $P(t-1)$ is satisfied for some iteration $t-1$ of Algorithm 6, and then proving that $P(t)$ holds. First, we remark that if inductive hypotheses $P_2(t-1)$ and $P_4(t-1)$ hold true, then by the precondition of statement 1 of the lemma (that is $|S| \leq b$) the if-statement in line 7 of Algorithm 6 is False and hence lines 7 and 8 of the algorithm can be ignored in our analysis. We proceed to prove the induction by considering the three cases that can happen in iteration t:

Case 1 – the if-statement in line 9 of Algorithm 6 is True. In this case, the algorithm computes $\{\widehat{H}_u\}_{u \in \mathrm{Marked}^{(t-1)}}$ by running the procedure ESTIMATE in line 11 and then updates $\widehat{\chi}_v^{(t)}(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$ for every $u \in \mathrm{Marked}^{(t-1)}$ and $\widehat{\chi}_v^{(t)}(\boldsymbol{\xi}) = \widehat{\chi}_v^{(t-1)}(\boldsymbol{\xi})$ at every other frequency $\boldsymbol{\xi}$. Therefore, if we let $L := \left\{ \boldsymbol{f}_u : u \in \mathrm{Marked}^{(t-1)} \right\}$, then $S^{(t)} \setminus S^{(t-1)} = L$, by inductive hypothesis $P_4(t-1)$. By $P_3(t-1)$ along with Lemma 18 (its deterministic version that succeeds with probability 1), we find that

$$
\begin{aligned}
\left\| (\widehat{\chi}_v^{(t)} - \widehat{y})_{S^{(t)}} \right\|_2^2 &= \left\| (\widehat{\chi}_v^{(t)} - \widehat{y})_{S^{(t-1)}} \right\|_2^2 + \left\| (\widehat{\chi}_v^{(t)} - \widehat{y})_{S^{(t)} \setminus S^{(t-1)}} \right\|_2^2 \\
&= \left\| (\widehat{\chi}_v^{(t-1)} - \widehat{y})_{S^{(t-1)}} \right\|_2^2 + \left\| (\widehat{\chi}_v^{(t)} - \widehat{y})_L \right\|_2^2 \\
&\leq \frac{|S^{(t-1)}|}{20b} \mu^2 + \frac{|L|}{23b} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}\left(\mathrm{SIDETREE} \cup T^{(t-1)}\right)} \left| \left(\widehat{y} - \widehat{\chi}_v^{(t-1)}\right)(\boldsymbol{\xi}) \right|^2 .
\end{aligned} \tag{14}
$$

Now we bound the second term above,

$$\sum_{\boldsymbol{\xi}\in[n]^d\setminus\operatorname{supp}\left(\textsc{SideTree}\cup T^{(t-1)}\right)}\left|\left(\widehat{y}-\widehat{\chi}_v^{(t-1)}\right)(\boldsymbol{\xi})\right|^2$$

$$=\sum_{\boldsymbol{\xi}\in[n]^d\setminus\operatorname{supp}(\textsc{SideTree})}|\widehat{y}(\boldsymbol{\xi})|^2+\sum_{\boldsymbol{\xi}\in\operatorname{FreqCone}_{\textsc{SideTree}}(v)\setminus\operatorname{supp}\left(T^{(t-1)}\right)}\left|\left(\widehat{y}-\widehat{\chi}_v^{(t-1)}\right)(\boldsymbol{\xi})\right|^2$$

$$=\sum_{\boldsymbol{\xi}\in[n]^d\setminus\operatorname{supp}(\textsc{SideTree})}|\widehat{y}(\boldsymbol{\xi})|^2$$

$$+\sum_{\boldsymbol{\xi}\in\operatorname{FreqCone}_{\textsc{SideTree}}(v)\setminus\left(\operatorname{supp}\left(T^{(t-1)}\right)\cup S^{(t-1)}\right)}|\widehat{y}(\boldsymbol{\xi})|^2+\left\|\widehat{y}_{S^{(t-1)}}-\widehat{\chi}_v^{(t-1)}\right\|_2^2$$

$$=\sum_{\boldsymbol{\xi}\in[n]^d\setminus\left(\operatorname{supp}\left(\textsc{SideTree}\cup T^{(t-1)}\right)\cup S^{(t-1)}\right)}|\widehat{y}(\boldsymbol{\xi})|^2+\left\|\widehat{y}_{S^{(t-1)}}-\widehat{\chi}_v^{(t-1)}\right\|_2^2$$

$$\le\sum_{\boldsymbol{\xi}\in[n]^d\setminus\textsc{Head}_\mu(\widehat{y})}|\widehat{y}(\boldsymbol{\xi})|^2+\left\|\widehat{y}_{S^{(t-1)}}-\widehat{\chi}_v^{(t-1)}\right\|_2^2 \qquad \text{(by } P_1(t-1)\text{, precondition i and definition of } S)$$

$$\le\frac{23}{20}\cdot\mu^2 \qquad \text{(by } P_3(t-1) \text{ and } P_4(t-1) \text{ and precondition } |S|\le b).$$

Therefore, by plugging the above bound back to (14) we find that,

$$\left\|\left(\widehat{\chi}_v^{(t)}-\widehat{y}\right)_{S^{(t)}}\right\|_2^2\le\frac{\left|S^{(t-1)}\right|}{20b}\cdot\mu^2+\frac{|L|}{23b}\cdot\left(\frac{23}{20}\mu^2\right)=\frac{\left|S^{(t)}\right|}{20b}\cdot\mu^2,$$

which proves the inductive claim $P_3(t)$. Moreover, $P_2(t-1)$ implies that $L\subseteq S$. Thus, the fact $S^{(t)}=S^{(t-1)}\cup L$ together with inductive hypothesis $P_4(t-1)$ as well as the construction of $T^{(t)}$ ($T^{(t)}$ is constructed by removing leaves of $\textsc{Marked}^{(t-1)}$ from tree $T^{(t-1)}$), imply $P_4(t)$. The construction of $T^{(t)}$ together with the fact that $|\operatorname{FreqCone}_{\textsc{SideTree}\cup T^{(t-1)}}(u)|=1$ for every $u\in\textsc{Marked}^{(t-1)}$ give $P_1(t)$ and $P_2(t)$.

We now consider the other two cases. Let $z\in\textsc{Leaves}\left(T^{(t-1)}\right)$ be the smallest weight leaf chosen by the algorithm in line 17.

Case 2 – the if-statement in line 9 is False and if-statement in line 18 is True. In this case, in line 19, the algorithm updates Marked by adding the leaf z to this set, i.e., $\textsc{Marked}^{(t)}\leftarrow\textsc{Marked}^{(t-1)}\cup\{z\}$. Additionally, in this case the tree T and signal $\widehat{\chi}_v$ stay unchanged, i.e., $\widehat{\chi}_v^{(t)}=\widehat{\chi}_v^{(t-1)}$ and $T^{(t)}=T^{(t-1)}$. Therefore, $P_1(t)$, $P_2(t)$, $P_3(t)$, and $P_4(t)$ all trivially hold because of the inductive hypothesis $P(t-1)$.

Case 3 – both if-statements in lines 9 and 18 are False. In this case, the algorithm constructs tree T' by adding leaves z_{right} and z_{left} to tree $T^{(t-1)}$ as right and left children of z in line 22. Then we compute Heavy_ℓ and Heavy_r in lines 23 and 24 by running the primitive $\textsc{HeavyTest}$ with inputs $\left(x,\widehat{\chi}_v^{(t-1)}+\widehat{\chi}_{in},\textsc{SideTree}\cup T',z_{\text{left}},O(b\log^3 N),6\mu^2\right)$ and $\left(x,\widehat{\chi}_v^{(t-1)}+\widehat{\chi}_{in},\textsc{SideTree}\cup T',z_{\text{right}},O(b\log^3 N)\right)$ respectively. There are two possibilities that can happen to each of Heavy_ℓ and Heavy_r. In the following we focus on analyzing Heavy_ℓ, but Heavy_r can be analyzed exactly the same way.

Possibility 1) $\operatorname{FreqCone}_{\textsc{SideTree}\cup T'}(z_{\text{left}})\cap\textsc{Head}_\mu(\widehat{y})=\varnothing$. Note that, by construction of T' we have

$$\operatorname{FreqCone}_{\textsc{SideTree}\cup T^{(t-1)}}(z)=\operatorname{FreqCone}_{\textsc{SideTree}\cup T'}(z_{\text{left}})\cup\operatorname{FreqCone}_{\textsc{SideTree}\cup T'}(z_{\text{right}}).$$

Hence, by inductive hypothesis $P_4(t-1)$ we have,

$$\sum_{\boldsymbol{\xi}\in[n]^d\setminus\text{supp}(\text{SIDETREE}\cup T')}\left|\left(\widehat{y}-\widehat{\chi}_v^{(t-1)}\right)(\boldsymbol{\xi})\right|^2$$

$$=\sum_{\boldsymbol{\xi}\in[n]^d\setminus\text{supp}(\text{SIDETREE})}|\widehat{y}(\boldsymbol{\xi})|^2$$

$$+\sum_{\boldsymbol{\xi}\in\text{FreqCone}_{\text{SIDETREE}}(v)\setminus\text{supp}(T^{(t-1)})}\left|\left(\widehat{y}-\widehat{\chi}_v^{(t-1)}\right)(\boldsymbol{\xi})\right|^2$$

$$=\sum_{\boldsymbol{\xi}\in[n]^d\setminus\text{supp}(\text{SIDETREE})}|\widehat{y}(\boldsymbol{\xi})|^2$$

$$+\sum_{\boldsymbol{\xi}\in\text{FreqCone}_{\text{SIDETREE}}(v)\setminus\left(\text{supp}(T^{(t-1)})\cup S^{(t-1)}\right)}|\widehat{y}(\boldsymbol{\xi})|^2+\left\|\widehat{y}_{S^{(t-1)}}-\widehat{\chi}_v^{(t-1)}\right\|_2^2$$

$$\leq\sum_{\boldsymbol{\xi}\in[n]^d\setminus\left(\text{supp}(\text{SIDETREE}\cup T')\cup S^{(t-1)}\right)}|\widehat{y}(\boldsymbol{\xi})|^2+\frac{\mu^2}{20},$$

where the last inequality above follows by inductive hypotheses $P_3(t-1)$ and $P_4(t-1)$ and precondition $|S|\leq b$. Therefore, if $\widehat{G}_\ell$ is a $(z_{\text{left}},\text{SIDETREE}\cup T')$-isolating filter as per the construction in Lemma 9, then by Corollary 1 along with the above inequality, we have

$$\left\|\left(\widehat{y}-\widehat{\chi}_v^{(t-1)}\right)\cdot\widehat{G}_\ell\right\|_2^2\leq\left\|\widehat{y}_{\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{left}})}\right\|_2^2+\sum_{\boldsymbol{\xi}\in[n]^d\setminus\text{supp}(\text{SIDETREE}\cup T')}\left|\left(\widehat{y}-\widehat{\chi}_v^{(t-1)}\right)(\boldsymbol{\xi})\right|^2$$

$$\leq\left\|\widehat{y}_{\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{left}})}\right\|_2^2+\sum_{\boldsymbol{\xi}\in[n]^d\setminus\left(\text{supp}(\text{SIDETREE}\cup T')\cup S^{(t-1)}\right)}|\widehat{y}(\boldsymbol{\xi})|^2+\frac{\mu^2}{20}$$

$$\leq\sum_{\boldsymbol{\xi}\in[n]^d\setminus\text{HEAD}_\mu(\widehat{y})}|\widehat{y}(\boldsymbol{\xi})|^2+\frac{\mu^2}{20}$$

$$\leq\frac{23}{20}\cdot\mu^2$$

where the third line above follows from the assumption that $\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{left}})\cap\text{HEAD}_\mu(\widehat{y})=\varnothing$, inductive hypothesis $P_1(t-1)$, precondition i of the lemma together with the definition of set S. This proves that the precondition of the second claim of Lemma 17 holds and therefore by invoking this lemma (the deterministic version of it that succeeds with probability 1), we have that Heavy_ℓ in line 23 of the algorithm is False. Using a similar argument, if $\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{right}})\cap\text{HEAD}_\mu(\widehat{y})=\varnothing$, then Heavy_r is False.

Possibility 2) Suppose that $\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{left}})\cap\text{HEAD}_\mu(\widehat{y})\neq\varnothing$. If filter $\widehat{G}_\ell$ is a $(z_{\text{left}},\text{SIDETREE}\cup T')$-isolating filter constructed in Lemma 9, then by Corollary 1 along with inductive hypothesis $P_4(t-1)$,

$$\left\|\left(\left(\widehat{y}-\widehat{\chi}_v^{(t-1)}\right)\cdot\widehat{G}_\ell\right)_{[n]^d\setminus S}\right\|_2^2=\left\|\left(\widehat{y}\cdot\widehat{G}_\ell\right)_{[n]^d\setminus S}\right\|_2^2$$

$$\leq\left\|\widehat{y}_{\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{left}})\setminus S}\right\|_2^2+\sum_{\boldsymbol{\xi}\in[n]^d\setminus(\text{supp}(\text{SIDETREE}\cup T')\cup S)}|\widehat{y}(\boldsymbol{\xi})|^2$$

$$\leq\left\|\widehat{y}-\widehat{y}_{\text{HEAD}_\mu(\widehat{y})}\right\|_2^2\leq\frac{11}{10}\cdot\mu^2.\qquad\text{(precondition ii)}$$

54

Additionally,

$$\left\|\left(\left(\widehat{y}-\widehat{\chi}_v^{(t-1)}\right)\cdot\widehat{G}_\ell\right)_S\right\|_2^2 \geq \left\|\left(\widehat{y}-\widehat{\chi}_v^{(t-1)}\right)_{\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{left}})\cap S}\right\|_2^2$$

$$= \left\|\widehat{y}_{\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{left}})\cap S}\right\|_2^2 \geq 9\mu^2,$$

which follows by the assumption $\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y}) \neq \varnothing$ along with the definition of S and $\text{HEAD}_\mu(\cdot)$. Hence, by the above inequalities and the precondition $|S| \leq b$, we can invoke Lemma 17 to conclude that Heavy_ℓ in line 23 of the algorithm is True. Using a similar argument, if $\text{FreqCone}_{\text{SIDETREE}\cup T'}(z_{\text{right}}) \cap \text{HEAD}_\mu(\widehat{y}) \neq \varnothing$ then Heavy_r is True.

Based on the above arguments, according to the values of Heavy_ℓ and Heavy_r, there are various cases that can happen. First, it cannot happen that Heavy_ℓ and Heavy_r are both False unless $z = v$, by the inductive hypothesis $P(t-1)$. If $\text{Heavy}_\ell = \text{Heavy}_r = \text{False}$ and $z = v$, the algorithm returns $\widehat{\chi}_v^{(t)} \equiv \{0\}^{n^d}$ which satisfies all properties in $P(t)$. The second case corresponds to $\text{Heavy}_\ell = \text{False}$ and $\text{Heavy}_r = \text{True}$. In this case, tree $T^{(t)}$ is obtained from $T^{(t-1)}$ by adding z_{right} as the right child of z. Therefore, by inductive hypothesis $P(t-1)$, all properties in $P(t)$ immediately hold. One can show that $P(t)$ holds in the case of $\text{Heavy}_\ell = \text{True}$ and $\text{Heavy}_r = \text{False}$ in exactly the same fashion. Finally, if both of Heavy_ℓ and Heavy_r are True, then tree $T^{(t)}$ is obtained by adding leaves z_{right} and z_{left} as right and left children of z to tree $T^{(t-1)}$. It follows straightforwardly from the inductive hypothesis $P(t-1)$ that $P(t)$ holds.

So far we have showed that under **cases 1**, **2**, and **3**, the property $P(t)$ is maintained. Recall that in the proof of Lemma 19 we showed that, at every iteration, exactly one of these three cases happen and hence the algorithm never stalls. This completess the induction and proves that properties $P(t)$ are maintained throughout the execution of Algorithm 6, assuming that preconditions i, ii, and iii of the lemma along with the precondition $|S| \leq b$ hold.

In Lemma 19 we showed that Algorithm 6 must terminate after some q iterations. When the algorithm terminates, the condition of the *Repeat-Until* loop in line 31 of the algorithm must be True. Thus, when the algorithm terminates, at q^{th} iteration, there is no leaf in subtree $T_v^{(q)}$ besides v and as a consequence the set $\text{Marked}^{(q)}$ must be empty. This, together with $P_1(q)$ imply that the signal $\widehat{\chi}_v^{(q)}$ satisfies,

$$\text{supp}\left(\widehat{\chi}_v^{(q)}\right) = S = \text{FreqCone}_{\text{SIDETREE}}(v) \cap \text{HEAD}_\mu(\widehat{y}).$$

Moreover, $P_3(q)$ together with precondition $|S| \leq b$ imply that

$$\left\|\widehat{y}_S - \widehat{\chi}_v^{(q)}\right\|_2^2 \leq \frac{|S|}{20b}\cdot\mu^2 \leq \frac{\mu^2}{20}.$$

Now we analyze the if-statement in line 32 of the algorithm. The above equalities and inequalities on $\widehat{\chi}_v^{(q)}$ imply that,

$$\left\|\left(\widehat{y}-\widehat{\chi}_v^{(q)}\right)_{\text{FreqCone}_{\text{SIDETREE}}(v)}\right\|_2^2 = \left\|\widehat{y}_{\text{FreqCone}_{\text{SIDETREE}}(v)\setminus S}\right\|_2^2 + \left\|\left(\widehat{y}-\widehat{\chi}_v^{(q)}\right)_S\right\|_2^2$$

$$\leq \left\|\widehat{y}_{\text{FreqCone}_{\text{SIDETREE}}(v)\setminus\text{HEAD}_\mu(\widehat{y})}\right\|_2^2 + \frac{\mu^2}{20}.$$

Therefore, if $\widehat{G}_v$ is a Fourier domain $(v, \text{SIDETREE})$-isolating filter constructed in Lemma 9, then

by Corollary 1 along with the above inequality, we have

$$\left\|\left(\widehat{y}-\widehat{\chi}_v^{(q)}\right)\cdot\widehat{G}_v\right\|_2^2 \le \sum_{\boldsymbol{\xi}\in[n]^d\setminus\mathrm{supp}(\textsc{SideTree})}|\widehat{y}(\boldsymbol{\xi})|^2 + \left\|\left(\widehat{y}-\widehat{\chi}_v^{(q)}\right)_{\mathrm{FreqCone}_{\textsc{SideTree}}(v)}\right\|_2^2$$

$$\le \sum_{\boldsymbol{\xi}\in[n]^d\setminus\mathrm{supp}(\textsc{SideTree})}|\widehat{y}(\boldsymbol{\xi})|^2 + \left\|\widehat{y}_{\mathrm{FreqCone}_{\textsc{SideTree}}(v)\setminus\mathrm{HEAD}_\mu(\widehat{y})}\right\|_2^2 + \frac{\mu^2}{20}$$

$$\le \left\|\widehat{y}-\widehat{y}_{\mathrm{HEAD}_\mu(\widehat{y})}\right\|_2^2 + \frac{\mu^2}{20} \le \frac{23}{20}\cdot\mu^2.$$

Thus, the preconditions of the second claim of Lemma 17 hold. So, we can invoke this lemma to conclude that the if-statement in line 32 of the algorithm is False and hence the algorithm outputs $\left(\text{True}, \widehat{\chi}_v^{(q)}\right)$. This proves statement 1 of the lemma.

Now we prove the second statement of lemma. Suppose that preconditions i, ii, iii along with the precondition of statement 2 (that is $|S| > b$) hold. Lemma 19 proved that the signal $\widehat{\chi}_v$ always satisfies $\mathrm{supp}(\widehat{\chi}_v) \subseteq \mathrm{FreqCone}_{\textsc{SideTree}}(v)$ and $\|\widehat{\chi}_v\|_0 \le b$. Therefore, $S \setminus \mathrm{supp}(\widehat{\chi}_v) \ne \varnothing$. Consequently, if $\widehat{G}_v$ is a Fourier domain $(v, \textsc{SideTree})$-isolating filter constructed in Lemma 9, then by definition of isolating filters we have

$$\left\|\left((\widehat{y}-\widehat{\chi}_v)\cdot\widehat{G}_v\right)_{S\cup\mathrm{supp}(\widehat{\chi}_v)}\right\|_2^2 \ge \left\|(\widehat{y}-\widehat{\chi}_v)_{S\cup\mathrm{supp}(\widehat{\chi}_v)}\right\|_2^2 \ge \left\|\widehat{y}_{S\setminus\mathrm{supp}(\widehat{\chi}_v)}\right\|_2^2 \ge 9\mu^2,$$

which follows from the definition of S and $\mathrm{HEAD}_\mu(\cdot)$. On the other hand,

$$\left\|\left((\widehat{y}-\widehat{\chi}_v)\cdot\widehat{G}_v\right)_{[n]^d\setminus(S\cup\mathrm{supp}(\widehat{\chi}_v))}\right\|_2^2 = \left\|\left(\widehat{y}\cdot\widehat{G}_v\right)_{[n]^d\setminus(S\cup\mathrm{supp}(\widehat{\chi}_v))}\right\|_2^2$$

$$\le \left\|\left(\widehat{y}\cdot\widehat{G}_v\right)_{[n]^d\setminus S}\right\|_2^2$$

$$\le \left\|\widehat{y}_{\mathrm{FreqCone}_{\textsc{SideTree}}(v)\setminus S}\right\|_2^2 + \sum_{\boldsymbol{\xi}\in[n]^d\setminus\mathrm{supp}(\textsc{SideTree})}|\widehat{y}(\boldsymbol{\xi})|^2$$

$$\le \left\|\widehat{y}-\widehat{y}_{\mathrm{HEAD}_\mu(\widehat{y})}\right\|_2^2 \le \frac{11}{10}\cdot\mu^2. \qquad\qquad \text{(precondition ii)}$$

Additionally note that $|S \cup \mathrm{supp}(\widehat{\chi}_v)| \le k + b \le 2k$ by preconditions of the lemma and property of $\mathrm{supp}(\widehat{\chi}_v)$ that we have proved. Hence, by invoking the first claim of Lemma 17, the if-statement in line 32 of the algorithm is True and hence the algorithm outputs $\left(\text{False}, \{0\}^{n^d}\right)$. This proves statement 2 of the lemma.

Finally, observe that throughout this analysis we have assumed that Lemma 17 holds with probability 1 for all the invocations of HEAVYTEST by our algorithm. Moreover, we assumed that ESTIMATE successfully works with probability 1. In reality, we have to take the fact that these primitives are randomized into acount of our analysis.

The first source of randomness is the fact that HEAVYTEST only succeeds with some high probability. In fact, Lemma 17 tells us that every invocation of HEAVYTEST succeeds with probability at least $1 - 1/N^5$. Our analysis in proof of Lemma 19 shows that ROBUSTPROMISESFT makes at most $O(b\log N)$ calls to HEAVYTEST. Therefore, by a union bound, the overall failure probability of all invocations of HEAVYTEST is bounded by $O\left(\frac{b\log N}{N^5}\right)$.

The second source of randomness is the fact that ESTIMATE only succeeds with some high probability. Lemma 18 tells us that every invocation of ESTIMATE on a set Marked, succeeds with

probability $1 - |\text{Marked}|/N^8$. Therefore if the algorithm invokes ESTIMATE at iterations $t_1, t_2, \ldots$, then, by union bound, the total failure probability of all invocations of this primitive will be bounded by $\sum_i \frac{\left|\text{Marked}^{(t_i)}\right|}{N^8} = \frac{|\text{supp}(\widehat{\chi}_v)|}{N^8} \leq \frac{b}{N^8}$.

Finally, by another application of union bound, the overall failure probability of Algorithm 6, is bounded by $\frac{1}{N^4}$. This proves that the lemma holds. $\qquad\square$

Analysis of RobustSparseFT. Now we present the invariants of RobustSparseFT.

Lemma 21 (Invariant of RobustSparseFT: Signal Containment and Energy Control)**.** *For every integer $t \geq 0$, let $\widehat{\chi}^{(t)}$ and $\text{Marked}^{(t)}$ denote the signal $\widehat{\chi}$ and the set Marked at the end of iteration t of Algorithm 7, respectively. Furthermore, let FRONTIER$^{(t)}$ denote the tree FRONTIER at the end of tth iteration and let $\text{Est}^{(t)}$ denote the set of "estimated frequencies" so far, i.e., $\text{Est}^{(t)} := \text{supp}\left(\widehat{\chi}^{(t)}\right)$. Additionaly, for every leaf v of FRONTIER$^{(t)}$, let $L_v^{(t)}$ denote the "unestimated" frequencies in support of $\widehat{x}$ that lie in frequency cone of v, i.e., $L_v^{(t)} := \text{FreqCone}_{\text{FRONTIER}^{(t)}}(v) \cap \text{HEAD}_\mu(\widehat{x})$, where $\text{HEAD}_\mu(\cdot)$ is defined as per (7). If $|\text{HEAD}_\mu(\widehat{x})| \leq k$ and $\left\|\widehat{x} - \widehat{x}_{\text{HEAD}_\mu(\widehat{x})}\right\|_2 \leq \mu$, then for every non-negative integer t the following properties are maintained at the end of tth iteration of Algorithm 7, with probability at least $1 - \frac{4t}{N^4}$,*

$P_1(t)$ $\text{HEAD}_\mu(\widehat{x}) \setminus \text{Est}^{(t)} \subseteq \text{supp}\left(\text{FRONTIER}^{(t)}\right)$;

$P_2(t)$ *For every leaf $u \neq \text{root}$ of tree* FRONTIER$^{(t)}$, $\left|L_u^{(t)}\right| \geq 1$. *Additionally, if $u \notin \text{Marked}^{(t)}$, then* $\left|L_u^{(t)}\right| > b$;

$P_3(t)$ $\left\|\widehat{x}_{\text{Est}^{(t)}} - \widehat{\chi}^{(t)}\right\|_2^2 \leq \epsilon \cdot \frac{\left|\text{Est}^{(t)}\right|}{k} \cdot \mu^2$;

$P_4(t)$ $\text{Est}^{(t)} \subseteq \text{HEAD}_\mu(\widehat{x})$ *and* $\text{Est}^{(t)} \cap \text{supp}\left(\text{FRONTIER}^{(t)}\right) = \varnothing$;

$P_5(t)$ *In every iteration $t > 1$, if the if-statement in line 7 of Algorithm 7 is* False, *then the following potential function decreases by at least b. Additionally, when the if-statement in line 7 is* True, *the potential decreases by at least $\log N$. Furthermore, the potential does not increase at iteration $t = 1$.*

$$\phi_t := \sum_{u \in \text{LEAVES}\left(\text{FRONTIER}^{(t)}\right)} \left(2 \log N - l_{\text{FRONTIER}^{(t)}}(u)\right) \cdot \left|L_u^{(t)}\right|;$$

Proof. The proof is by induction on the *Repeat-Until loop* of the algorithm. The **base of induction** corresponds to the zeroth iteration ($t = 0$), at which point FRONTIER$^{(0)} = \{\text{root}\}$ is a tree that solely consists of a root and has no other leaves. Moreover, $\widehat{\chi}^{(0)} \equiv 0$. The statement $P_1(t)$ trivially holds because $\text{FreqCone}_{\text{FRONTIER}^{(0)}}(r) = [n]^d$. The statement $P_2(t)$ holds since there exists no leaf $u \neq \text{root}$ in FRONTIER$^{(0)}$. The statements $P_3(t)$ and $P_4(t)$ hold because of the facts $\widehat{\chi}^{(0)} \equiv 0$ and $\text{Est}^{(0)} = \varnothing$.

We now prove the **inductive step** by assuming that the inductive hypotheses, i.e property $P(t-1)$ is satisfied for some iteration $t-1$ of Algorithm 7 with probability a least $1 - \frac{4(t-1)}{N^4}$, and then proving that property $P(t)$ holds at the end of iteration t with probabiliy at least $1 - \frac{4t}{N^4}$. We also show that the value of the quantity ϕ_t defined in $P_5(t)$, satisfies $\phi_t - \phi_{t-1} \leq -b$ if the if-statement in line 7 of the algorithm is False in iteration $t > 1$ and $\phi_t - \phi_{t-1} \leq -\log N$ if the if-statement in line 7 is True in iteration t and also $\phi_1 - \phi_0 \leq 0$. At any given iteration t of

the algorithm, there are two possibilities that can happen. We proceed to prove the induction by considering any of the two possibilities:

Case 1 – the if-statement in line 7 of Algorithm 7 is True. In this case, we have that $\sum_{u \in \text{Marked}^{(t-1)}} 2^{-w_{\text{Frontier}^{(t-1)}}(u)} \geq \frac{1}{2}$. As a result, by Claim 6, the set $\text{Cheap} \subseteq \text{Marked}^{(t-1)}$ that the algorithm computes in line 8 by running the primitive ExtractCheapSubset satisfies the property that $|\text{Cheap}| \cdot \left(8 + 4 \log |\text{Marked}^{(t-1)}|\right) \geq \max_{u \in \text{Cheap}} 2^{w_{\text{Frontier}^{(t-1)}}(u)}$. Clearly $\text{Cheap} \neq \varnothing$, by Claim 6. Then the algorithm computes $\{\widehat{H}_u\}_{u \in \text{Cheap}}$ by running the procedure Estimate in line 10 and then updates $\widehat{\chi}^{(t)}(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$ for every $u \in \text{Cheap}$ and $\widehat{\chi}^{(t)}(\boldsymbol{\xi}) = \widehat{\chi}^{(t-1)}(\boldsymbol{\xi})$ at every other frequency $\boldsymbol{\xi}$. Therefore, if we let $L := \{\boldsymbol{f}_u : u \in \text{Cheap}\}$, then $\text{Est}^{(t)} \setminus \text{Est}^{(t-1)} = L$, by inductive hypothesis $P_4(t-1)$. By $P_3(t-1)$ along with Lemma 18, we find that with probability at least $1 - \frac{|\text{Cheap}|}{N^8} \geq 1 - \frac{1}{N^7}$ the following holds,

$$
\left\|\widehat{\chi}^{(t)} - \widehat{x}_{\text{Est}^{(t)}}\right\|_2^2 = \left\|\widehat{\chi}^{(t-1)} - \widehat{x}_{\text{Est}^{(t-1)}}\right\|_2^2 + \left\|\left(\widehat{\chi}^{(t)} - \widehat{x}\right)_L\right\|_2^2
$$
$$
\leq \frac{\epsilon |\text{Est}^{(t-1)}|}{k} \mu^2 + \frac{\epsilon |L|}{2k} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \text{supp}\left(\text{Frontier}^{(t-1)}\right)} \left|\left(\widehat{\chi}^{(t-1)} - \widehat{x}\right)(\boldsymbol{\xi})\right|^2. \qquad (15)
$$

Now we bound the second term above,

$$
\sum_{\boldsymbol{\xi} \in [n]^d \setminus \text{supp}\left(\text{Frontier}^{(t-1)}\right)} \left|\left(\widehat{x} - \widehat{\chi}^{(t-1)}\right)(\boldsymbol{\xi})\right|^2
$$
$$
= \sum_{\boldsymbol{\xi} \in [n]^d \setminus \left(\text{supp}\left(\text{Frontier}^{(t-1)}\right) \cup \text{Est}^{(t-1)}\right)} |\widehat{x}(\boldsymbol{\xi})|^2 + \left\|\widehat{x}_{\text{Est}^{(t-1)}} - \widehat{\chi}^{(t-1)}\right\|_2^2
$$
$$
\leq \sum_{\boldsymbol{\xi} \in [n]^d \setminus \text{HEAD}_\mu(\widehat{x})} |\widehat{x}(\boldsymbol{\xi})|^2 + \left\|\widehat{x}_{\text{Est}^{(t-1)}} - \widehat{\chi}^{(t-1)}\right\|_2^2 \qquad \text{(by } P_1(t-1)\text{)}
$$
$$
\leq 2\mu^2 \qquad \text{(by } P_3(t-1) \text{ and } P_4(t-1)\text{, preconditions of lemma and } \epsilon \leq 1\text{).}
$$

Therefore, by plugging the above bound back to (15) we find that,

$$
\left\|\widehat{\chi}^{(t)} - \widehat{x}_{\text{Est}^{(t)}}\right\|_2^2 \leq \epsilon \cdot \frac{|\text{Est}^{(t-1)}|}{k} \cdot \mu^2 + \epsilon \cdot \frac{|L|}{2k} \cdot \left(2\mu^2\right) = \epsilon \cdot \frac{|\text{Est}^{(t)}|}{k} \cdot \mu^2,
$$

which proves the inductive claim $P_3(t)$. Moreover, $P_2(t-1)$ implies that $L \subseteq \text{HEAD}_\mu(\widehat{x})$. Thus, the fact that $\text{Est}^{(t)} = \text{Est}^{(t-1)} \cup L$ together with inductive hypothesis $P_4(t-1)$ as well as the construction of Frontier (Frontier$^{(t)}$ is constructed by removing leaves of Cheap from tree Frontier$^{(t-1)}$), imply $P_4(t)$. The construction of Frontier$^{(t)}$ together with the fact that $|\text{FreqCone}_{\text{Frontier}^{(t-1)}}(u)| = 1$ for every $u \in \text{Cheap}$ give $P_1(t)$ and $P_2(t)$. Additionally, we have,

$$
\phi_t - \phi_{t-1} = -\sum_{u \in \text{Cheap}} (2 \log N - l_{\text{Frontier}^{(t-1)}}(u)) \cdot \left|L_u^{(t-1)}\right|
$$
$$
= -\sum_{u \in \text{Cheap}} \log N \cdot \left|L_u^{(t-1)}\right|
$$
$$
= -\sum_{u \in \text{Cheap}} \log N \leq -\log N,
$$

where the last inequality follows from the fact that $\text{Cheap} \neq \varnothing$. This proves $P_5(t)$.

58

Case 2 – the if-statement in line 7 is False. Let $v \in \text{LEAVES}(\text{FRONTIER}^{(t-1)}) \setminus \text{Marked}^{(t-1)}$ be the smallest weight leaf chosen by the algorithm in line 16. The algorithm constructs tree T by adding leaves v_{right} and v_{left} to tree $\text{FRONTIER}^{(t-1)}$ as right and left children of v, in line 19. Then, the algorithm runs ROBUSTPROMISESFT with inputs $(x, \widehat{\chi}^{(t-1)}, T, v_{\text{left}}, b, k, \mu)$ and $(x, \widehat{\chi}^{(t-1)}, T, v_{\text{right}}, b, k, \mu)$ in lines 20 and 21 respectively. In the following we focus on analyzing $(\text{ISCORR}_{\text{left}}, \widehat{\chi}_{\text{left}})$ but $(\text{ISCORR}_{\text{right}}, \widehat{\chi}_{\text{right}})$ can be analyzed exactly the same way. There are two possibilities that can happen:

Possibility 1) $|\text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{x})| \leq b$. In this case, the inductive hypothesis $P_4(t-1)$ implies that $|\text{Est}^{(t-1)}| \leq k$ and hence inductive hypothesis $P_3(t-1)$ along with the assumption $\epsilon \leq \frac{1}{10}$ gives

$$\left\| \widehat{x}_{\text{Est}^{(t-1)}} - \widehat{\chi}^{(t-1)} \right\|_2^2 \leq \epsilon \mu^2 \leq \frac{\mu^2}{10}, \tag{16}$$

hence, $\text{HEAD}_\mu\left(\widehat{x} - \widehat{\chi}^{(t-1)}\right) = \text{HEAD}_\mu(\widehat{x}) \setminus \text{Est}^{(t-1)}$. Consequently, if we let $\widehat{y} := \widehat{x} - \widehat{\chi}^{(t-1)}$, then: i) $\text{HEAD}_\mu(\widehat{y}) \subseteq \text{supp}(T')$, by $P_1(t-1)$, ii) $\|\widehat{y} - \widehat{y}_{\text{HEAD}_\mu(\widehat{y})}\|_2^2 \leq \frac{11\mu^2}{10}$, by precondition of the lemma along with (16), and iii) $|\text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})| \leq b$, by the assumption that $|\text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{x})| \leq b$. Therefore, all preconditions of the first statement of Lemma 20 hold, and thus, by invoking this lemma we have that, with probability at least $1 - \frac{1}{N^4}$, $\text{ISCORR}_{\text{left}} = \text{True}$, and $\text{supp}(\widehat{\chi}_{\text{left}}) \subseteq \text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})$, and $\|\widehat{y}_{\text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})} - \widehat{\chi}_{\text{left}}\|_2^2 \leq \frac{\mu^2}{20}$. This together with inductive hypothesis $P_4(t-1)$ imply that, with probability at least $1 - \frac{1}{N^4}$, $\text{ISCORR}_{\text{left}} = \text{True}$ and $\text{supp}(\widehat{\chi}_{\text{left}}) = \text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{x})$.

So, the if-statement in line 22 of the algorithm is True and consequently the algorithm adds all leaves that correspond to frequencies in $\text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{x})$ to $\text{FRONTIER}^{(t-1)}$ and also updates

$$\text{Marked}^{(t)} \leftarrow \text{Marked}^{(t-1)} \cup \{u \in \text{LEAVES}(\text{FRONTIER}) : \boldsymbol{f}_u \in \text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{x})\}.$$

By a similar argument, if $|\text{FreqCone}_T(v_{\text{right}}) \cap \text{HEAD}_\mu(\widehat{x})| \leq b$, then, with probability at least $1 - \frac{1}{N^4}$, the algorithm adds all leaves corresponding to frequencies in $\text{FreqCone}_T(v_{\text{right}}) \cap \text{HEAD}_\mu(\widehat{x})$ to $\text{FRONTIER}^{(t-1)}$ and updates

$$\text{Marked}^{(t)} \leftarrow \text{Marked}^{(t-1)} \cup \{u \in \text{LEAVES}(\text{FRONTIER}) : \boldsymbol{f}_u \in \text{FreqCone}_T(v_{\text{right}}) \cap \text{HEAD}_\mu(\widehat{x})\}.$$

Possibility 2) $|\text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{x})| > b$. Same as in **possibility 1**, the inductive hypothesis $P_4(t-1)$ implies that $|\text{Est}^{(t-1)}| \leq k$ and hence inductive hypothesis $P_3(t-1)$ along with the assumption $\epsilon \leq \frac{1}{10}$ gives (16). Hence, $\text{HEAD}_\mu\left(\widehat{x} - \widehat{\chi}^{(t-1)}\right) = \text{HEAD}_\mu(\widehat{x}) \setminus \text{Est}^{(t-1)}$. Consequently, if we let $\widehat{y} := \widehat{x} - \widehat{\chi}^{(t-1)}$, then it holds that: i) $\text{HEAD}_\mu(\widehat{y}) \subseteq \text{supp}(T)$, by $P_1(t-1)$, ii) $\|\widehat{y} - \widehat{y}_{\text{HEAD}_\mu(\widehat{y})}\|_2^2 \leq \frac{11\mu^2}{10}$, by precondition of the lemma along with (16), and iii) $|\text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})| \leq |\text{HEAD}_\mu(\widehat{x})| \leq k$, by precondition of the lemma. Additionally, by $P_4(t-1)$, we find that

$$|\text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})| = |\text{FreqCone}_T(v_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{x})| > b.$$

Therefore, all preconditions of the second statement of Lemma 20 hold, and thus, by invoking this lemma we have that, with probability at least $1 - \frac{1}{N^4}$, $\text{ISCORR}_{\text{left}} = \text{False}$, and $\widehat{\chi}_{\text{left}} \equiv 0$. So, the if-statement in line 22 of the algorithm is False and consequently the algorithm adds leaf v_{left} as the left child of v to tree $\text{FRONTIER}^{(t-1)}$. By a similar argument, if $|\text{FreqCone}_T(v_{\text{right}}) \cap \text{HEAD}_\mu(\widehat{x})| > b$, then, with probability $1 - \frac{1}{N^4}$, the algorithm adds leaf v_{right} as the left child of v to tree $\text{FRONTIER}^{(t-1)}$.

Based on the above arguments, according to the values of $\text{ISCORR}_{\text{left}}$ and $\text{ISCORR}_{\text{right}}$, there are various cases that can happen. From the way tree $\text{FRONTIER}^{(t)}$ and set $\text{Marked}^{(t)}$ are obtained

from $\text{FRONTIER}^{(t-1)}$ and $\text{Marked}^{(t-1)}$, it follows that in any case, the first 4 properties of $P(t)$ are maintained with probability at least $1 - \frac{2}{N^4}$. Furthermore, the way tree $T^{(t)}$ is constructed implies that,

$$\sum_{u \in \text{LEAVES}\left(\text{FRONTIER}_v^{(t)}\right)} \left|L_u^{(t)}\right| = \left|L_v^{(t-1)}\right|.$$

Therefore, for every $t > 1$, by inductive hypothesis $P_2(t-1)$, the change in potential is bounded as follows,

$$\phi_t - \phi_{t-1} = \sum_{u \in \text{LEAVES}\left(\text{FRONTIER}_v^{(t)}\right)} \left(2 \log N - l_{\text{FRONTIER}^{(t)}}(u)\right) \cdot \left|L_u^{(t)}\right| - \left(2 \log N - l_{\text{FRONTIER}^{(t-1)}}(v)\right) \cdot \left|L_v^{(t-1)}\right|$$

$$\leq - \left|L_v^{(t-1)}\right| < -b.$$

Moreover, if $t = 1$ then the change in potential satisfies $\phi_1 - \phi_0 \leq - \left|L_v^{(t-1)}\right| \leq 0$ (because in this case $v = \text{root}$). This proves the inductive claim $P_5(t)$.

We have proved that for every t, if the inductive hypothesis $P(t-1)$ is satisfied then the property $P(t)$ is maintained with probability at least $1 - \frac{2}{N^4} - \frac{1}{N^7} \geq 1 - \frac{4}{N^4}$. Therefore, using the inductive hypothesis that $\Pr[P(t-1)] \geq 1 - \frac{4(t-1)}{N^4}$, by using union bound we find that

$$\Pr[P(t)] \geq \Pr[P(t)|P(t-1)] \cdot \Pr[P(t-1)] \geq 1 - \frac{4t}{N^4}.$$

This complets the proof of the lemma. $\qquad\square$

Now we are in a position to prove the main result of this section.

Proof of Theorem 13. The proof basically follows by invoking Lemma 21 and then analyzing the runtime and sample complexity of Algorithm 7. If we let $\mu := \|\eta\|_2$ then because x is a signal in the k-high SNR regime, we have that $|\text{HEAD}_\mu(\widehat{x})| \leq k$ and $\left\|\widehat{x} - \widehat{x}_{\text{HEAD}_\mu(\widehat{x})}\right\|_2 \leq \mu$. Therefore, if we run the procedure ROBUSTSPARSEFT (Algorithm 7) with inputs (x, k, ϵ, μ), then the preconditions of Lemma 21 hold and hence by invoking this lemma we conclude that all the invariants $P_1(t)$ through $P_5(t)$, defined in Lemma 21, hold throughout the execution of Algorithm 7 for every non-negative integer t.

Using this, we first prove the termination of the algorithm. Let $q = O\left(k + \frac{k \log N}{b}\right)$ be some large enough integer. We show that the algorithm must terminate in q iterations. Note that the probability that the properties $P(t)$ hold for all iterations $t \in \{0, 1, \ldots q\}$ of algorithm RO-BUSTSPARSEFFT is at least $1 - \frac{4(q+1)}{N^4} \geq 1 - \frac{1}{N^3}$, by Lemma 21. From now on, we condition on the event corresponding to $P(t)$ holding for all iterations $t \in \{0, 1, \ldots q\}$, which holds with probability at least $1 - \frac{1}{N^3}$. Conditioned on this event we prove that the algorithm terminates in less than q iterations.

Note that, the potential function ϕ_t defined in $P_5(t)$ is non-negative for every t. Moreover, at the zeroth iteration of the algorithm $T^{(0)} = \{\text{root}\}$ and hence $L_{\text{root}}^{(0)} = \text{HEAD}_\mu(\widehat{x})$, thus

$$\phi_0 \leq 2k \log N.$$

Therefore, it follows from $P_5(t)$ that Algorithm 7 must terminate in at most $q = O\left(k + \frac{k \log N}{b}\right)$ iterations.

When the algorithm terminates, the condition of the *Repeat-Until* loop in line 32 of the algorithm must be True. Thus, when the algorithm terminates, there is no leaf in tree $T^{(q)}$ besides the root. Cosequently, by invariants $P_1(q)$ and $P_3(q)$, the output of the algorithm satisfies, $\text{HEAD}_\mu(\widehat{x}) \subseteq \text{supp}(\widehat{\chi})$ and $\|\widehat{x}_{\text{Est}} - \widehat{\chi}\|_2^2 \leq \frac{\epsilon|\text{Est}|}{k} \cdot \mu^2$, where $\text{Est} = \text{supp}(\widehat{\chi})$. Using the invariant $P_4(q)$, the latter can be Further upper bounded as $\|\widehat{x}_{\text{Est}} - \widehat{\chi}\|_2^2 \leq \epsilon \cdot \mu^2$. This together with the k-high SNR assumption of the theorem gives the approximation guarantee of the theorem $\|\widehat{x} - \widehat{\chi}\|_2^2 \leq (1 + \epsilon) \cdot \|\eta\|_2^2$.

Runtime and Sample Complexity. The expensive components of the algorithm are primitive ESTIMATE in line 10 and primitive ROBUSTPROMISESFT in lines 20 and 21 of the algorithm. We first bound the time and sample complexity of invoking ESTIMATE in line 10. We remark that, at any iteration t, the algorithm runs primitive ESTIMATE only if **case 1** that we mentioned earlier in the proof happens. Therefore, in this case, the set $\varnothing \neq \text{Cheap}^{(t)} \subseteq \text{Marked}^{(t-1)}$ that our algorithm computes in line 8 by running the primitive EXTRACTCHEAPSUBSET satisfies the property that $\left|\text{Cheap}^{(t)}\right| \cdot \left(8 + 4\log\left|\text{Marked}^{(t-1)}\right|\right) \geq \max_{u \in \text{Cheap}^{(t)}} 2^{w_{\text{FRONTIER}^{(t-1)}}(u)}$. By $P_2(t-1)$, and k-high SNR assumption, this implies that $\left|\text{Cheap}^{(t)}\right| \cdot (8 + 4\log k) \geq \max_{u \in \text{Cheap}^{(t)}} 2^{w_{\text{FRONTIER}^{(t-1)}}(u)}$.

Thus, by Lemma 18, the runtime and sample complexity of every invocation of ESTIMATE in line 10 of our algorithm are bounded by $\widetilde{O}\left(\frac{k}{\epsilon \cdot |\text{Cheap}^{(t)}|} \sum_{u \in \text{Cheap}^{(t)}} 2^{w_{\text{FRONTIER}^{(t-1)}}(u)} + \frac{k}{\epsilon}\|\widehat{\chi}^{(t-1)}\|_0\right)$ and $\widetilde{O}\left(\frac{k}{\epsilon \cdot |\text{Cheap}^{(t)}|} \sum_{u \in \text{Cheap}^{(t)}} 2^{w_{\text{FRONTIER}^{(t-1)}}(u)}\right)$, respectively. Using $P_4(t-1)$, the runtime and sample complexity of ESTIMATE can be further upper bounded by $\widetilde{O}\left(\frac{k}{\epsilon} \cdot \left|\text{Cheap}^{(t)}\right| + \frac{k^2}{\epsilon}\right)$ and $\widetilde{O}\left(\frac{k}{\epsilon} \cdot \left|\text{Cheap}^{(t)}\right|\right)$, respectively. By property $P_5(t)$ we find that the total number of iterations in which **case 1** happens, and hence number of times we run ESTIMATE in line 10 of the algorithm, is bounded by $O(k)$. Using this together with the fact that $\sum_{t: \text{ if-statement in line 7 is True}} \left|\text{Cheap}^{(t)}\right| = \|\widehat{\chi}\|_0 \leq k$, the total runtime and sample complexity of all invocations of ESTIMATE in all iterations can be upper bounded by $\widetilde{O}\left(\frac{k^3}{\epsilon}\right)$ and $\widetilde{O}\left(\frac{k^2}{\epsilon}\right)$, respectively.

Now we bound the runtime and sample complexity of invoking ROBUSTPROMISESFT in lines 20 and 21 of the algorithm. Note that at any iteration t, the algorithm runs ROBUSTPROMISESFT in lines 20 and 21 only if **case 2** that we mentioned earlier in the proof happens. Since we pick leaf v in line 16 of the algorithm with smallest weight, and since the number of leaves that are not in the set $\text{Marked}^{(t-1)}$ are bounded by $\frac{k}{b}$ (by invariant $P_2(t-1)$), we have $w_{T^{(t-1)}}(v) \leq \log\frac{k}{b}$. Also note that $\left\|\widehat{\chi}^{(t-1)}\right\|_0 \leq k$ by invariant $P_4(t-1)$ and the k-high SNR assumption.

Therefore, by Lemma 19, the runtime and sample complexity of each invokation of ROBUSTPROMISESFT by our algorithm are bounded by $\widetilde{O}\left(k \cdot (b^2 + k) + \frac{k}{b} \cdot (b^3 + k)\right)$ and $\widetilde{O}\left(\frac{k}{b} \cdot (b^3 + k)\right)$. By property $P_5(t)$ we find that the total number of iterations in which **case 2** happens, and hence the number of times we run ROBUSTPROMISESFT in lines 20 and 21 of the algorithm, is bounded by $O\left(\frac{k \log N}{b}\right)$. Therefore, by using $b \approx k^{1/3}$, we find that the total runtime and sample complexity of all invocations of ROBUSTPROMISESFT are bounded by $\widetilde{O}\left(k^{8/3}\right)$ and $\widetilde{O}\left(k^{7/3}\right)$, respectively. Hence, the total time and sample complexity of the algorithm are bounded by $\widetilde{O}\left(\frac{k^3}{\epsilon}\right)$ and $\widetilde{O}\left(k^{7/3} + \frac{k^2}{\epsilon}\right)$, respectively.

12.3 Proving the Correctness of our Computational Primitives.

In this subsection, we shall prove Lemmas 17, 18, and Claim 6. We proceed by proving them in the aforementioned order.

Proof of Lemma 17:

By convolution-multiplication theorem, h_Δ^z computed in line 8 of Algorithm 4 satisfies $h_\Delta^z = N \cdot (\chi \star G_v)(\Delta)$, and thus

$$H^z = \frac{1}{|\mathrm{RIP}_m^z|} \sum_{\Delta \in \mathrm{RIP}_m^z} \left| N \cdot \sum_{\boldsymbol{j} \in [n]^d} G_v(\Delta - \boldsymbol{j}) \cdot x(\boldsymbol{j}) - h_\Delta^z \right|^2$$

$$= \frac{N^2}{|\mathrm{RIP}_m^z|} \sum_{\Delta \in \mathrm{RIP}_m^z} |((x - \chi) \star G_v)(\Delta)|^2.$$

Therefore, by the convolution-multiplication duality and using the definition $\widehat{y} := (\widehat{x} - \widehat{\chi}) \cdot \widehat{G}_v$, if we let y be the inverse Fourier transform of $\widehat{y}$, we find that for every $z \in [32 \log N]$,

$$H^z = \frac{N^2}{|\mathrm{RIP}_m^z|} \sum_{\Delta \in \mathrm{RIP}_m^z} |y(\Delta)|^2.$$

We first prove the **first claim of the Lemma**. Let us write $\widehat{y} = \widehat{y}_S + \widehat{y}_{\bar{S}}$, where $\widehat{y}_S \in \mathbb{C}^{n^d}$ is defined as $\widehat{y}_S(\boldsymbol{f}) := \widehat{y}(\boldsymbol{f}) \cdot \mathbb{1}_{\{\boldsymbol{f} \in S\}}$ and $\widehat{y}_{\bar{S}} \in \mathbb{C}^{n^d}$ is defined as $\widehat{y}_{\bar{S}}(\boldsymbol{f}) := \widehat{y}(\boldsymbol{f}) \cdot \mathbb{1}_{\{\boldsymbol{f} \notin S\}}$. By the assumption of lemma $\|\widehat{y}_S\|_2^2 > \frac{11\theta}{10}$. Let y_S and $y_{\bar{S}}$ denote the inverse Fourier transform of $\widehat{y}_S$ and $\widehat{y}_{\bar{S}}$ respectively. We have $y = y_S + y_{\bar{S}}$. Thus we find that,

$$\frac{1}{|\mathrm{RIP}_m^z|} \sum_{\Delta \in \mathrm{RIP}_m^z} |y(\Delta)|^2 = \frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} |y_S(\Delta) + y_{\bar{S}}(\Delta)|^2$$

$$= \frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} |y_S(\Delta)|^2 + |y_{\bar{S}}(\Delta)|^2 + 2\Re\{y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)\}$$

$$\geq \frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} |y_S(\Delta)|^2 + 2\Re\{y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)\}$$

First note that since $\widehat{y}_S$ is $|S|$-sparse and because we assumed $m = \Omega\left(|S| \log^2 |S| \log N\right)$ and because Δ's are i.i.d. uniform samples from $[n]^d$, by Theorem 9,

$$\Pr\left[\frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} |y_S(\Delta)|^2 \geq 0.99 \cdot \frac{\|\widehat{y}_S\|_2^2}{N^2}\right] \geq 1 - \frac{1}{N^2}. \tag{17}$$

Now it suffices to bound the term $\frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} 2\Re\{y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)\}$. First, note that

$$\mathbb{E}\left[\frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} 2\Re\{y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)\}\right] = \frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} \mathbb{E}[y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)] + \mathbb{E}[y_S(\Delta) \cdot y_{\bar{S}}(\Delta)^*]$$

$$= \frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} \frac{1}{N} \langle y_S, y_{\bar{S}} \rangle + \frac{1}{N} \langle y_{\bar{S}}, y_S \rangle$$

$$= \frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} \frac{1}{N^2} \langle \widehat{y}_S, \widehat{y}_{\bar{S}} \rangle + \frac{1}{N^2} \langle \widehat{y}_{\bar{S}}, \widehat{y}_S \rangle$$

$$= 0,$$

where the last line follows because the support of $\widehat{y}_{\bar{S}}$ and $\widehat{y}_S$ are disjoint. We proceed by bounding the second moment of the quantity $\frac{1}{m} \sum_{\Delta \in \mathrm{RIP}_m^z} 2\Re\{y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)\}$ as follows,

$$
\mathbb{E}\left[\left|\frac{1}{m}\sum_{\Delta \in \mathrm{RIP}_m^z} 2\Re\{y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)\}\right|^2\right] \leq \mathbb{E}\left[\left|\frac{2}{m}\sum_{\Delta \in \mathrm{RIP}_m^z} y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)\right|^2\right]
$$

$$
= \frac{4}{m}\mathbb{E}\left[|y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)|^2\right] \quad \text{(By independence of Δ's)}
$$

$$
\leq \frac{4}{m}\mathbb{E}\left[\|y_S\|_\infty^2 |y_{\bar{S}}(\Delta)|^2\right]
$$

$$
= \frac{4}{m}\|y_S\|_\infty^2 \mathbb{E}\left[|y_{\bar{S}}(\Delta)|^2\right]
$$

$$
= \frac{4}{m}\|y_S\|_\infty^2 \frac{\|\widehat{y}_{\bar{S}}\|_2^2}{N^2}
$$

By Chebyshev's inequality we have the following,

$$
\Pr\left[\left|\frac{1}{m}\sum_{\Delta \in \mathrm{RIP}_m^z} 2\Re\{y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)\}\right| \geq 1/20 \cdot \frac{\|\widehat{y}_S\|_2^2}{N^2}\right] \leq \frac{1600 N^2 \|y_S\|_\infty^2 \|\widehat{y}_{\bar{S}}\|_2^2}{m\|\widehat{y}_S\|_2^4}
$$

$$
\leq \frac{1600\|\widehat{y}_S\|_1^2\|\widehat{y}_{\bar{S}}\|_2^2}{m\|\widehat{y}_S\|_2^4}
$$

$$
\leq \frac{1600|S| \cdot \|\widehat{y}_S\|_2^2\|\widehat{y}_{\bar{S}}\|_2^2}{m\|\widehat{y}_S\|_2^4} \quad \text{(Cauchy-Schwarz)}
$$

$$
= \frac{1600|S| \cdot \|\widehat{y}_{\bar{S}}\|_2^2}{m\|\widehat{y}_S\|_2^2}.
$$

Therefore because we assumed that $m = \Omega\left(|S|\frac{\|\widehat{y}\|_2^2}{\|\widehat{y}_S\|_2^2}\right)$, the following holds,

$$
\Pr\left[\left|\frac{1}{m}\sum_{\Delta \in \mathrm{RIP}_m^z} 2\Re\{y_S(\Delta)^* \cdot y_{\bar{S}}(\Delta)\}\right| \geq 1/20 \cdot \frac{\|\widehat{y}_S\|_2^2}{N^2}\right] \leq 1/10.
$$

Combining the above inequality with (17) using union bound gives,

$$
\Pr\left[H^z \leq 0.94 \cdot \|\widehat{y}_S\|_2^2\right] \leq 1/8.
$$

Since in line 11 of the algorithm we compare $\mathrm{MEDIAN}_{z \in [32\log N]}\{H^z\}$ to θ, using the fact that $\|\widehat{y}_S\|_2^2 > \frac{11\theta}{10}$, we have the following,

$$
\Pr[\mathrm{HEAVYTEST} = \mathrm{False}] \leq \Pr\left[\mathrm{MEDIAN}_{z \in [32\log N]}\{H^z\} \leq 10/11 \cdot \|\widehat{y}_S\|_2^2\right]
$$

$$
\leq \binom{32\log N}{16\log N}\frac{1}{8^{16\log N}}
$$

$$
\leq \frac{2^{32\log N}}{8^{16\log N}} = \frac{1}{N^{16}}.
$$

This completes the proof of the first claim.

The proof of the **second claim of the lemma** is more straightforward. The expected value of H^z is,

$$
\mathbb{E}[H^z] = \frac{N^2}{|\mathrm{RIP}_m^z|}\sum_{\Delta \in \mathrm{RIP}_m^z}\mathbb{E}\left[|y(\Delta)|^2\right] = \|\widehat{y}\|_2^2.
$$

63

Therefore by Markov's inequality we find that for every $z \in [32 \log N]$,

$$\Pr\left[H^z \geq 5\|\widehat{y}\|_2^2\right] \leq 1/5.$$

The assumption of the lemma in this case is that $\|\widehat{y}\|_2^2 \leq \theta/5$, thus we have,

$$\Pr\left[\text{HeavyTest} = \text{True}\right] \leq \Pr\left[\text{Median}_{z \in [32 \log N]}\left\{H_{\boldsymbol{f}}^z\right\} > 5 \cdot \|\widehat{y}_S\|_2^2\right]$$

$$\leq \binom{32 \log N}{16 \log N} \frac{1}{5^{16 \log N}}$$

$$\leq \frac{2^{32 \log N}}{5^{16 \log N}} = \frac{1}{N^5}.$$

This completes the proof of the second claim of the lemma.

Sample Complexity and Runtime: Computing the filters $(G_v, \widehat{G}_v)$ uses $O\left(2^{w_T(v)} + \log N\right)$ runtime, by Lemma 9. Given filter $\widehat{G}_v$, computing the quantities h_Δ^z for all Δ and z in line 8 of the algorithm uses $O\left(\|\widehat{\chi}\|_0 \cdot \sum_z |\text{RIP}_m^z|\right) = O\left(\|\widehat{\chi}\|_0 \cdot m \log N\right)$ time. Given filter G_v with $|\text{supp}(G_v)| = 2^{w_T(v)}$, computing the quantity H^z for all z requires $O\left(2^{w_T(v)} \cdot \sum_z |\text{RIP}_m^z|\right) = O\left(2^{w_T(v)} \cdot m \log N\right)$ accesses to the signal x and $O\left(2^{w_T(v)} \cdot m \log N\right)$ runtime. Therefore, the total sample complexity of the algorithm is $O\left(2^{w_T(v)} \cdot m \log N\right)$ and the total runtime of the algorithm is $O\left(2^{w_T(v)} \cdot m \log N + \|\widehat{\chi}\|_0 \cdot m \log N\right)$

$\square$

Proof of Lemma 18: Note that the algorithm constructs (v, T)-isolating filters $(G_v, \widehat{G}_v)$ for every leaf $v \in S$. By Lemma 9, constructing filters G_v and $\widehat{G}_v$ takes time $O\left(2^{w_T(v)} + \log N\right)$. Moreover, Lemma 9 tells us that filter G_v has support size $|\text{supp}(G_v)| = 2^{w_T(v)}$ and $\widehat{G}_v$ can be accessed at any frequency using $O(\log N)$ operations.

Therefore, for every fixed $v \in S$, computing $h_v^z = \sum_{\Delta \in \text{RIP}_m^z} e^{-2\pi i \frac{\boldsymbol{f}^\top \Delta}{n}} \sum_{\boldsymbol{\xi} \in [n]^d} e^{2\pi i \frac{\boldsymbol{\xi}^T \Delta}{n}} \cdot \widehat{\chi}_{\boldsymbol{\xi}} \cdot \widehat{G}_v(\boldsymbol{\xi})$ in line 7 of Algorithm 5 can be done in total time $O\left(|\text{RIP}_m^z| \log N \cdot \|\widehat{\chi}\|_0\right) = O\left(B \log N \cdot \|\widehat{\chi}\|_0\right)$ for all z.

By convolution-multiplication duality theorem, h_v^z satisfies $h_v^z = N \cdot \sum_{\Delta \in \text{RIP}_m^z} e^{-2\pi i \frac{\boldsymbol{f}^\top \Delta}{n}} (\chi \star G_v)(\Delta)$, and thus, for every leaf $v \in S$:

$$H_v^z = \frac{1}{|\text{RIP}_m^z|} \cdot \left(N \cdot \sum_{\Delta \in \text{RIP}_m^z} \left(e^{-2\pi i \frac{\boldsymbol{f}^\top \Delta}{n}} \sum_{\boldsymbol{j} \in [n]^d} G_v(\Delta - \boldsymbol{j}) \cdot x(\boldsymbol{j})\right) - h_v^z\right)$$

$$= \frac{N}{|\text{RIP}_m^z|} \sum_{\Delta \in \text{RIP}_m^z} e^{-2\pi i \frac{\boldsymbol{f}^\top \Delta}{n}} ((x - \chi) \star G_v)(\Delta).$$

To simplify the notation, let us use $y_v := (x - \chi) \star G_v$. Because G_v is (v, T)-isolating, by Definition 12, we have that $\widehat{y}_v(\boldsymbol{\xi}) = 0$ for every $\boldsymbol{\xi} \in \bigcup_{\substack{u \in \text{Leaves}(T) \\ u \neq v}} \text{FreqCone}_T(u)$ and also $\widehat{y}_v(\boldsymbol{f}) = \widehat{(x - \chi)}(\boldsymbol{f})$, where $\boldsymbol{f} := \boldsymbol{f}_v$ is the frequency label of the leaf v. Using these facts together with the above equality and the assumption of the lemma on $\text{IsIdentified}(T, v) = \text{True}$, we can write,

$$H_v^z = \frac{N}{|\text{RIP}_m^z|} \sum_{\Delta \in \text{RIP}_m^z} e^{-2\pi i \frac{\boldsymbol{f}^\top \Delta}{n}} y_v(\Delta)$$

$$= \widehat{y}_v(\boldsymbol{f}) + \frac{1}{|\text{RIP}_m^z|} \sum_{\Delta \in \text{RIP}_m^z} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \text{supp}(T)} e^{2\pi i \frac{(\boldsymbol{\xi} - \boldsymbol{f})^\top \Delta}{n}} \cdot \widehat{y}_v(\boldsymbol{\xi}).$$

64

We continue by computing the expectation of the above quantity. Since $\boldsymbol{f} \in \mathrm{FreqCone}_T(v)$, $\boldsymbol{\xi} - \boldsymbol{f} \neq 0$ for every $\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)$, which in turn implies that,

$$\mathbb{E}\left[H_v^z\right] = \widehat{y}_v(\boldsymbol{f}) + \frac{1}{|\mathrm{RIP}_m^z|} \sum_{\Delta \in \mathrm{RIP}_m^z} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \mathbb{E}_\Delta\left[e^{2\pi i \frac{(\boldsymbol{\xi} - \boldsymbol{f})^\top \Delta}{n}}\right] \widehat{y}_v(\boldsymbol{\xi}) = \widehat{y}_v(\boldsymbol{f}).$$

In the above expectation we used the fact that Δ is distributed uniformly on $[n]^d$. Next we compute the second moment of H_v^z. We have,

$$\mathbb{E}\left[|H_v^z - \widehat{y}_v(\boldsymbol{f})|^2\right] = \frac{1}{|\mathrm{RIP}_m^z|^2} \sum_{\Delta \in \mathrm{RIP}_m^z} \mathbb{E}\left[\left|\sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} e^{2\pi i \frac{(\boldsymbol{\xi} - \boldsymbol{f})^\top \Delta}{n}} \widehat{y}_v(\boldsymbol{\xi})\right|^2\right] \quad \text{(by independence of } \Delta\text{'s)}$$

$$= \frac{1}{|\mathrm{RIP}_m^z|} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} |\widehat{y}_v(\boldsymbol{\xi})|^2 \quad \text{(since } \Delta \text{ is uniform over } [n]^d \text{ and } \boldsymbol{\xi} - \boldsymbol{f} \neq 0)$$

$$= \frac{1}{B} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \left|\widehat{(x - \chi)}(\boldsymbol{\xi}) \cdot \widehat{G}_v(\boldsymbol{\xi})\right|^2. \quad \text{(by definition of } y)$$

In the final line above we used the fact that the multiset RIP_m^z defined in Algorithm 5 has size m. Therefore, Markov's inequality implies that for every $z \in [16 \log N]$,

$$\Pr\left[|H_v^z - \widehat{y}_v(\boldsymbol{f})|^2 \geq \frac{8}{m} \cdot \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \left|\widehat{(x - \chi)}(\boldsymbol{\xi}) \cdot \widehat{G}_v(\boldsymbol{\xi})\right|^2\right] \leq \frac{1}{8}.$$

Since in line 9 of Algorithm 5 we set $\widehat{H}_v = \mathrm{MEDIAN}_{z \in [16 \log N]}\{H_v^z\}$, where the median of real and imaginary parts are computed separately, we find that

$$\Pr\left[\left|\widehat{H}_v - \widehat{y}_v(\boldsymbol{f})\right|^2 \geq \frac{16}{m} \cdot \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \left|\widehat{(x - \chi)}(\boldsymbol{\xi}) \cdot \widehat{G}_v(\boldsymbol{\xi})\right|^2\right] \leq \binom{16 \log N}{8 \log N} \frac{1}{8^{8 \log N}}$$

$$\leq \frac{2^{16 \log N}}{8^{8 \log N}} = \frac{1}{N^8}.$$

By recalling that $\widehat{y}_v(\boldsymbol{f}) = \widehat{(x - \chi)}(\boldsymbol{f}_v)$ for every $v \in S$ and applying union bound we find that,

$$\Pr\left[\sum_{v \in S} \left|\widehat{H}_v - \widehat{(x - \chi)}(\boldsymbol{f}_v)\right|^2 \geq \frac{16}{m} \cdot \sum_{v \in S} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \left|\widehat{(x - \chi)}(\boldsymbol{\xi}) \cdot \widehat{G}_v(\boldsymbol{\xi})\right|^2\right] \leq \frac{|S|}{N^8}. \quad (18)$$

In the last step, we bound the quantity $\sum_{v \in S} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \left|\widehat{(x - \chi)}(\boldsymbol{\xi}) \cdot \widehat{G}_v(\boldsymbol{\xi})\right|^2$ as follows,

$$\sum_{v \in S} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \left|\widehat{(x - \chi)}(\boldsymbol{\xi}) \cdot \widehat{G}_v(\boldsymbol{\xi})\right|^2 = \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \left|\widehat{(x - \chi)}(\boldsymbol{\xi})\right|^2 \cdot \sum_{v \in S} \left|\widehat{G}_v(\boldsymbol{\xi})\right|^2$$

$$\leq \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \left|\widehat{(x - \chi)}(\boldsymbol{\xi})\right|^2 \cdot \sum_{v \in \mathrm{LEAVES}(T)} \left|\widehat{G}_v(\boldsymbol{\xi})\right|^2$$

$$= \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)} \left|\widehat{(x - \chi)}(\boldsymbol{\xi})\right|^2, \quad \text{(By Lemma 16)}$$

65

hence, plugging the above bound into (18) gives,

$$\Pr\left[\sum_{v \in S}\left|\widehat{H}_v - \widehat{(x - \chi)}(\boldsymbol{f}_v)\right|^2 \geq \frac{16}{m} \cdot \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(T)}\left|\widehat{(x - \chi)}(\boldsymbol{\xi})\right|^2\right] \leq \frac{|S|}{N^8}.$$

□

Lastly, we prove the correctness of EXTRACTCHEAPSUBSET, and in particular Claim 6.

Proof of Claim 6: First let $S' := \left\{u \in S : 2^{w_T(u)} \leq 4|S|\right\}$. It easily follows that $\sum_{u \in S'} 2^{-w_T(u)} \geq \frac{1}{4}$. For every $j = 0, 1, \ldots \lfloor \log(4|S|)\rfloor$, let L_j denote the subset of S' defined as $L_j := \{u : u \in S', w_T(u) = j\}$. We can write,

$$\sum_{u \in S'} 2^{-w_T(u)} = \sum_{j=0}^{\lfloor \log(4|S|)\rfloor} \frac{|L_j|}{2^j}$$

Therefore, by the fact that $\sum_{u \in S'} 2^{-w_T(u)} \geq \frac{1}{4}$, we have that there must exist an integer $j \in \{0, 1, \ldots \lfloor \log(4|S|)\rfloor\}$ such that $\frac{|L_j|}{2^j} \geq \frac{1}{4\lceil \log(4|S|)\rceil}$. Hence, there must exist a set $L \subseteq S$ such that $|L| \cdot (8 + 4\log|S|) \geq \max_{v \in L} 2^{w_T(v)}$. The primitive EXTRACTCHEAPSUBSET finds this set L efficiently.
□

13 Robust Sparse Fourier Transform II.

In this section we present an algorithm that can compute a $1 + \epsilon$ approximation to the Fourier transform of a singnal in the k-high SNR regime using a sample complexity that is nearly quadratic in k and a runtime that is cubic in k, fully making use of techniques I-IV.

Formally we prove the following theorem,

Theorem 4 (Robust Sparse Fourier Transform with Near-quadratic Sample Complexity). [11] *Given oracle access to $x : [n]^d \to \mathbb{C}$ in the k-high SNR model and parameter $\epsilon > 0$, we can solve the ℓ_2/ℓ_2 Sparse Fourier Transform problem with high probability in N using*

$$m = \widetilde{O}\left(\frac{k^2}{\epsilon} + k^2 \cdot 2^{\Theta\left(\sqrt{\log k \cdot \log \log N}\right)}\right)$$

samples from x and $\widetilde{O}\left(\frac{k^3}{\epsilon}\right)$ running time.

We first present a recursive procedure in Algorithm 8 that is the main computational component of achieving the abovementioned theorem for a constant value of $\epsilon = \frac{1}{20}$. Any sparse $\widehat{\chi}$ that satisfies the approximation guarantee of Theorem 4 for constant ϵ, by the k-high SNR assumption, must recover all the *head* elements of $\widehat{x}$ correctly. Once we have the set of heavy frequencies of $\widehat{x}$ we can estimate the head vlaues to a higher ϵ precision for arbitrarily small ϵ using a simple algorithm. We present the procedure that achieves such $1 + \epsilon$ approximation and thus achieves the guarantee of Theorem 4 in Algorithm 9. We demonstrate the execution of primitive RECURSIVEROBUSTSFT (Algorithm 8) in Figure 5.

Overview of RecursiveRobustSFT (Algorithm 8): Consider an invocation of RECURSIVEROBUSTSFT$(x, \widehat{\chi}_{in}, \text{FRONTIER}, v, k, \alpha, \mu)$. Suppose that $\widehat{y} := \widehat{x} - \widehat{\chi}_{in}$ is a signal in the high SNR regime, i.e., the value of each heavy frequency of signal $\widehat{y}$ is at least 3 times higher than the tail's norm. More formally, let HEAD $\subseteq [n]^d$ denote the set of heavy (head) frequencies of $\widehat{y}$ and suppose that the tail norm of $\widehat{y}$ satisfies $\|\widehat{y} - \widehat{y}_{\text{HEAD}}\|_2 \leq \mu$ and additionally suppose that $|\widehat{y}(\boldsymbol{f})| \geq 3\mu$ for every $\boldsymbol{f} \in$ HEAD. If FRONTIER fully captures the heavy frequencies of $\widehat{y}$, i.e., HEAD $\subseteq$ supp(FRONTIER), and the number of heavy frequencies in frequency cone of node v is bounded by k, i.e., $|\text{HEAD} \cap \text{FreqCone}_{\text{FRONTIER}}(v)| \leq k$, then RECURSIVEROBUSTSFT finds a signal $\widehat{\chi}_v$ such that supp$(\widehat{\chi}_v) = $ HEAD $\cap \text{FreqCone}_{\text{FRONTIER}}(v) := S$ and $\|\widehat{y}_S - \widehat{\chi}_v\|_2^2 \leq \frac{\mu^2}{40 \log_{\frac{1}{\alpha}}^2 k}$. An example of the input tree FRONTIER is illustrated in Figure 5 with thin solid black edges. Additionally, one can see node v which is a leaf of FRONTIER in this figure.

Algorithm 8 recovers heavy frequencies of signal $\widehat{y}$ that lie in the subree of v, i.e., set $S =$ HEAD $\cap \text{FreqCone}_{\text{FRONTIER}}(v)$, by iteratively exploring the subtree of FRONTIER rooted at v, which we denote by T, and simultaneously updating the proxy signal $\widehat{\chi}_v$. We show an example of subtree T at some iteration of our algorithm in Figure 5 with thick solid edges. The algorithm also maintains a subset of leaves denoted by Marked that contains the leaves of FRONTIER that are fully identified, that is the set of leaves that are at the bottom level and hence there is no ambiguity in their frequency content (there is exactly one element in frequency cone of marked leaves). We show the set of marked leaves in Figure 5 using blue squares. Subtree T, in all iterations of our algorithm, maintains the invariant that the frequency cone of each of its leaves contain at least one head

---

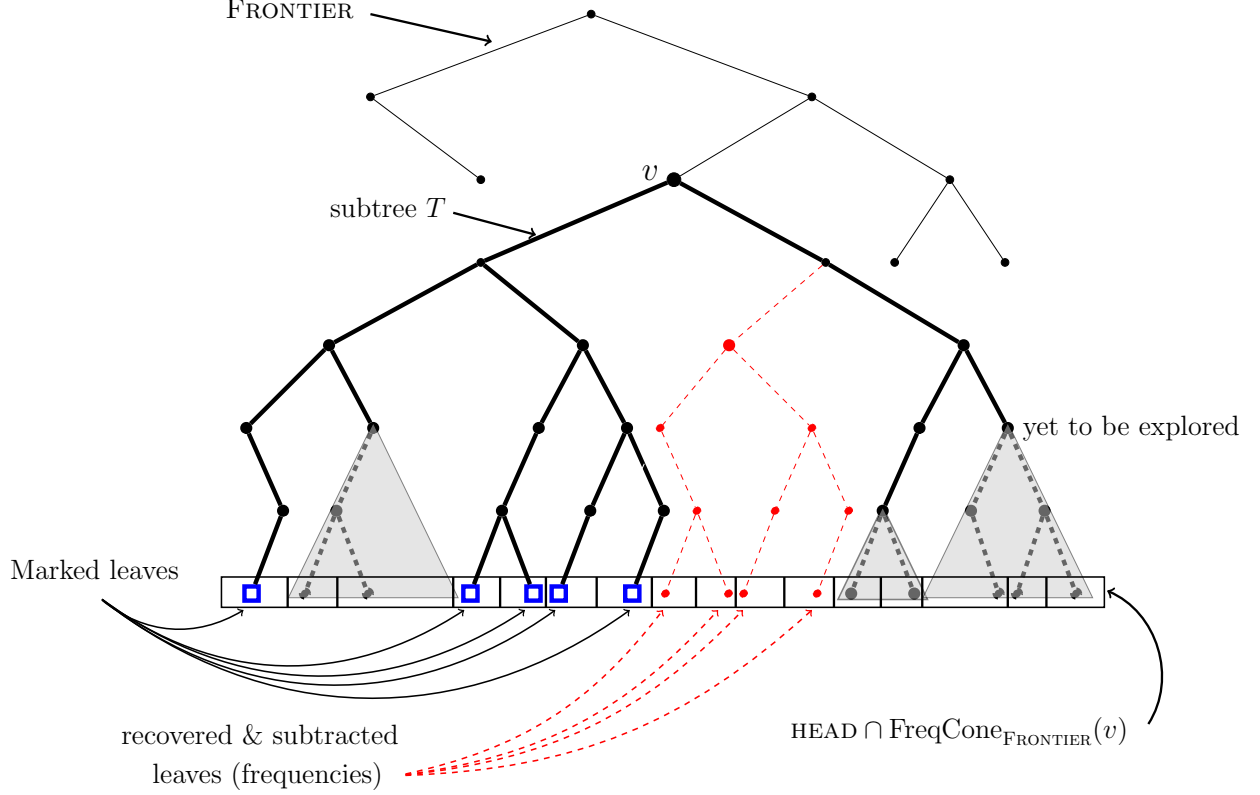[11] proved as Theorem 13 in Section 12

Figure 5: Illustration of an instance of RECURSIVEROBUSTSFT (Algorithm 8). This procedure takes in a tree FRONTIER (shown with thin edges) together with a leaf $v \in$ LEAVES(FRONTIER) and adaptively explores/constructs the subtree T rooted at v to find all heavy frequencies that lie in FreqCone$_{\text{FRONTIER}}(v)$. If HEAD denotes the set of heavy frequencies, then the algorithm finds HEAD$\cap$ FreqCone$_{\text{SIDETREE}}(v)$ by exploring T. Once the identity of a leaf is fully revealed, the algorithm adds that leaf to the set Marked. When the number of marked leaves grows to the point where there exists a subset of marked frequencies that can be estimated cheaply, our algorithm estimates the Cheap subset in a batch, subtracts off the estimated signal, and removes all corresponding leaves from T and Marked.

element and furthermore the frequency cone of each of its *unmarked* leaves contain at least $b + 1$ head element, where $b = \alpha k$, i.e.,

$$| \text{FreqCone}_{\text{FRONTIER} \cup T}(u) \cap \text{HEAD}| \geq \begin{cases} 1 & \text{for every } u \in \text{Marked} \\ b + 1 & \text{for every } u \in \text{LEAVES}(T) \setminus \text{Marked} \end{cases}. \tag{19}$$

We demonstrate, in Figure 5, the leaves that correspond to set $S = $ HEAD $\cap$ FreqCone$_{\text{FRONTIER}}(v)$ via leaves at bottom level of the subtree rooted at v. Assuming that for the example shown in this figure $b = \alpha k = 2$, one can easily verify (19) by noting that the frequency cone of each leaf of T contains at least one element from the set HEAD and frequency cones of *unmarked* leaves contain at least two element of HEAD. Additionally, at every iteration of the algorithm, the union of all frequency cones of subtree T captures all heavy frequencies that are not recovered yet, i.e.,

$$S \setminus \text{supp}(\widehat{\chi}_v) \subseteq \text{supp}(\text{FRONTIER} \cup T). \tag{20}$$

In Figure 5, we show the set of fully recovered leaves (frequencies), i.e., supp$(\widehat{\chi}_v)$, using red thin

dashed subtrees. These frequencies are subtracted from the residual signal $\widehat{y} - \widehat{\chi}_v$ and their corresponding leaves are removed from subtree T, as well. One can verify that condition 20 holds in the example depicted in Figure 5. Moreover, the estimated value of every frequency that is recovered so far, is accurate up to an average error of $\frac{\mu}{\sqrt{40k \cdot \log_{\frac{1}{\alpha}} k}}$. More precisely, in every iteration of the algorithm the following property is maintained,

$$\frac{\sum_{\boldsymbol{f} \in \mathrm{supp}(\widehat{\chi}_v)} |\widehat{y}(\boldsymbol{f}) - \widehat{\chi}_v(\boldsymbol{f})|^2}{|\mathrm{supp}(\widehat{\chi}_v)|} \leq \frac{\mu^2}{40k \cdot \log^2_{\frac{1}{\alpha}} k}. \tag{21}$$

At the begining of the procedure, subtree T is initialized to be the leaf v, i.e., $T = \{v\}$, and will be dynamically changing throughout the execution of our algorithm. Moreover, we initialize $\widehat{\chi}_v \equiv 0$. Trivially, these initial values satisfy (19), (20), and (21).

The algorithm operates by picking the *unmarked* leaf of T that has the smallest weight. Then the algorithm explores the children of this node by recursively running RECURSIVEROBUSTSFT on them with a *reduced budget* to recover the heavy frequencies that lie in their frequency cones. To be more precise, let us call the *unmarked* leaf of T that has the smallest weight z. We denote by z_{left} and z_{right} the left and right children of z. Let us consider exploration of the left child z_{left}, the right child is exactly the same. If the number of heavy frequencies in the frequency cone of z_{left} is bounded by $b = \alpha k$, i.e., $|\mathrm{HEAD} \cap \mathrm{FreqCone}_{\mathrm{FRONTIER} \cup \{z_{\mathrm{left}}, z_{\mathrm{right}}\}}(z_{\mathrm{left}})| \leq b$, then RECURSIVER-OBUSTSFT$(x, \widehat{\chi}_{in} + \widehat{\chi}_v, \mathrm{FRONTIER} \cup T \cup \{z_{\mathrm{left}}, z_{\mathrm{right}}\}, z_{\mathrm{left}}, b, \alpha, \mu)$ recovers every frequency in the set $\mathrm{HEAD} \cap \mathrm{FreqCone}_{\mathrm{FRONTIER} \cup \{z_{\mathrm{left}}, z_{\mathrm{right}}\}}(z_{\mathrm{left}})$ up to an average error of $\frac{\mu}{\sqrt{40b \cdot \log_{\frac{1}{\alpha}} b}}$. Note that this everage estimation error is not sufficient for achieving the invariant (21), hence, instead of directly using the values that the recursive call of RECURSIVEROBUSTSFT recovered to update $\widehat{\chi}_v$ at the newly recovered heavy frequencies, our algorithm adds the leaves corresponding to the recovered set of frequencies, i.e., $\mathrm{HEAD} \cap \mathrm{FreqCone}_{\mathrm{FRONTIER} \cup \{v_{\mathrm{left}}, v_{\mathrm{right}}\}}(v_{\mathrm{left}})$, at the bottom level of T and marks them as fully identified (adds them to Marked). It can be seen in Figure 5 that all marked leaves are at the bottom level of the tree. For achieving maximum efficinecy we employ a new *lazy estimation* scheme, that is, the estimation of values of marked leaves is delayed until there is a large number of marked leaves and thus there exists a subset of them that is cheap to estimate. On the other hand, if the number of head elements in frequency cone of z_{left} is more than b then our algorithm detects this and subsequently adds node z_{left} to T. These operations ensure that the invariants (19), (20), and (21) are maintained.

Once the size of set Marked grows sufficiently such that it contains a subset that is cheap to estimate, our algorithm estimates the values of the cheap frequencies. More precisely, at some point, Marked will contains a non-empty subset Cheap such that the values of all frequencies in Cheap can be estimated cheaply and subsequently, our algorithm esimates those frequencies in a batch up to an average error of $O\left(\frac{\mu}{\sqrt{k} \cdot \log N}\right)$, updates $\widehat{\chi}$ accordingly and removes all estimated (Cheap) leaves from FRONTIER and Marked. This ensures that invariants (19), (20), and (21) are maintained. The estimated leaves are illustrated in Figure 5 using red thin dashed subtrees. We also demontrate the subtrees of T that contain HEAD element and are yet to be explored by our algorithm using gray cones and dashed edges in Figure 5. The gray cone means that there are heavy elements in that frequency cone that need to be identified as that node has not reached the bottom level yet.

Finally, the algorithm keeps tabs on the runtime it spends and ensures that even if the input signal does not satisfy the preconditions for successful recovery, in particular if $|\mathrm{HEAD} \cap \mathrm{FreqCone}_{\mathrm{FRONTIER}}(v)| > k$, the runtime stays bounded. Additionally, the algorithm performs a

quality control by running HEAVYTEST on the residual and if the recovered signal is not correct due to violation of some preconditions, it will be reflected in the output of our algorithm.

Analysis of RecursiveRobustSparseFT. Frirst we analyze the runtime and sample complexity of RECURSIVEROBUSTSPARSEFT in the following lemma.

Lemma 22 (RECURSIVEROBUSTSFT – Time and Sample Complexity). *For every subtree* FRONTIER *of* T_N^{full}, *every leaf* v *of* FRONTIER, *positive integer* k, *every* $\alpha = o\left(\frac{1}{\log N}\right)$ *and* $\mu \geq 0$, *and every signals* $x, \widehat{\chi}_{in} : [n]^d \to \mathbb{C}$, *consider an invocation of primitive* RECURSIVEROBUSTSFT *(Algorithm 8) with inputs* $(x, \widehat{\chi}_{in}, \text{FRONTIER}, v, k, \alpha, \mu)$. *Then,*

- *The running time of primitive is bounded by*

$$\widetilde{O}\left(\left(\frac{k^2}{\alpha} \cdot 2^{w_{\text{FRONTIER}}(v)} + \frac{k}{\alpha} \cdot \|\widehat{\chi}_{in}\|_0\right) \cdot (2\log N)^{\log_{\frac{1}{\alpha}} k} + k^2 \cdot \|\widehat{\chi}_{in}\|_0 + k^3\right).$$

- *The number of accesses it makes on* x *is always bounded by*

$$\widetilde{O}\left(\frac{k^2}{\alpha} \cdot 2^{w_{\text{FRONTIER}}(v)} \cdot (2\log N)^{\log_{\frac{1}{\alpha}} k}\right).$$

Moreover, the output signal $\widehat{\chi}_v$ *always satisfies* $\text{supp}(\widehat{\chi}_v) \subseteq \text{FreqCone}_{\text{FRONTIER}}(v)$ *and* $\|\widehat{\chi}_v\|_0 \leq k$.

Proof. The proof is by induction on parameter k. The **base of induction** corresponds to $k \leq \frac{1}{\alpha}$. For every $k \leq \frac{1}{\alpha}$, Algorithm 8 simply runs PROMISESPARSEFT$(x, \widehat{\chi}_{in}, \text{FRONTIER}, v, k, \lceil \frac{k}{\alpha} \rceil, \mu)$ in line 2. Therefore, by Lemma19, the runtime and sample complexity of our algorithm are bounded by $\widetilde{O}\left(\frac{k}{\alpha} \cdot \|\widehat{\chi}_{in}\|_0 + \frac{k^2}{\alpha} \cdot 2^{w_{\text{FRONTIER}}(v)}\right)$ and $\widetilde{O}\left(\frac{k^2}{\alpha} \cdot 2^{w_{\text{FRONTIER}}(v)}\right)$, respectively. Moreover, by Lemma19, the output signal $\widehat{\chi}_v$ satisfies $\|\widehat{\chi}_v\|_0 \leq k$ as well as $\text{supp}(\widehat{\chi}_v) \subseteq \text{FreqCone}_{\text{FRONTIER}}(v)$. This proves that the inductive hypothesis holds for every integer $k \leq \frac{1}{\alpha}$, hence the base of induction holds.

To prove the **inductive step**, suppose that the lemma holds for every $k \leq m - 1$ for some integer $m \geq \lfloor \frac{1}{\alpha} \rfloor + 1$. Assuming the inductive hypothesis, we prove that the lemma holds for $k = m$. First, we prove that Algorithm 8 terminates after a bounded number of iterations. For the purpose of having a tight analysis of the runtime and sample complexity, we need to have tight upper bounds on the number of times our algorithm invokes primitive ESTIMATE in line 12 as well as the number of times our algorithm recursively calls itself in lines 21 and 22. First, we show that the number of iterations in which the if-staement in line 9 is True, and hence the number of times we invoke ESTIMATE in line 12, is bounded by $O(k)$. The reason is, everytime the if-staement in line 9 becomes True the sparsity of $\widehat{\chi}_v$, i.e., $\|\widehat{\chi}_v\|_0$, increases by $|\text{Cheap}| \geq 1$, because the if-staement in line 9 ensures that preconditions of Claim 6 hold, hence, by invoking this claim, Cheap $\neq \varnothing$. On the other hand, we can see from the way our algorithm operates that the sparity of $\widehat{\chi}_v$ does not decrease in any of the iterations of our algorithm. Therefore, because the if-statement in line 7 of the algorithm makes sure that $\|\widehat{\chi}_v\|_0$ does not exceed k, we conclude that the total number of iterations in which the if-statement in line 9 is True is bounded by $O(k)$. Hence, the number of times our algorithm calls ESTIMATE in line 12 is $O(k)$.

In order to bound the number of iterations of our algorithm in which the if-statement in line 9 is False, we use a potential function. Let $\widehat{\chi}_v^{(t)}$ denote the signal $\widehat{\chi}_v$ at the end of iteration t of the algorithm. Furthermore, let $T^{(t)}$ denote the subtree T at the end of tth iteration. Additionally, let Marked$^{(t)}$ denote the set Marked (defined in Algorithm 8) at the end of iteration t. We prove that

Algorithm 8 A Recursive Robust High-dimensional Sparse FFT Algorithm

---

1: **procedure** RECURSIVEROBUSTSFT$(x, \widehat{\chi}_{in}, \text{FRONTIER}, v, k, \alpha, \mu)$ $\triangleright$ μ: upper bound on tail norm $\|\eta\|_2$

2:    **if** $k \leq \frac{1}{\alpha}$ **then return** PROMISESPARSEFT $\left(x, \widehat{\chi}_{in}, \text{FRONTIER}, v, k, \lceil \frac{k}{\alpha} \rceil, \mu\right)$

3:    Let T denote the subtree of FRONTIER rooted at v – i.e. $T \leftarrow \{v\}$

4:    $\widehat{\chi}_v \leftarrow \{0\}^{n^d}$ $\triangleright$ Sparse vector to approximate $(\widehat{x} - \widehat{\chi}_{in})_{\text{FreqCone}_{\text{FRONTIER}}(v)}$

5:    $b \leftarrow \lceil \alpha k \rceil$, Marked $\leftarrow \varnothing$ $\triangleright$ Marked: set of fully identified leaves (frequencies)

6:    **repeat**

7:      **if** $(b+1) \cdot |\text{LEAVES}(T_v) \setminus \text{Marked}| + |\text{Marked}| + \|\widehat{\chi}_v\|_0 > k$ **then**

8:        **return** $\left(\text{False}, \{0\}^{n^d}\right)$ $\triangleright$ Exit because budget of v is wrong

9:      **if** $\sum_{u \in \text{Marked}} 2^{-w_T(u)} \geq \frac{1}{2}$ **then**

10:        Cheap $\leftarrow$ FINDCHEAPTOESTIMATE (T, Marked)

11:           $\triangleright$ Lazy estimation: We extract from the batch of marked leaves a subset that is cheap to estimate on average

12:        $\left\{\widehat{H}_u\right\}_{u \in \text{Cheap}} \leftarrow$ ESTIMATE $\left(x, \widehat{\chi}_{in} + \widehat{\chi}_v, \text{FRONTIER} \cup T, \text{Cheap}, \frac{736 k \cdot \log^2 N}{|\text{Cheap}|}\right)$

13:        **for** $u \in$ Cheap **do**

14:          $\widehat{\chi}_v(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$

15:          Remove node u from subtree T

16:        Marked $\leftarrow$ Marked $\setminus$ Cheap

17:        **continue**

18:      $z \leftarrow \text{argmin}_{u \in \text{LEAVES}(T) \setminus \text{Marked}} w_T(u)$ $\triangleright$ pick the minimum weight leaf in subtree T which is not in Marked

19:      $z_{\text{left}} :=$ left child of z and $z_{\text{right}} :=$ right child of z

20:      $T' \leftarrow T \cup \{z_{\text{left}}, z_{\text{right}}\}$ $\triangleright$ Explore children of z

21:      $(\text{ISCORR}_{\text{left}}, \widehat{\chi}_{\text{left}}) \leftarrow$ RECURSIVEROBUSTSFT $(x, \widehat{\chi}_{in} + \widehat{\chi}_v, \text{FRONTIER} \cup T', z_{\text{left}}, b, \alpha, \mu)$

22:      $(\text{ISCORR}_{\text{right}}, \widehat{\chi}_{\text{right}}) \leftarrow$ RECURSIVEROBUSTSFT $(x, \widehat{\chi}_{in} + \widehat{\chi}_v, \text{FRONTIER} \cup T', z_{\text{right}}, b, \alpha, \mu)$

23:      **if** $\text{ISCORR}_{\text{left}}$ and $\text{ISCORR}_{\text{right}}$ and $z \neq v$ and $\|\widehat{\chi}_{\text{left}}\|_0 + \|\widehat{\chi}_{\text{right}}\|_0 \leq b$ **then**

24:        **return** $\left(\text{False}, \{0\}^{n^d}\right)$ $\triangleright$ Exit because budget of v is wrong

25:      **if** $\text{ISCORR}_{\text{left}}$ **then**

26:        $\forall \boldsymbol{f} \in \text{supp}(\widehat{\chi}_{\text{left}})$, add the unique leaf corresponding to $\boldsymbol{f}$ to subtree T and Marked

27:      **else**

28:        Add z_{left} to subtree T

29:      **if** $\text{ISCORR}_{\text{right}}$ **then**

30:        $\forall \boldsymbol{f} \in \text{supp}(\widehat{\chi}_{\text{right}})$, add the unique leaf corresponding to $\boldsymbol{f}$ to subtree T and Marked

31:      **else**

32:        Add z_{right} to subtree T

33:    **until** T has no leaves besides v

34:    **if** HEAVYTEST $\left(x, \widehat{\chi}_{in} + \widehat{\chi}_v, \text{FRONTIER}, v, O\left(\frac{k}{\alpha} \log^3 N\right), 6\mu^2\right)$ **then**

35:        $\triangleright$ The number of heavy coordinates in $\text{FreqCone}_{\text{FRONTIER}}(v)$ is more than k

36:      **return** $\left(\text{False}, \{0\}^{n^d}\right)$

37:    **else**

38:      **return** $(\text{True}, \widehat{\chi}_v)$

Algorithm 9 Robust High-dimensional Sparse FFT with $\widetilde{O}(k^3)$ Time and $\widetilde{O}\left(k^{2+o(1)}\right)$ Samples

1: **procedure** ROBUSTSFT(x, k, ϵ, μ)
2: $\quad \alpha \leftarrow 2^{-\sqrt{\log k \cdot \log(2 \log N)}}$
3: $\quad$ (ISCORR, $\widehat{\chi}$) $\leftarrow$ RECURSIVEROBUSTSFT $\left(x, \{0\}^{n^d}, \{\text{root}\}, \text{root}, k, \alpha, \mu\right)$
4: $\quad$ Let T be the splitting tree corresponding to the set supp$(\widehat{\chi})$
5: $\quad \widehat{\chi}_\epsilon \leftarrow \{0\}^{n^d}$
6: $\quad$ **while** tree T has a leaf besides its root **do**
7: $\quad\quad$ Cheap $\leftarrow$ FINDCHEAPTOESTIMATE $(T, \text{LEAVES}(T))$
8: $\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad$ $\triangleright$ The set of frequencies that are cheap to estimate on average
9: $\quad\quad \left\{\widehat{H}_u\right\}_{u \in \text{Cheap}} \leftarrow$ ESTIMATE $\left(x, \widehat{\chi}_\epsilon, T, \text{Cheap}, \frac{32k}{\epsilon \cdot |\text{Cheap}|}\right)$
10: $\quad\quad$ **for** $u \in$ Cheap **do**
11: $\quad\quad\quad \widehat{\chi}_\epsilon(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$
12: $\quad\quad\quad$ Remove node u from tree T
13: $\quad$ **return** $\widehat{\chi}_\epsilon$

---

the number of iterations in which the if-statement in line 9 of our algorithm is False is bounded by $O\left(\frac{\log N}{\alpha}\right)$ using the following potential function, defined for non-negative integer t:

$$\phi_t := (\log N + 1) \cdot |\text{Marked}^{(t)}| + 2 \log N \cdot \|\widehat{\chi}_v^{(t)}\|_0 + b \cdot \sum_{u \in \text{LEAVES}\left(T^{(t)}\right) \setminus \text{Marked}^{(t)}} l_{T^{(t)}}(u).$$

We prove that assuming the algorithm does not terminate in q iterations, for some integer q, then in every positive iteration $t \leq q$, if the if-statement in line 9 of Algorithm 8 is False, then the above potential function increases by at least b, i.e., $\phi_t \geq \phi_{t-1} + b$. Additionally, when the if-statement in line 9 is True, the potential increases by at least $\log N - 1$, i.e., $\phi_t \geq \phi_{t-1} + \log N - 1$. We show that at any given iteration t of the algorithm the potential function ϕ_t increases in the abovementioned fashion.

Case 1 – the if-statement in line 9 of Algorithm 8 is True. In this case, we have that $\sum_{u \in \text{Marked}^{(t-1)}} 2^{-w_{T^{(t-1)}}(u)} \geq \frac{1}{2}$. As a result, by Claim 6, the set Cheap$^{(t)} \subseteq$ Marked$^{(t-1)}$ that the algorithm computes in line 10 by running the primitive FINDCHEAPTOESTIMATE is non-empty. Then, the algorithm constructs $T^{(t)}$ by removing all leaves that are in the set Cheap$^{(t)}$ from tree $T^{(t-1)}$ and leaving the rest of the tree unchanged. Furthermore, the algorithm updates the set Marked$^{(t)}$ by subtracting Cheap$^{(t)}$ from Marked$^{(t-1)}$. Additionally, in this case, the algorithm computes $\{\widehat{H}_u\}_{u \in \text{Cheap}^{(t)}}$ by running the procedure ESTIMATE in line 12 and then updates $\widehat{\chi}_v^{(t)}(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$ for every $u \in$ Cheap$^{(t)}$ and $\widehat{\chi}_v^{(t)}(\boldsymbol{\xi}) = \widehat{\chi}_v^{(t-1)}(\boldsymbol{\xi})$ at every other frequency $\boldsymbol{\xi}$. Therefore, $\|\widehat{\chi}_v^{(t)}\|_0 = \|\widehat{\chi}_v^{(t)}\|_0 + |\text{Cheap}^{(t)}|$. Thus,

$$\phi_t - \phi_{t-1} = (\log N - 1) \cdot |\text{Cheap}^{(t)}| \geq \log N - 1,$$

where the inequality follows from Cheap$^{(t)} \neq \varnothing$. This proves the potential increase that we wanted.

Case 2 – the if-statement in line 9 is False. In this case, either the algorithm terminates by the if-statement in line 23, which contradicts with our assumption that the algorithm does not

terminate after $q \geq t$ iterations, or the following holds,

$$|\text{Marked}^{(t)}| + b \cdot \sum_{u \in \text{LEAVES}\left(T^{(t)}\right) \setminus \text{Marked}^{(t)}} l_{T^{(t)}}(u)$$

$$\geq |\text{Marked}^{(t-1)}| + b \cdot \sum_{u \in \text{LEAVES}\left(T^{(t-1)}\right) \setminus \text{Marked}^{(t-1)}} l_{T^{(t-1)}}(u) + b,$$

while $|\text{Marked}^{(t)}| \geq |\text{Marked}^{(t-1)}|$ and $\|\widehat{\chi}_v^{(t)}\|_0 = \|\widehat{\chi}_v^{(t-1)}\|_0$. Thus, in this case, $\phi_{t+1} - \phi_t \geq b$ which is the potential increase that we wanted to prove.

So far, we proved that ϕ_t must increase by at least $\log N - 1$ at every iteration of the algorithm. Moreover, at every iteration of the algorithm where the if-statement in line 9 is False the potential increases by at least b. Also, the potential function ϕ_t is non-negative for every t. On the other hand, the if-statement in line 7 ensures that at any iteration $t \leq q$ it must hold that $\phi_t \leq 2k \log N$. Therefore, the potential increse that we proved implies that Algorithm 8 must terminate after at most $q = 2k \log N$ iterations, where only in $\frac{2 \log N}{\alpha}$ of the iterations the if-statement in line 9 can be False. Therefore, the total number of times our algorithm recursively invokes itself in lines 21 and 22 is bounded by $\frac{2 \log N}{\alpha}$.

Now that we have the termination quarantee, we can use the fact that our algorithm constructs $\widehat{\chi}_v$ by exclusively estimating the values of frequencies that lie in $\text{FreqCone}_{\text{FRONTIER}}(v)$ in line 12, one can see that the output signal $\widehat{\chi}_v$ always satisfies $\text{supp}(\widehat{\chi}_v) \subseteq \text{FreqCone}_{\text{FRONTIER}}(v)$. Additionally, the if-staement in line 7, ensures that $\|\widehat{\chi}_v\|_0 \leq k$. Now we bound the running time and sample complexity of the algorithm.

Sample Complexity and Runtime: The expensive components of the algorithm are primitive ESTIMATE in line 12, the recursive call of RECURSIVEROBUSTSFT in lines 21 and 22, and invocation of HEAVYTEST in line 34 of the algorithm.

We first bound the time and sample complexity of invoking ESTIMATE in line 12. We remark that, at any iteration t, the algorithm runs primitive ESTIMATE only if **case 1** that we mentioned earlier in the proof happens. Therefore, by Claim 6, the set $\varnothing \neq \text{Cheap}^{(t)} \subseteq \text{Marked}^{(t-1)}$ that our algorithm computes in line 10 by running the primitive FINDCHEAPTOESTIMATE satisfies the property that $|\text{Cheap}^{(t)}| \cdot \left(8 + 4 \log |\text{Marked}^{(t-1)}|\right) \geq \max_{u \in \text{Cheap}^{(t)}} 2^{w_{T^{(t-1)}}(u)}$. By the if-statement in line 7 of the algorithm, this implies that $|\text{Cheap}^{(t)}| \cdot (8 + 4 \log k) \geq \max_{u \in \text{Cheap}^{(t)}} 2^{w_{T^{(t-1)}}(u)}$. Thus, by Lemma 18, the time and sample complexity of every invocation of ESTIMATE in line 12 of our algorithm are bounded by

$$\widetilde{O}\left(\frac{k}{|\text{Cheap}^{(t)}|} \sum_{u \in \text{Cheap}^{(t)}} 2^{w_{\text{FRONTIER} \cup T^{(t-1)}}(u)} + k \cdot \left\|\widehat{\chi}_v^{(t-1)} + \widehat{\chi}_{in}\right\|_0\right)$$

and $\widetilde{O}\left(\frac{k}{|\text{Cheap}^{(t)}|} \sum_{u \in \text{Cheap}^{(t)}} 2^{w_{\text{FRONTIER} \cup T^{(t-1)}}(u)}\right)$, respectively. Using the fact that $\|\widehat{\chi}_v^{(t-1)}\|_0 \leq k$, these time and sample complexities are further upper bounded by

$$\widetilde{O}\left(k \cdot \left(2^{w_{\text{FRONTIER}}(v)} \cdot |\text{Cheap}^{(t)}| + \|\widehat{\chi}_{in}\|_0\right) + k^2\right)$$

and $\widetilde{O}\left(k \cdot 2^{w_{\text{FRONTIER}}(v)} \cdot |\text{Cheap}^{(t)}|\right)$, respectively. We proved that the total number of times we run ESTIMATE in line 12 of the algorithm, is bounded by $O(k)$. Using this together with the fact that

$\sum_{t:\text{ if-statement in line 9 is True}} \left|\text{Cheap}^{(t)}\right| = \|\widehat{\chi}_v\|_0 \leq k$, the total runtime and sample complexity of all invocations of ESTIMATE in all iterations can be upper bounded by $\widetilde{O}\left(k^3 + k^2(\|\widehat{\chi}_{in}\|_0 + 2^{w_{\text{FRONTIER}}(v)})\right)$ and $\widetilde{O}\left(k^2 \cdot 2^{w_{\text{FRONTIER}}(v)}\right)$, respectively.

Now we bound the runtime and sample complexity of invoking RECURSIVEROBUSTSFT in lines 21 and 22 of the algorithm. Note that at any iteration t, our algorithm recursively calls RECURSIVEROBUSTSFT only if **case 2** that we mentioned earlier in the proof occurs. As we showed, the total number of times that this happens is bounded by $\frac{2\log N}{\alpha}$. Since, in line 18 of the algorithm, we pick leaf z with the smallest weight, and since the number of leaves of subtree $T^{(t-1)}$ that are not in the set Marked$^{(t-1)}$ are bounded by $\frac{k}{b+1}$ (ensured by the if-statement in line 7), we have $w_{\text{FRONTIER} \cup T'}(z_{\text{left}}) = w_{\text{FRONTIER} \cup T'}(z_{\text{right}}) \leq w_{\text{FRONTIER}}(v) + \log\frac{k}{b+1} + 1$. Also note that $\|\widehat{\chi}_v^{(t-1)}\|_0 \leq k$, ensured by the if-statement in line 7. Therefore, by the inductive hypothesis, the time and sample complexities of each recursive invocation of RECURSIVEROBUSTSFT by our algorithm are bounded by

$$\widetilde{O}\left(\left(\frac{b^2 \cdot 2^{w_{\text{FRONTIER}}(v)}}{\alpha^2} + \frac{b}{\alpha} \cdot \|\widehat{\chi}_{in}\|_0\right) \cdot (2\log N)^{\log\frac{1}{\alpha} b} + b^2 \cdot \|\widehat{\chi}_{in}\|_0 + kb^2\right)$$

and $\widetilde{O}\left(\frac{b^2}{\alpha^2} \cdot 2^{w_{\text{FRONTIER}}(v)} \cdot (2\log N)^{\log\frac{1}{\alpha} b}\right)$. We proved that the total number of iterations in which **case 2** happens, and hence the number of times we run RECURSIVEROBUSTSFT in lines 21 and 22 of the algorithm, is bounded by $\frac{2\log N}{\alpha}$. Therefore, the total time and sample complexity of all invocations of PROMISESPARSEFT in lines 21 and 22 are bounded by

$$\widetilde{O}\left(\left(\frac{k^2}{\alpha} \cdot 2^{w_{\text{FRONTIER}}(v)} + \frac{k}{\alpha} \cdot \|\widehat{\chi}_{in}\|_0\right) \cdot (2\log N)^{\log\frac{1}{\alpha} k} + \alpha k^2 \cdot \|\widehat{\chi}_{in}\|_0 + \alpha k^3\right)$$

and $\widetilde{O}\left(\frac{k^2}{\alpha} \cdot 2^{w_{\text{FRONTIER}}(v)} \cdot (2\log N)^{\log\frac{1}{\alpha} k}\right)$, respectively.

Finally, we bound the time and sample complexity of invoking HEAVYTEST in line 34 of our algorithm. Since $\|\widehat{\chi}_v\|_0 \leq k$, by Lemma 17, the time and sample complexity of the HEAVYTEST in line 34 are bounded by $\widetilde{O}\left(\|\widehat{\chi}_{in}\|_0 \cdot \frac{k}{\alpha} + \frac{k^2}{\alpha} + 2^{w_{\text{FRONTIER}}(v)} \cdot \frac{k}{\alpha}\right)$ and $\widetilde{O}\left(2^{w_{\text{FRONTIER}}(v)} \cdot \frac{k}{\alpha}\right)$, respectively. Hence, we find that the total time and sample complexity of our algorithm are bounded by

$$\widetilde{O}\left(\left(\frac{k^2 \cdot 2^{w_{\text{FRONTIER}}(v)}}{\alpha} + \frac{k}{\alpha} \cdot \|\widehat{\chi}_{in}\|_0\right) \cdot (2\log N)^{\log\frac{1}{\alpha} k} + k^2 \cdot \|\widehat{\chi}_{in}\|_0 + k^3\right)$$

and $\widetilde{O}\left(\frac{k^2}{\alpha} \cdot 2^{w_{\text{FRONTIER}}(v)} \cdot (2\log N)^{\log\frac{1}{\alpha} k}\right)$, respectively. This proves the inductive step of the proof and consequently completes the proof of our lemma. $\square$

Now we are in a position to present the main invariant of primitive RECURSIVEROBUSTSFT.

Lemma 23 (RECURSIVEROBUSTSFT - Invariants). *Consider the preconditions of Lemma 22. Let $\widehat{y} := \widehat{x} - \widehat{\chi}_{in}$ and $S := \text{FreqCone}_T(v) \cap \text{HEAD}_\mu(\widehat{y})$, where $\text{HEAD}_\mu(\cdot)$ is defined as per (7). If i) $\text{HEAD}_\mu(\widehat{y}) \subseteq \text{supp}(\text{FRONTIER})$, ii) $\|\widehat{y} - \widehat{y}_{\text{HEAD}_\mu(\widehat{y})}\|_2^2 \leq \frac{21\mu^2}{20} + \frac{\mu^2}{20\log\frac{1}{\alpha}(k/\alpha)}$, and iii) $|S| \leq \frac{k}{\alpha}$, then with probability at least $1 - O\left(\left(\frac{2\log N}{\alpha}\right)^{\log\frac{1}{\alpha} k} \cdot N^{-4}\right)$, the output (Budget, $\widehat{\chi}_v$) of Algorithm 8 satisfies the following,*

1. *If $|S| \leq k$ then $\text{Budget} = \text{True}$, $\text{supp}(\widehat{\chi}_v) \subseteq S$, and $\|\widehat{y}_S - \widehat{\chi}_v\|_2^2 \leq \frac{\mu^2}{40\log_{1/\alpha}^2 k}$;*

2. *If $|S| > k$ then* Budget = False *and* $\widehat{\chi}_v \equiv \{0\}^{n^d}$.

Proof. The proof is by induction on parameter k. The **base of induction** corresponds to $k \leq \frac{1}{\alpha}$. For every $k \leq \frac{1}{\alpha}$, Algorithm 8 simply runs PROMISESPARSEFT$\left(x, \widehat{\chi}_{in}, \text{FRONTIER}, v, k, \lceil \frac{k}{\alpha} \rceil, \mu\right)$ in line 2. Therefore, by Lemma20, the claims of the lemma hold with probability at least $1 - \frac{1}{N^4}$. This proves that the inductive hypothesis holds for every integer $k \leq \frac{1}{\alpha}$, hence the base of induction holds.

To prove the **inductive step**, suppose that the lemma holds for every $k \leq m - 1$ for some integer $m \geq \lfloor \frac{1}{\alpha} \rfloor + 1$. Assuming the inductive hypothesis, we prove that the lemma holds for $k = m$. To prove the inductive claim, we first analyze the algorithm under the assumption that the primitives HEAVYTEST and ESTIMATE are replaced with more powerful primitives that succeeds deterministically. Hence, we assume that HEAVYTEST correctly tests the "heavy" hypothesis on its input signal with probability 1 and also ESTIMATE achieves the estimation guarantee of Lemma 18 deterministrically. Moreover, we assume that our inductive invocation of RECURSIVEROBUSTSFT in lines 21 and 22 of the algorithm succeed deterministically, hence, we assume that the inductive hypothesis (the lemma) holds with probability 1. With these assumptions in place, we prove that the lemma holds deterministically (with probability 1). We then establish a coupling between this idealized execution and the actual execution of our algorithm, leading to our result.

We prove the first statement of lemma by (another) induction on the *Repeat-Until loop* of the algorithm. Note that we are proving the inductive step of an inductive proof using another induction (two nested inductions). The first (outer) induction was on the integer k and the second (inner) induction is on the iteration number t of the *Repeat-Until loop* of our algorithm. Let $\widehat{\chi}_v^{(t)}$ denote the signal $\widehat{\chi}_v$ at the end of iteration t of the algorithm. Furthermore, let FRONTIER$^{(t)}$ denote the subtree T at the end of tth iteration. Also, let Marked$^{(t)}$ denote the set Marked (defined in Algorithm 8) at the end of iteration t. Additionaly, for every leaf u of subtree $T^{(t)}$, let $L_u^{(t)}$ denote the "unestimated" frequencies in support of $\widehat{y}$ that lie in frequency cone of u, i.e., $L_u^{(t)} :=$ FreqCone$_{\text{FRONTIER} \cup T^{(t)}}(u) \cap \text{HEAD}_\mu(\widehat{y})$ We prove that if preconditions i, ii and iii together with the presondition of statement 1 (that is $|S| \leq k$), hold, then at every iteration $t = 0, 1, 2, \ldots$ of Algorithm 8, the following properties are maintained,

$P_1(t)$ $S \setminus \text{supp}\left(\widehat{\chi}_v^{(t)}\right) \subseteq \text{supp}\left(T^{(t)}\right) := \bigcup_{u \in \text{LEAVES}(T^{(t)})} \text{FreqCone}_{\text{FRONTIER} \cup T^{(t)}}(u)$;

$P_2(t)$ For every leaf $u \neq v$ of subtree $T^{(t)}$, $\left| L_u^{(t)} \right| \geq 1$. Additionally, if $u \notin \text{Marked}^{(t)}$, then $\left| L_u^{(t)} \right| > b$;

$P_3(t)$ $\left\| \widehat{y}_{S^{(t)}} - \widehat{\chi}_v^{(t)} \right\|_2^2 \leq \frac{|S^{(t)}|}{40 k \cdot \log_{1/\alpha}^2 k} \cdot \mu^2$, where $S^{(t)} := \text{supp}\left(\widehat{\chi}_v^{(t)}\right)$;

$P_4(t)$ $S^{(t)} \subseteq S$ and $S^{(t)} \cap \left(\bigcup_{\substack{u \in \text{LEAVES}(T^{(t)}) \\ u \neq v}} \text{FreqCone}_{\text{FRONTIER} \cup T^{(t)}}(u) \right) = \varnothing$;

The **base of induction** corresponds to the zeroth iteration ($t = 0$), at which point $T^{(0)}$ is a subtree that solely consists of node v and has no other leaves. Moreover, $\widehat{\chi}_v^{(0)} \equiv 0$. Thus, statement $P_1(0)$ trivially holds by definition of set S. The statement $P_2(0)$ holds since there exists no leaf $u \neq v$ in $T^{(0)}$. The statements $P_3(0)$ and $P_4(0)$ hold because of the fact $\widehat{\chi}_v^{(0)} \equiv 0$.

We now prove the **inductive step** by assuming that the inductive hypothesis, $P(t - 1)$ is satisfied for some iteration $t-1$ of Algorithm 8, and then proving that $P(t)$ holds. First, we remark that if inductive hypotheses $P_2(t-1)$ and $P_4(t-1)$ hold true, then by the precondition of statement 1 of the lemma (that is $|S| \leq k$) the if-statement in line 7 of Algorithm 8 is False and hence lines 7

and 8 of the algorithm can be ignored in our analysis. We proceed to prove the induction by considering the two cases that can happen in every iteration t of the algorithm:

Case 1 – the if-statement in line 9 of Algorithm 8 is True. In this case, we have that $\sum_{u \in \mathrm{Marked}^{(t-1)}} 2^{-w_{T^{(t-1)}}(u)} \geq \frac{1}{2}$. As a result, by Claim 6, the set $\mathrm{Cheap} \subseteq \mathrm{Marked}^{(t-1)}$ that the algorithm computes in line 10 by running the primitive FINDCHEAPTOESTIMATE satisfies the property that $|\mathrm{Cheap}| \cdot \left(8 + 4 \log |\mathrm{Marked}^{(t-1)}|\right) \geq \max_{u \in \mathrm{Cheap}} 2^{w_{T^{(t-1)}}(u)}$. Clearly $\mathrm{Cheap} \neq \varnothing$, by Claim 6. Then the algorithm computes $\{\widehat{H}_u\}_{u \in \mathrm{Cheap}}$ by running the procedure ESTIMATE in line 12 and then updates $\widehat{\chi}^{(t)}(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$ for every $u \in \mathrm{Cheap}$ and $\widehat{\chi}^{(t)}(\boldsymbol{\xi}) = \widehat{\chi}^{(t-1)}(\boldsymbol{\xi})$ at every other frequency $\boldsymbol{\xi}$. Therefore, if we let $L := \{\boldsymbol{f}_u : u \in \mathrm{Cheap}\}$, then $S^{(t)} \setminus S^{(t-1)} = L$, by inductive hypothesis $P_4(t-1)$. By $P_3(t-1)$ along with Lemma 18 (its deterministic version that succeeds with probability 1), we find that

$$
\begin{aligned}
\left\|\widehat{\chi}_v^{(t)} - \widehat{y}_{S^{(t)}}\right\|_2^2 &= \left\|(\widehat{\chi}_v^{(t)} - \widehat{y})_{S^{(t-1)}}\right\|_2^2 + \left\|(\widehat{\chi}_v^{(t)} - \widehat{y})_{S^{(t)} \setminus S^{(t-1)}}\right\|_2^2 \\
&= \left\|\widehat{\chi}_v^{(t-1)} - \widehat{y}_{S^{(t-1)}}\right\|_2^2 + \left\|(\widehat{\chi}_v^{(t)} - \widehat{y})_L\right\|_2^2 \\
&\leq \frac{\left|S^{(t-1)}\right| \cdot \mu^2}{40k \log_{1/\alpha}^2 k} + \frac{|L|}{46k \log^2 N} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}\left(\mathrm{FRONTIER} \cup T^{(t-1)}\right)} \left|\left(\widehat{y} - \widehat{\chi}_v^{(t-1)}\right)(\boldsymbol{\xi})\right|^2. \quad (22)
\end{aligned}
$$

Now we bound the second term above,

$$
\sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}\left(\mathrm{FRONTIER} \cup T^{(t-1)}\right)} \left|\left(\widehat{y} - \widehat{\chi}_v^{(t-1)}\right)(\boldsymbol{\xi})\right|^2
$$

$$
= \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(\mathrm{FRONTIER})} |\widehat{y}(\boldsymbol{\xi})|^2 + \sum_{\boldsymbol{\xi} \in \mathrm{FreqCone}_{\mathrm{FRONTIER}}(v) \setminus \mathrm{supp}\left(T^{(t-1)}\right)} \left|\left(\widehat{y} - \widehat{\chi}_v^{(t-1)}\right)(\boldsymbol{\xi})\right|^2
$$

$$
= \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{supp}(\mathrm{FRONTIER})} |\widehat{y}(\boldsymbol{\xi})|^2
$$

$$
+ \sum_{\boldsymbol{\xi} \in \mathrm{FreqCone}_{\mathrm{FRONTIER}}(v) \setminus \left(\mathrm{supp}\left(T^{(t-1)}\right) \cup S^{(t-1)}\right)} |\widehat{y}(\boldsymbol{\xi})|^2 + \left\|\widehat{y}_{S^{(t-1)}} - \widehat{\chi}_v^{(t-1)}\right\|_2^2
$$

$$
= \sum_{\boldsymbol{\xi} \in [n]^d \setminus \left(\mathrm{supp}\left(\mathrm{FRONTIER} \cup T^{(t-1)}\right) \cup S^{(t-1)}\right)} |\widehat{y}(\boldsymbol{\xi})|^2 + \left\|\widehat{y}_{S^{(t-1)}} - \widehat{\chi}_v^{(t-1)}\right\|_2^2
$$

$$
\leq \sum_{\boldsymbol{\xi} \in [n]^d \setminus \mathrm{HEAD}_\mu(\widehat{y})} |\widehat{y}(\boldsymbol{\xi})|^2 + \left\|\widehat{y}_{S^{(t-1)}} - \widehat{\chi}_v^{(t-1)}\right\|_2^2 \qquad \text{(by } P_1(t-1)\text{, precondition i and definition of } S)
$$

$$
\leq \frac{21\mu^2}{20} + \frac{\mu^2}{20 \log_{\frac{1}{\alpha}}(k/\alpha)} + \frac{\mu^2}{40 \log_{\frac{1}{\alpha}}^2 k} \qquad \text{(by } P_3(t-1) \text{ and } P_4(t-1) \text{ and precondition } |S| \leq b)
$$

$$
\leq \frac{23\mu^2}{20}.
$$

Therefore, by plugging the above bound back to (22) we find that,

$$
\left\|\widehat{\chi}_v^{(t)} - \widehat{y}_{S^{(t)}}\right\|_2^2 \leq \frac{\left|S^{(t-1)}\right|}{40k \log_{\frac{1}{\alpha}}^2 k} \cdot \mu^2 + \frac{|L|}{46k \log^2 N} \cdot \left(\frac{23}{20}\mu^2\right) \leq \frac{\left|S^{(t)}\right|}{40k \log_{\frac{1}{\alpha}}^2 k} \cdot \mu^2,
$$

which proves the inductive claim $P_3(t)$.

Moreover, in this case, the algorithm constructs $T^{(t)}$ by removing all leaves that are in the set Cheap from tree $T^{(t-1)}$ and leaving the rest of the tree unchanged. Furthermore, the algorithm updates the set Marked$^{(t)}$ by subtracting Cheap from Marked$^{(t-1)}$. Note that, $P_2(t-1)$ implies that $L \subseteq S$. Thus, the fact $S^{(t)} = S^{(t-1)} \cup L$ together with inductive hypothesis $P_4(t-1)$ as well as the construction of $T^{(t)}$, imply $P_4(t)$. The construction of $T^{(t)}$ together with the fact that $|\text{FreqCone}_{\text{FRONTIER} \cup T^{(t-1)}}(u)| = 1$ for every $u \in \text{Marked}^{(t-1)}$ give $P_1(t)$ and $P_2(t)$.

Case 2 – the if-statement in line 9 is False. Let $z \in \text{LEAVES}\left(T^{(t-1)}\right) \setminus \text{Marked}^{(t-1)}$ be the smallest weight leaf chosen by the algorithm in line 18. In this case, the algorithm constructs tree T' by adding leaves z_{right} and z_{left} to tree $T^{(t-1)}$ as right and left children of z in line 20. Then, the algorithm runs RECURSIVEROBUSTSFT with inputs $\left(x, \widehat{\chi}_{in} + \widehat{\chi}_v^{(t-1)}, T', z_{\text{left}}, b, \alpha, \mu\right)$ and $\left(x, \widehat{\chi}_{in} + \widehat{\chi}_v^{(t-1)}, T', z_{\text{right}}, b, \alpha, \mu\right)$ in lines 21 and 22 respectively. Now we analyze the output of the recursive invocation of RECURSIVEROBUSTSFT in lines 21 and 22. In the following we focus on analyzing $(\text{ISCORR}_{\text{left}}, \widehat{\chi}_{\text{left}})$ but $(\text{ISCORR}_{\text{right}}, \widehat{\chi}_{\text{right}})$ can be analyzed exactly the same way. There are two possibilities that can happen:

Possibility 1) $|\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})| \leq b$. In this case, the inductive hypothesis $P_4(t-1)$ implies that $|S^{(t-1)}| \leq k$ and hence inductive hypothesis $P_3(t-1)$ gives

$$\left\|\widehat{y}_{S^{(t-1)}} - \widehat{\chi}_v^{(t-1)}\right\|_2^2 \leq \frac{\mu^2}{40 \log_{1/\alpha}^2 k}, \tag{23}$$

hence, $\text{HEAD}_\mu\left(\widehat{y} - \widehat{\chi}_v^{(t-1)}\right) = \text{HEAD}_\mu(\widehat{y}) \setminus S^{(t-1)}$. Consequently, if we let $\widehat{g} := \widehat{y} - \widehat{\chi}_v^{(t-1)}$, then: i) $\text{HEAD}_\mu(\widehat{g}) \subseteq \text{supp}(\text{FRONTIER} \cup T')$, by (23) along with $P_1(t-1)$, ii) $\|\widehat{g} - \widehat{g}_{\text{HEAD}_\mu(\widehat{g})}\|_2^2 \leq \frac{21\mu^2}{20} + \frac{\mu^2}{20 \log_{\frac{1}{\alpha}}(b/\alpha)}$, by precondition of the lemma along with (23), and iii)

$$|\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{g})| \leq b,$$

by assumption $|\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})| \leq b$. Therefore, all preconditions of the first statement of Lemma 23 hold. Since we invoke primitive RECURSIVEROBUSTSFT with sparsity $b \leq m - 1$, by our inducive hypothesis that Lemma 23 holds for any sparsity parameter $k \leq m - 1$, we can invoke this lemma (a deterministic version of it that succeeds with probability 1) and conclude that, $\text{ISCORR}_{\text{left}} = \text{True}$, and $\text{supp}(\widehat{\chi}_{\text{left}}) \subseteq \text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{g})$, and $\|\widehat{g}_{\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{g})} - \widehat{\chi}_{\text{left}}\|_2^2 \leq \frac{\mu^2}{40 \log_{1/\alpha}^2 b} \leq \frac{\mu^2}{10}$. This together with inductive hypothesis $P_4(t-1)$ imply that, $\text{supp}(\widehat{\chi}_{\text{left}}) = \text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})$.

So, if $|\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})| \leq b$, then the algorithm adds all leaves that correspond to frequencies in $\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})$ to tree $T^{(t-1)}$ as well as set Marked$^{(t-1)}$. By a similar argument, if $|\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{right}}) \cap \text{HEAD}_\mu(\widehat{y})| \leq b$, then the algorithm adds all leaves corresponding to frequencies in $\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{right}}) \cap \text{HEAD}_\mu(\widehat{y})$ to tree $T^{(t-1)}$ and set Marked$^{(t-1)}$.

Possibility 2) $|\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})| > b$. Same as in **possibility 1**, the inductive hypothesis $P_4(t-1)$ implies that $|S^{(t-1)}| \leq k$, hence, inductive hypothesis $P_3(t-1)$ gives (23). Hence, $\text{HEAD}_\mu\left(\widehat{y} - \widehat{\chi}_v^{(t-1)}\right) = \text{HEAD}_\mu(\widehat{y}) \setminus S^{(t-1)}$. Consequently, if we let $\widehat{g} := \widehat{y} - \widehat{\chi}_v^{(t-1)}$, then we find that i) $\text{HEAD}_\mu(\widehat{g}) \subseteq \text{supp}(\text{FRONTIER} \cup T')$, by $P_1(t-1)$, ii) $\|\widehat{g} - \widehat{g}_{\text{HEAD}_\mu(\widehat{g})}\|_2^2 \leq \frac{21\mu^2}{20} + \frac{\mu^2}{20 \log_{\frac{1}{\alpha}}(b/\alpha)}$, by precondition of the lemma along with (23), and iii)

$$|\text{FreqCone}_{\text{FRONTIER} \cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{g})| \leq |S| \leq k,$$

77

by precondition of statement 1 of the lemma. Additionally, by $P_4(t-1)$, we find that

$$|\text{FreqCone}_{\text{FRONTIER}\cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{g})| = |\text{FreqCone}_{\text{FRONTIER}\cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})| > b.$$

Since we invoke primitive RECURSIVEROBUSTSFT with sparsity $b \leq m - 1$, by our inducive hypothesis that Lemma 23 holds for any sparsity parameter $k \leq m - 1$, we can invoke this lemma (a deterministic version of it that succeeds with probability 1) and conclude that, $\text{IsCORR}_{\text{left}} = \text{False}$, and $\widehat{\chi}_{\text{left}} \equiv 0$.

We remark that since

$$|\text{FreqCone}_{\text{FRONTIER}\cup T'}(z_{\text{left}}) \cap \text{HEAD}_\mu(\widehat{y})| + |\text{FreqCone}_{\text{FRONTIER}\cup T'}(z_{\text{right}}) \cap \text{HEAD}_\mu(\widehat{y})| = \left| L_z^{(t-1)} \right|,$$

the inductive hypothesis $P_2(t-1)$ along with the above arguments imply that the if-statement in line 23 of our algorithm cannot be True and hence in the rest of our analysis we can ignore lines 23 and 24 of the algorithm. Furthermore, in this case the algorithm adds leaf z_{left} as the left child of v to tree $T^{(t-1)}$. By a similar argument, if $|\text{FreqCone}_{\text{FRONTIER}\cup T'}(z_{\text{right}}) \cap \text{HEAD}_\mu(\widehat{y})| > b$, then the algorithm adds leaf z_{right} as the left child of v to tree $T^{(t-1)}$.

Based on the above arguments, according to the values of $\text{IsCORR}_{\text{left}}$ and $\text{IsCORR}_{\text{right}}$, there are various cases that can happen. From the way tree $T^{(t)}$ and set $\text{Marked}^{(t)}$ are obtained from $T^{(t-1)}$ and $\text{Marked}^{(t-1)}$, it follows that in any case all 4 properties of $P(t)$ are maintained. We have proved that for every t, if the inductive hypothesis $P(t-1)$ is satisfied then the property $P(t)$ is maintained. This completess the induction (i.e., the inner induction, recall that we have nested inductions) and proves that properties $P(t)$ is maintained throughout the execution of Algorithm 8, assuming that preconditions i, ii, and iii of the lemma along with the precondition $|S| \leq k$ of statement 1 of the lemma hold.

Lemma 22 proves that Algorithm 8 must terminate after some q iterations. When the algorithm terminates, the condition of the *Repeat-Until* loop in line 33 of the algorithm must be True. Thus, when the algorithm terminates, at q^{th} iteration, there is no leaf in subtree $T^{(q)}$ besides v and as a consequence the set $\text{Marked}^{(q)}$ must be empty. This, together with $P_1(q)$ imply that the signal $\widehat{\chi}_v^{(q)}$ satisfies,

$$\text{supp}\left(\widehat{\chi}_v^{(q)}\right) = S = \text{FreqCone}_{\text{FRONTIER}}(v) \cap \text{HEAD}_\mu(\widehat{y}).$$

Moreover, $P_3(q)$ together with precondition $|S| \leq k$ imply that

$$\left\| \widehat{y}_S - \widehat{\chi}_v^{(q)} \right\|_2^2 \leq \frac{|S|}{40k \log_{1/\alpha}^2 k} \cdot \mu^2 \leq \frac{\mu^2}{40 \log_{1/\alpha}^2 k}.$$

Now we analyze the if-statement in line 34 of the algorithm. The above equalities and inequalities on $\widehat{\chi}_v^{(q)}$ imply that,

$$\left\| \left(\widehat{y} - \widehat{\chi}_v^{(q)}\right)_{\text{FreqCone}_{\text{FRONTIER}}(v)} \right\|_2^2 = \left\| \widehat{y}_{\text{FreqCone}_{\text{FRONTIER}}(v)\setminus S} \right\|_2^2 + \left\| \left(\widehat{y} - \widehat{\chi}_v^{(q)}\right)_S \right\|_2^2$$

$$\leq \left\| \widehat{y}_{\text{FreqCone}_{\text{FRONTIER}}(v)\setminus \text{HEAD}_\mu(\widehat{y})} \right\|_2^2 + \frac{\mu^2}{40}.$$

Therefore, if $\widehat{G}_v$ is a Fourier domain $(v, \text{FRONTIER})$-isolating filter constructed in Lemma 9, then

by Corollary 1 along with the above inequality, we have

$$\left\|\left(\widehat{y}-\widehat{\chi}_v^{(q)}\right)\cdot\widehat{G}_v\right\|_2^2 \leq \sum_{\boldsymbol{\xi}\in[n]^d\setminus\mathrm{supp}(\mathrm{FRONTIER})}|\widehat{y}(\boldsymbol{\xi})|^2 + \left\|\left(\widehat{y}-\widehat{\chi}_v^{(q)}\right)_{\mathrm{FreqCone}_{\mathrm{FRONTIER}}(v)}\right\|_2^2$$

$$\leq \sum_{\boldsymbol{\xi}\in[n]^d\setminus\mathrm{supp}(\mathrm{FRONTIER})}|\widehat{y}(\boldsymbol{\xi})|^2 + \left\|\widehat{y}_{\mathrm{FreqCone}_{\mathrm{FRONTIER}}(v)\setminus\mathrm{HEAD}_\mu(\widehat{y})}\right\|_2^2 + \frac{\mu^2}{40}$$

$$\leq \left\|\widehat{y}-\widehat{y}_{\mathrm{HEAD}_\mu(\widehat{y})}\right\|_2^2 + \frac{\mu^2}{40} \leq \frac{11}{10}\cdot\mu^2.$$

Thus, the preconditions of the second claim of Lemma 17 hold. So, we can invoke this lemma to conclude that the if-statement in line 34 of the algorithm is False and hence the algorithm outputs $\left(\mathrm{True},\widehat{\chi}_v^{(q)}\right)$. This completes the inductive proof of statement 1 of the lemma.

Now we proceed with the inductive step towards proving the second statement of lemma. Suppose that preconditions i, ii, iii along with the precondition of statement 2 (that is $|S| > k$) hold. Lemma 22 proved that the signal $\widehat{\chi}_v$ always satisfies $\mathrm{supp}(\widehat{\chi}_v) \subseteq \mathrm{FreqCone}_{\mathrm{FRONTIER}}(v)$ and $\|\widehat{\chi}_v\|_0 \leq k$. Therefore, $S\setminus\mathrm{supp}(\widehat{\chi}_v) \neq \varnothing$. Consequently, if $\widehat{G}_v$ is a Fourier domain $(v,\mathrm{FRONTIER})$-isolating filter constructed in Lemma 9, then by definition of isolating filters we have

$$\left\|\left((\widehat{y}-\widehat{\chi}_v)\cdot\widehat{G}_v\right)_{S\cup\mathrm{supp}(\widehat{\chi}_v)}\right\|_2^2 \geq \left\|(\widehat{y}-\widehat{\chi}_v)_{S\cup\mathrm{supp}(\widehat{\chi}_v)}\right\|_2^2 \geq \left\|\widehat{y}_{S\setminus\mathrm{supp}(\widehat{\chi}_v)}\right\|_2^2 \geq 9\mu^2,$$

which follows from the definition of S and $\mathrm{HEAD}_\mu(\cdot)$. On the other hand,

$$\left\|\left((\widehat{y}-\widehat{\chi}_v)\cdot\widehat{G}_\ell\right)_{[n]^d\setminus(S\cup\mathrm{supp}(\widehat{\chi}_v))}\right\|_2^2 = \left\|\left(\widehat{y}\cdot\widehat{G}_\ell\right)_{[n]^d\setminus(S\cup\mathrm{supp}(\widehat{\chi}_v))}\right\|_2^2$$

$$\leq \left\|\left(\widehat{y}\cdot\widehat{G}_\ell\right)_{[n]^d\setminus S}\right\|_2^2$$

$$\leq \left\|\widehat{y}_{\mathrm{FreqCone}_{\mathrm{FRONTIER}}(v)\setminus S}\right\|_2^2$$

$$+ \sum_{\boldsymbol{\xi}\in[n]^d\setminus\mathrm{supp}(\mathrm{FRONTIER})}|\widehat{y}(\boldsymbol{\xi})|^2$$

$$\leq \left\|\widehat{y}-\widehat{y}_{\mathrm{HEAD}_\mu(\widehat{y})}\right\|_2^2 \leq \frac{11}{10}\cdot\mu^2. \qquad \text{(precondition ii)}$$

Additionally note that $|S\cup\mathrm{supp}(\widehat{\chi}_v)| \leq k/\alpha + k \leq 2k/\alpha$ by preconditions of the lemma and property of $\mathrm{supp}(\widehat{\chi}_v)$ that we have proved. Hence, by invoking the first claim of Lemma 17, the if-statement in line 34 of the algorithm is True and hence the algorithm outputs $\left(\mathrm{False},\{0\}^{n^d}\right)$. This proves statement 2 of the lemma.

Finally, observe that throughout this analysis we have assumed that Lemma 17 holds with probability 1 for all the invocations of HEAVYTEST by our algorithm. Moreover, we assumend that ESTIMATE successfully works with probability 1. Also we assumed that the inductive hypothesis (that is Lemma 23 for sparsity parameters $k \leq m-1$) holds deterministically. In reality, we have to take the fact that these primitives are randomized into acount of our analysis.

The first source of randomness is the fact that HEAVYTEST only succeeds with some high probability. In fact, Lemma 17 tells us that every invocation of HEAVYTEST succeeds with probability at least $1-1/N^5$.

The second source of randomness is the fact that ESTIMATE only succeeds with some high probability. Lemma 18 tells us that every invocation of ESTIMATE on a set Cheap, succeeds

with probability $1 - \frac{|\text{Cheap}|}{N^8} \geq 1 - \frac{1}{N^7}$. Since, our analysis in proof of Lemma 22 shows that RECURSIVEROBUSTSFT makes at most k recursive calls to ESTIMATE, by a union bound, the overall failure probability of all invocations of this primitive will be bounded by $\frac{k}{N^7}$.

The third and last source of randomness in our algorithm is the recursive invocations of RECURSIVEROBUSTSFT in lines 21 and 22 of our algorithm. By the inductive hypothesis (statement of Lemma 23), the invocation of this primitive succeeds with probability $1 - O\left(\left(\frac{2\log N}{\alpha}\right)^{\log_{1/\alpha} b} \cdot N^{-4}\right)$. Our analysis in proof of Lemma 22 shows that RECURSIVEROBUSTSFT makes at most $\frac{2\log N}{\alpha}$ recursive calls to RECURSIVEROBUSTSFT. Therefore, by a union bound, the overall failure probability of all invocations of RECURSIVEROBUSTSFT is bounded by $O\left(\left(\frac{2\log N}{\alpha}\right)^{\log_{1/\alpha} k} \cdot N^{-4}\right)$.

Finally, by another application of union bound, the overall failure probability of Algorithm 8, is bounded by $O\left(\left(\frac{2\log N}{\alpha}\right)^{\log_{1/\alpha} k} \cdot N^{-4}\right)$. This completes the proof of the lemma.

$\square$

Now we are ready to present our main robust sparse Fourier transform algorithm that achieves the guarantee of Theorem 4 for any ϵ using a number of samples that is near quadratic in k and a runtime that is cubic and prove the main result of this section.

Proof of Theorem 4: The procedure that achieves the guarantees of the theorem is presented in Algorithm 9. The correctness proof basically follows by invoking Lemma 23 and the runtime and sample complexity follows from Lemma 22. If we let $\mu := \|\eta\|_2$ then because x is a signal in the k-high SNR regime, we have that $|\text{HEAD}_\mu(\widehat{x})| \leq k$ and $\left\|\widehat{x} - \widehat{x}_{\text{HEAD}_\mu(\widehat{x})}\right\|_2 \leq \mu$. Therefore, the signal $\widehat{\chi}$ that we computed in line 3 of Algorithm 9 by running procedure RECURSIVEROBUSTSFT (Algorithm 8) with inputs $\left(x, \{0\}^{n^d}, \{\text{root}\}, \text{root}, k, \alpha, \mu\right)$, then all preconditions of Lemma 23 hold and hence by invoking the first statement of this lemma we conclude that, with probability at least $1 - \frac{1}{2N^3}$, $\widehat{\chi}$ satisfies the following properties:

$$\|\widehat{x} - \widehat{\chi}\|_2^2 \leq \frac{\mu^2}{40} \quad \text{and} \quad \text{supp}(\widehat{\chi}) \subseteq \text{HEAD}_\mu(\widehat{x}).$$

This together with the k-high SNR assumption imply that, with probability at least $1 - \frac{1}{2N^3}$, $\text{supp}(\widehat{\chi}) = \text{HEAD}_\mu(\widehat{x})$. Therefore, tree T that we construct in line 4 of Algorithm 9 is in fact the spliting tree of the set $\text{HEAD}_\mu(\widehat{x})$, that is, $\text{supp}(T) = \text{HEAD}_\mu(\widehat{x})$ and $|\text{LEAVES}(T)| = |\text{HEAD}_\mu(\widehat{x})|$.

In the rest of the correctness proof we condition on the event that tree T is the spliting tree of the set $\text{HEAD}_\mu(\widehat{x})$ and analyze the evolution of singal $\widehat{\chi}_\epsilon$ and tree T in every iteration $t = 0, 1, 2, \ldots$ of the *while loop* in Algorithm 9. Let $\widehat{\chi}_\epsilon^{(t)}$ denote the signal $\widehat{\chi}_\epsilon$ at the end of iteration t, and let $T^{(t)}$ denote the tree T at the end of iteration t. In every iteration t, Algorithm 9 computes a subset $\text{Cheap}^{(t)}$ of leaves of the tree $T^{(t-1)}$ by running the primitive FINDCHEAPTOESTIMATE in line 7 of the algorithm. By Claim 6, the set $\text{Cheap}^{(t)} \subseteq \text{LEAVES}\left(T^{(t-1)}\right)$ satisfies the property that $\left|\text{Cheap}^{(t)}\right| \cdot (8 + 4\log k) \geq \max_{u \in \text{Cheap}^{(t)}} 2^{w_{T^{(t-1)}}(u)}$. Clearly $\text{Cheap}^{(t)} \neq \varnothing$, by Claim 6. Then the algorithm computes $\{\widehat{H}_u\}_{u \in \text{Cheap}^{(t)}}$ by running the procedure ESTIMATE in line 9 and then updates $\widehat{\chi}_\epsilon^{(t)}(\boldsymbol{f}_u) \leftarrow \widehat{H}_u$ for every $u \in \text{Cheap}^{(t)}$ and $\widehat{\chi}_\epsilon^{(t)}(\boldsymbol{\xi}) = \widehat{\chi}_\epsilon^{(t-1)}(\boldsymbol{\xi})$ at every other frequency $\boldsymbol{\xi}$. Moreover, the algorithm updates the tree $T^{(t)}$ by removing every leaf that is in the set Cheap from tree $T^{(t-1)}$. Hence, one can readily see that since at each iteration of the while loop, tree T looses at least one of its leaves, the algorithm terminates after at most $\left|\text{LEAVES}\left(T^{(0)}\right)\right| = k$ iterations, since initially the number of leaves of $T^{(0)}$ equals $|\text{HEAD}_\mu(\widehat{x})| = k$.

80

If we denote by $S^{(t)}$ the set $\text{supp}\left(\widehat{\chi}_\epsilon^{(t)}\right)$ for every t, then we claim that the following holds,

$$\Pr\left[\left\|\widehat{x}_{S^{(t)}} - \widehat{\chi}_\epsilon^{(t)}\right\|_2^2 \leq \frac{\epsilon\left|S^{(t)}\right|}{k} \cdot \mu^2\right] \geq 1 - \frac{\left|S^{(t)}\right|}{N^8}.$$

We prove the above claim by induction on iteration number t of the *while loop* of our algorithm. One can see that the base of induction trivially holds for $t = 0$ because $\widehat{\chi}_\epsilon^{(0)} \equiv 0$. To prove the inductive step, suppose that the inductive hypothesis holds for $t - 1$, that is,

$$\Pr\left[\left\|\widehat{x}_{S^{(t-1)}} - \widehat{\chi}_\epsilon^{(t-1)}\right\|_2^2 \leq \frac{\epsilon\left|S^{(t-1)}\right|}{k} \cdot \mu^2\right] \geq 1 - \frac{\left|S^{(t-1)}\right|}{N^8}.$$

If we let $L := \left\{\boldsymbol{f}_u : u \in \text{Cheap}^{(t)}\right\}$, then one can see from the way our algorithm updates signal $\widehat{\chi}_\epsilon^{(t)}$ and tree $T^{(t)}$ that $S^{(t)} \setminus S^{(t-1)} = L$ for every iteration t. Furthermore, by Lemma 18 and union bound, we find that with probability at least $1 - \frac{|S^{(t-1)}|}{N^8} - \frac{|\text{Cheap}^{(t)}|}{N^8} = 1 - \frac{|S^{(t-1)}|}{N^8}$ the following holds

$$\begin{aligned}
\left\|\widehat{x}_{S^{(t)}} - \widehat{\chi}_\epsilon^{(t)}\right\|_2^2 &= \left\|(\widehat{x} - \widehat{\chi}_\epsilon^{(t)})_{S^{(t-1)}}\right\|_2^2 + \left\|(\widehat{x} - \widehat{\chi}_\epsilon^{(t)})_{S^{(t)} \setminus S^{(t-1)}}\right\|_2^2 \\
&= \left\|\widehat{x}_{S^{(t-1)}} - \widehat{\chi}_\epsilon^{(t-1)}\right\|_2^2 + \left\|(\widehat{x} - \widehat{\chi}_\epsilon^{(t)})_L\right\|_2^2 \\
&\leq \frac{\epsilon|S^{(t-1)}|\mu^2}{k} + \frac{\epsilon|L|}{2k} \sum_{\boldsymbol{\xi} \in [n]^d \setminus \text{supp}(T^{(t-1)})} \left|\left(\widehat{x} - \widehat{\chi}_\epsilon^{(t-1)}\right)(\boldsymbol{\xi})\right|^2. \quad (24)
\end{aligned}$$

Now we bound the second term above,

$$\begin{aligned}
&\sum_{\boldsymbol{\xi} \in [n]^d \setminus \text{supp}(T^{(t-1)})} \left|\left(\widehat{x} - \widehat{\chi}_\epsilon^{(t-1)}\right)(\boldsymbol{\xi})\right|^2 \\
&= \sum_{\boldsymbol{\xi} \in [n]^d \setminus \left(\text{supp}(T^{(t-1)}) \cup S^{(t-1)}\right)} |\widehat{x}(\boldsymbol{\xi})|^2 + \left\|\widehat{x}_{S^{(t-1)}} - \widehat{\chi}_\epsilon^{(t-1)}\right\|_2^2 \\
&\leq \sum_{\boldsymbol{\xi} \in [n]^d \setminus \text{HEAD}_\mu(\widehat{x})} |\widehat{x}(\boldsymbol{\xi})|^2 + \left\|\widehat{x}_{S^{(t-1)}} - \widehat{\chi}_\epsilon^{(t-1)}\right\|_2^2 \quad (T \text{ was initially the splitting tree of } \text{HEAD}_\mu(\widehat{x})) \\
&\leq 2\mu^2 \quad \text{(by the inductive hypothesis)}.
\end{aligned}$$

Therefore, by plugging the above bound back to (24) we find that,

$$\Pr\left[\left\|\widehat{x}_{S^{(t)}} - \widehat{\chi}_\epsilon^{(t)}\right\|_2^2 \leq \frac{\epsilon\left|S^{(t)}\right|}{k} \cdot \mu^2\right] \geq 1 - \frac{\left|S^{(t)}\right|}{N^8},$$

which proves the inductive claim. Therefore, by another application of union bound, with probability at least $1 - \frac{1}{N^3}$, the output of the algorithm $\widehat{\chi}_\epsilon$ satisfies $\|\widehat{x} - \widehat{\chi}_\epsilon\|_2^2 \leq (1 + \epsilon) \cdot \mu^2$. This proves the correctness of Algorithm 9.

Runtime and Sample Complexity. By Lemma 22, the running time and sample complexity of invoking primitive RECURSIVEROBUSTSFT in line 3 of the algorithm are bounded by $\widetilde{O}(k^3)$ and $\widetilde{O}\left(k^2 \cdot 2^{2\sqrt{\log k \cdot \log(2\log N)}}\right)$, respectively. Additionally, by Lemma 18, the runtime and

81

sample complexity of every invocation of ESTIMATE in line 9 of our algorithm are bounded by $\widetilde{O}\left(\frac{k}{\epsilon|\text{Cheap}^{(t)}|}\sum_{u\in\text{Cheap}^{(t)}}2^{w_{T^{(t-1)}}(u)}+\frac{k}{\epsilon}\cdot\|\widehat{\chi}_\epsilon^{(t-1)}\|_0\right)$ and $\widetilde{O}\left(\frac{k}{\epsilon|\text{Cheap}^{(t)}|}\sum_{u\in\text{Cheap}^{(t)}}2^{w_{T^{(t-1)}}(u)}\right)$, respectively. Using the fact that $|\text{Cheap}^{(t)}|\cdot(8+4\log k)\geq\max_{u\in\text{Cheap}^{(t)}}2^{w_{T^{(t-1)}}(u)}$ together with $\|\widehat{\chi}_\epsilon^{(t-1)}\|_0\leq k$, these time and sample complexities are further upper bounded by $\widetilde{O}\left(\frac{k|\text{Cheap}^{(t)}|}{\epsilon}+\frac{k^2}{\epsilon}\right)$ and $\widetilde{O}\left(\frac{k}{\epsilon}\cdot|\text{Cheap}^{(t)}|\right)$, respectively. We proved that the total number of iterations, and hence number of times we run ESTIMATE in line 9 of the algorithm, is bounded by k. Using this together with the fact that $\sum_t\left|\text{Cheap}^{(t)}\right|=\|\widehat{\chi}_\epsilon\|_0=|\text{HEAD}_\mu(\widehat{x})|\leq k$, the total runtime and sample complexity of all invocations of ESTIMATE in all iterations can be upper bounded by $\widetilde{O}\left(\frac{k^3}{\epsilon}\right)$ and $\widetilde{O}\left(\frac{k^2}{\epsilon}\right)$, respectively. Therefore the total time and sample complexities of our algorithm are bounded by $\widetilde{O}\left(\frac{k^3}{\epsilon}\right)$ and $\widetilde{O}\left(\frac{k^2}{\epsilon}+k^2\cdot2^{2\sqrt{\log k\cdot\log(2\log N)}}\right)$, respectively. $\square$

14 Experiments.

In this section, we empirically show that our FFT backtracking algorithm for high dimensional sparse signals is extremely fast and can compete with highly optimized software packages such as the FFTW [Fri99, FJ]. Our experiments mainly focus on a modification of Algorithm 1 which exploits only one level of FFT backtracking and runs in $\widetilde{O}(k^{2.5})$ time. One of the baselines that we compare our algorithm to is the vanilla FFT tree pruning of [KVZ19], in order to demonstrate the speed gained by our backtracking technique. Furthermore, we compare our method against the SFFT 2.0 [HIKP12b, HIKP], which is optimized for 1-dimensional signals, and show that our method's performance for small sparsity k is comparable to that of the SFFT 2.0 even in dimension one.

In a subset of our experiments, we exploit a technique introduced in [GHI+13] to speed up the high-dimensional Sparse FFT algorithms. This method works as follows. By fixing one of the coordinates of a d-dimensional signal we get a $(d-1)$-dimensional signal whose Fourier transform corresponds to projecting (aliasing) the Fourier transform of the original signal along the coordinate that was fixed in time domain. Thus we can effectively *project* the Fourier spectrum into a $(d-1)$-dimensional plane by computing a $(d-1)$-dimensional FFT. Using a small number of measurements (projections with different values of the fixed coordinate) we can figure out which frequencies are projected without collision and recover them. We use this trick to recover the frequencies that get isolated under the projection and then run our algorithm on the residual signal. Since the residual signal is likely to have a smaller sparsity than the original one, this *projection technique* can speed up our Sparse FFT algorithms.

Sparse signal classes: In our experiments, we benchmark all methods on the following classes of k-sparse signals:

1. **Random support with overtones:** The Fourier spectrum of this signal class is the superposition of a set of random frequencies and a set of overtones of these frequencies. Specifically, the support of this signal is $\text{supp}(\widehat{x})=S_{\text{RANDOM}}\cup S_{\text{OVERTONE}}$, which are defined as follows,

$$S_{\text{RANDOM}}:=\left\{\boldsymbol{f}_1,\boldsymbol{f}_2,\ldots\boldsymbol{f}_{k/(d+1)}\sim\text{i.i.d. UNIF}(\mathbb{Z}_n^d)\right\},$$

$$S_{\text{OVERTONE}}:=\left\{\boldsymbol{f}+(n/2)\cdot\mathbf{e}_i:\forall\boldsymbol{f}\in S_{\text{RANDOM}},i\in[d]\right\},$$

where $\mathbf{e}_i$ is the standard basis vector along coordinate i in dimension d. Note that every $\boldsymbol{f} \in S_{\text{RANDOM}}$ will collide with at least one overtone under projection along any coordinate, thus, S_{RANDOM} cannot be recovered using the projection trick. We added the overtones precisely for this reason, i.e., to ensure that the projection trick does no recover the signal entirely and there will be something left for the Sparse FFT to recover.

2. **Randomly shifted d-dimensional Dirac Comb:** The Fourier support of a Dirac Comb (without shift) is the following,

$$S_{\text{COMB}} := \left\{ \left(i_1 \cdot \frac{n}{k^{1/d}}, i_2 \cdot \frac{n}{k^{1/d}}, \ldots i_d \cdot \frac{n}{k^{1/d}} \right) : i_1, i_2, \ldots i_d \in [k^{1/d}] \right\}.$$

We generate a random frequency shift $\tilde{\boldsymbol{f}} \sim \text{UNIF}(\mathbb{Z}_n^d)$ and a random phase shift $\tilde{\boldsymbol{t}} \sim \text{UNIF}(\mathbb{Z}_n^d)$ then define the k-sparse $\widehat{x}$ as,

$$\widehat{x}_{\boldsymbol{f}} := \sum_{\boldsymbol{j} \in S_{\text{COMB}}} e^{2\pi i \frac{\boldsymbol{f}^\top \tilde{\boldsymbol{t}}}{n}} \cdot \mathbb{1}_{\{\boldsymbol{f} = \boldsymbol{j} + \tilde{\boldsymbol{f}}\}}.$$

Note that the projection trick will not help at all on this signal and thus it is a good test case for the Sparse FFT algorithms. Additionally, this signal in time domain is also a randomly shifted Dirac Comb with sparsity N/k and thus distinguishing it from zero with constant probability would require $\Omega(k)$ samples. This makes the Dirac Comb a hard test case for our tree exploration algorithms which heavily rely on the ZEROTEST primitive to distinguish a sparse signal from a zero signal.

3. **Superposition of a $k/2$-sparse signal with random support and a d-dimensional Dirac Comb of sparsity $k/2$:** This signal is a mixture of instances defined in **(1)** and **(2)**

4. **Superposition of two randomly shifted d-dimensional Dirac Combs of sparsity $k/2$:** This signal is a mixture of two independent instances of the randomly shifted Dirac Comb defined in **(2)**.

14.1   FFT Backtracking vs Vanilla FFT Tree Pruning.

We first show that our backtracking technique highly improves the runtime of FFT tree pruning and compare our algorithm against the vanilla tree exploration of Kapralov et al. [KVZ19] as a baseline. We run both algorithms on a variety of sparse signals of size $N = 2^{21}$ in dimension $d = 3$. We tune the parameters of both algorithms to achieve success probabilities of higher than 90% over 100 independent trials with different random seeds. Projection recovery [GHI+13] is turned off for both algorithms to fairly demonstrate the effect of our backtracking technique. In Figure 6, we benchmark our methods on 3 different classes of k-sparse signals and observe that our Backtracked Sparse FFT algorithm consistently achieves a faster runtime and also scales slower as a function of sparsity k compared to the Vanialla Sparse FFT Tree Pruning of [KVZ19].

14.2   Sparse FFT Backtracking vs FFTW.

Next we compare our algorithm against the highly optimized FFTW 3.3.9 software package and show that our algorithm outperforms FFTW by a large margin when the signal size N is large. We run both algorithms on a variety of signals of sparsity $k = 32$ in dimension $d = 3$. As in previous set of experiments, the parameters of our algorithm is tuned to succeed in over 90% of instances. In Figure 7, we benchmark our method and the FFTW on 4 different classes of k-sparse signals and
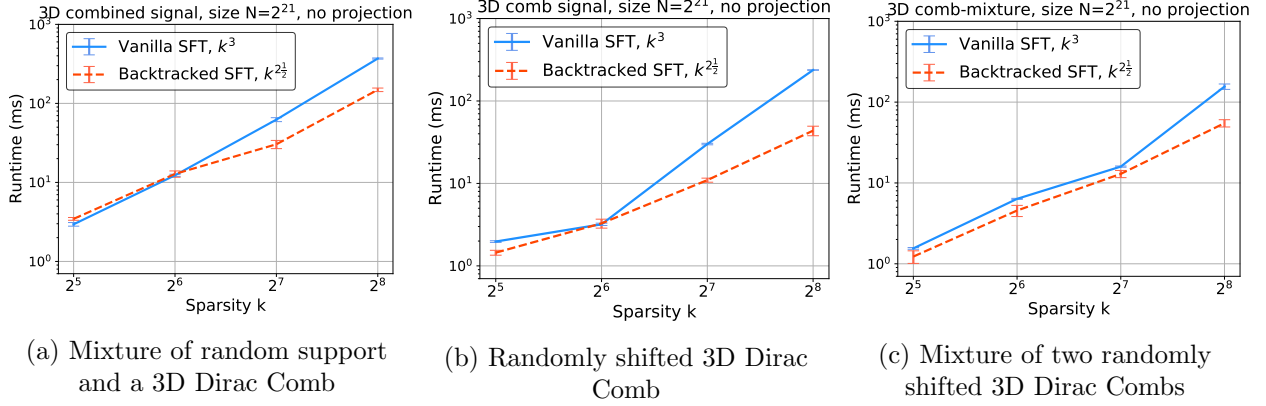
83

(a) Mixture of random support and a 3D Dirac Comb

(b) Randomly shifted 3D Dirac Comb

(c) Mixture of two randomly shifted 3D Dirac Combs

Figure 6: The runtime of recovering: **(a)** superposition of a $k/2$-sparse signal with random support and a 3D Dirac Comb of sparsity $k/2$, **(b)** a randomly shifted 3D Dirac Comb with sparsity k, and **(c)** mixture of two randomly shifted 3D Dirac Combs of sparsities $k/2$.
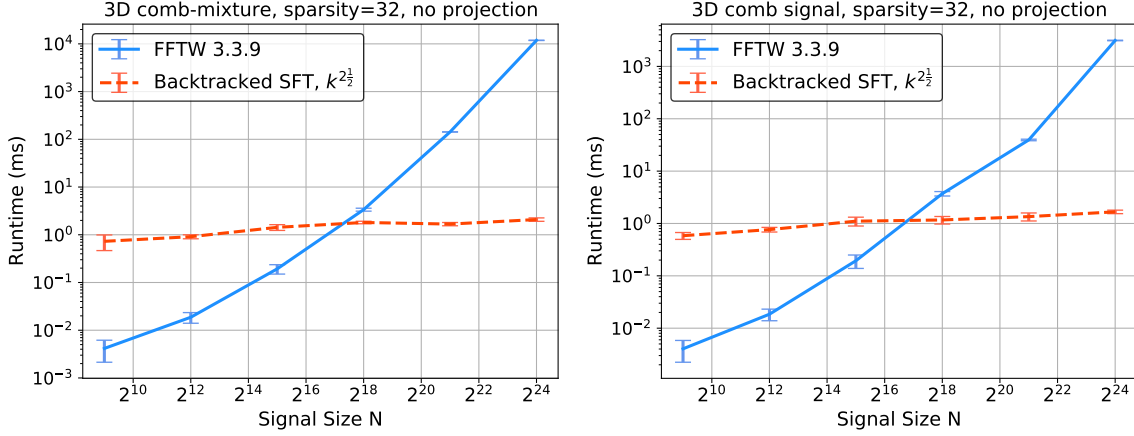
observe that in all cases the runtime of our Backtracked Sparse FFT algorithm scales very weakly with signal size N, particularly, our runtime grows far slower than that of FFTW. Consequently our algorithm is orders of magnitude faster than FFTW for any $N \geq 2^{18}$.

14.3   Comparison to SFFT 2.0 in Dimension One.

Finally, in this set of experiments we compare our modified Algorithm 1 against the SFFT software package [HIKP] which is highly optimized for 1-dimensional sparse signals and show that we can achieve comparable performance even in dimension one. We run both algorithms on two classes of signals with sparsity $k = 32$ in dimension $d = 1$. We remark that the runtime of SFFT, which is implemented based on [HIKP12b], will certainly scale badly in high dimensions due to filter support increasing. However, since there is no optimized code available for SFFT in high dimensions, we feel that it is more informative to compare our optimized code to their optimized code in 1D rather than have a weak extension of their approach as a benchmark.

The SFFT package includes two versions: 1.0 and 2.0. The difference is that SFFT 2.0 adds a Comb prefiltering heuristic to improve the runtime. The idea of this heuristic is to apply the aliasing filter, which is very efficient and has no leakage, to restrict the locations of the large coefficients according to their values mod some number $B = O(k)$. The heuristic, in a preprocessing stage, subsamples the signal at rate $1/B$ and then takes the FFT of the subsampled signal.

In Figure 8, we benchmark our method and SFFT (1.0 and 2.0) on 2 different classes of k-sparse signals and observe that the runtime of our Backtracked Sparse FFT algorithm is comparable to that of SFFT. In Fig. 8a we run the algorithms on a signal with random Fourier support and observe that SFFT 2.0 runs slightly faster. Since the support is random, the heuristic trick used in SFFT 2.0 can recover a large portion of the frequencies and thus SFFT 2.0 owes much of its speed to the heuristic trick. On the other hand, in Fig. 8b, we run the algorithms on a randomly shifted Dirac Comb and observe that our method outperforms SFFT 1.0. Note that since the Comb prefiltering heuristic used in SFFT 2.0 completely fails on a Dirac Comb input, we used SFFT 1.0 in this experiment instead. This result demonstrates that for signals with small sparsity k, our algorithm can run even faster than SFFT when the input's support is a multiplicative subgroup of $\mathbb{Z}_n$, such as the Dirac Comb.

84

(a) The input signal classes are: (Left) mixture of two randomly shifted 3D Dirac Combs and (Right) a randomly shifted 3D Dirac Comb

(b) The input signal classes are: (Left) a random support signal with overtones and (Right) mixture of random support and a randomly shifted 3D Dirac Comb

Figure 7: The runtime of recovering various signal classes with sparsity $k = 32$. We consider two variants of our Backtracked Sparse FFT: **(a)** purely modified Algorithm 1 with no prefiltering or projection tricks, **(b)** enhanced version of modified Algorithm 1 which first applies the projection trick.

15 Acknowledgements.

A Analysis of the Cubic Time Tree Exploration Algorithm.

This section is devoted to proving the correctness and runtime of Algorithm 10.

The idea behind Algorithm 10 is to recover all non-zero leaves in the subtree of T_N^{full} rooted at
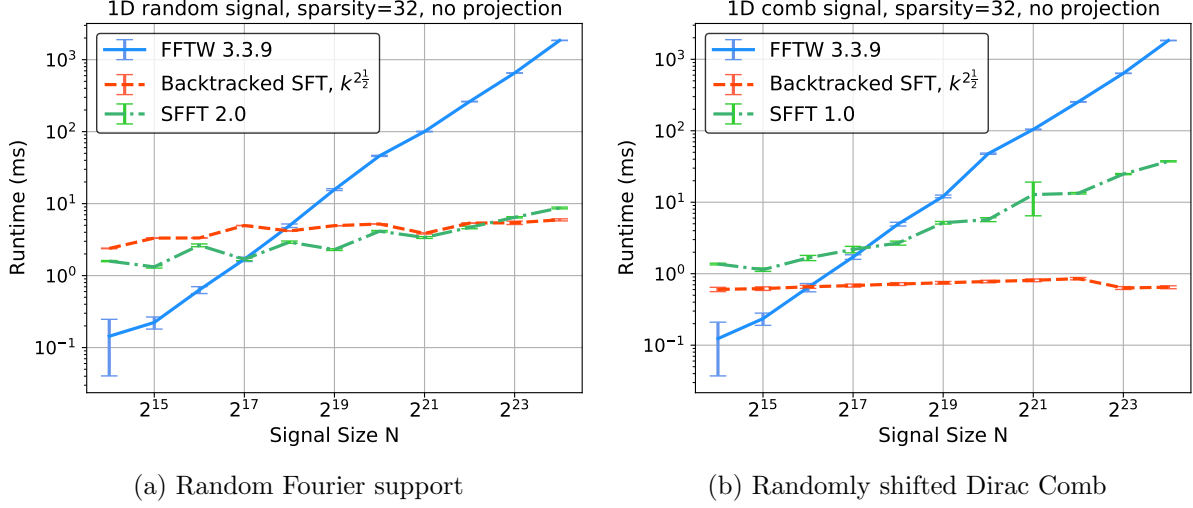
85

(a) Random Fourier support   (b) Randomly shifted Dirac Comb

Figure 8: The runtime of recovering: **(a)** k-sparse signal with random support and **(b)** a randomly shifted Dirac Comb with sparsity k.

v, given that $|\text{HEAVYLEAVES}(v)| \leq b$ and v is isolated by Found and Excluded. Algorithm 10 is essentially a slightly modified version of [KVZ19], but since in this paper we work with an abstracted problem, we still present the proof of its correctness and runtime. One of the useful tools for this algorithm is Lemma 10, which states that for any tree T, the minimum weight $w_T(\ell)$ of a leaf ℓ in T is at most $\log L$, where L is the number of leaves of T.

Theorem 14 (Theorem 5, restated). *If* $|\text{HEAVYLEAVES}(v)| \leq b$ *and* v *is isolated by* Found *and* Excluded, *then the procedure* SLOWEXACTSPARSERECOVERY *returns the correct estimates for all* HEAVYLEAVES(v).

Proof. Assume for the moment that the check in line 3 is not made. We will show inductively that the following invariant holds at the end of each **repeat** loop: all estimated values in Found_{out} are correct, and $\text{HEAVYLEAVES}(v) \setminus \text{Found}_{out} \subset \text{LEAVES}(\text{FRONTIER}_v)$. Then the correctness would follow from the fact that $\text{FRONTIER}_v = \varnothing$ at the end of the procedure.

It is easy to see that invariants hold before the loop starts. Now, suppose that at the start of an arbitrary iteration the invariants hold for current sets Found_{out} and FRONTIER_v. Because the invariants hold, the node z picked in line 5 is isolated by $\text{Found} + \text{Found}_{out}$ and $\text{Excluded} \cup \text{FRONTIER}_v$. If z is a leaf in T_N^{full}, because z is isolated, ESTIMATE produces the correct estimate for z, therefore, the invariants still hold at the end of the loop. If z is not a leaf in T_N^{full}, then because z is isolated and $\text{HEAVYLEAVES}(z) \subseteq \text{HEAVYLEAVES}(v)$, the prerequisites for calling ZEROTEST are fulfilled, and its output is correct. If it says True, we can just delete z from FRONTIER_v without violating the invariants, and otherwise we add both it's children instead. Since $\text{HEAVYLEAVES}(z_{\text{left}}) \cup \text{HEAVYLEAVES}(z_{\text{right}}) = \text{HEAVYLEAVES}(z)$, the invariants still hold.

Finally, even if we do the check in line 3, the execution will still be the same, since we do at most $3b \log N$ iterations. Notice that each node gets added at most once to FRONTIER_v, and on each iteration one node is removed from FRONTIER_v. Also notice that if for the node z, $\text{HEAVYLEAVES}(z) = \varnothing$, it's children will not be added to FRONTIER_v, because ZEROTEST would return True. Since there is at most $b \log N$ vertices z such that $\text{HEAVYLEAVES}(z) \neq \varnothing$, and each of them have only 2 children, there is at most $3b \log N$ vertices that can be added to FRONTIER_v, hence the maximum number of iterations is $6b \log N$. $\qquad\square$

86

Algorithm 10 SLOWEXACTSPARSERECOVERY(Found, Excluded, v, b)

1: FRONTIER$_v \leftarrow \{v\}$, Steps $\leftarrow 1$, Found$_{out} \leftarrow \varnothing$
2: **repeat**
3:     **if** Steps $> 6 \cdot b \log N$ **then** $\triangleright$ Have explored more than the expected sparsity
4:         **return** $\varnothing$
5:     $z :=$ vertex in FRONTIER$_v$ with the minimum weight with respect to FRONTIER$_v$.
6:     FRONTIER$_v = $ FRONTIER$_v \setminus \{z\}$, Steps $\leftarrow$ Steps $+ 1$
7:     Excluded$' \leftarrow$ Excluded $\cup$ FRONTIER$_v$, Found$' \leftarrow$ Found $+$ Found$_{out}$
8:     **if** z is a leaf in T^{full} **then**
9:         Found$_{out}(z) \leftarrow$ ESTIMATE(Found$'$, Excluded$'$, z)
10:     **else if** ZEROTEST(Found$'$, Excluded$'$, z, b) $=$ False **then** $\triangleright$ LEAVES(z) contains heavy leaves
11:         $z_{\text{left}} \leftarrow$ left child of z
12:         $z_{\text{right}} \leftarrow$ right child of z
13:         FRONTIER$_v \leftarrow$ FRONTIER$_v \cup \{z_{\text{left}}, z_{\text{right}}\}$
            $\triangleright$ LEAVES(z) has no heavy leaves we simply remove it from FRONTIER, (see line 6).
14: **until** FRONTIER$_v = \varnothing$
15: **return** Found$_{out}$

Theorem 15 (Theorem 7, restated). *If* LEAVES(v) $\cap$ LEAVES(Excluded) $= \varnothing$, *the running time of* SLOWEXACTSPARSERECOVERY *is upper bounded by*

$$\widetilde{O}\left(|\mathsf{Found}| \cdot b^2 + 2^{w_{\mathsf{Excluded}}(v)} \cdot b^3\right).$$

Proof. First, notice that since LEAVES(v) $\cap$ LEAVES(Excluded) $= \varnothing$, for all z in the subtree of v at any iteration, $w_{\mathsf{Excluded} \cup \text{FRONTIER}_v}(z) = w_{\text{FRONTIER}_v}(z) + w_{\mathsf{Excluded}}(v)$. Because of the check in line 3, the algorithm runs for at most $6b \log N$ iterations, and, consequently, $|\text{FRONTIER}_v| \leq 6b \log N$. By Lemma 10, for each chosen z, $2^{w_{\text{FRONTIER}_v}(z)} \leq 6b \log N$. Similarly, $|\mathsf{Found}_{out}| \leq 6b \log N$. Therefore, each call to ZEROTEST uses

$$\widetilde{O}(2^{w_{\mathsf{Excluded} \cup \text{FRONTIER}_v}(z)} \cdot b + |\mathsf{Found} + \mathsf{Found}_{out}| \cdot b) = \widetilde{O}(2^{w_{\mathsf{Excluded}}(v)} \cdot b^2 + |\mathsf{Found}| \cdot b + b^2)$$

operations. Summing over all iterations, we find that the total runtime of all calls to ZEROTEST is bounded by $\widetilde{O}(2^{w_{\mathsf{Excluded}}(v)} b^3 + |\mathsf{Found}| \cdot b^2)$.

Similarly, each call of ESTIMATE spends $\widetilde{O}(2^{w_{\mathsf{Excluded} \cup \text{FRONTIER}_v}(z)} + |\mathsf{Found} + \mathsf{Found}_{out}|)$ operations, which accumulates to $\widetilde{O}(2^{w_{\mathsf{Excluded}}(v)} \cdot b^2 + |\mathsf{Found}| \cdot b)$ across all iteration.

Finally, notice that to maintain the set Excluded$'$ and its tree $T(\mathsf{Excluded})$, we first need to copy it from Excluded and add v. However, because we only work inside the subtree of v, we can reduce the tree $T(\mathsf{Excluded})$ and, respectively, it's set to only contain the path to v and all of the children of the vertices in this path. This can be done in time $O(\log N + w_{\mathsf{Excluded}}(v))$. Then, on each iteration, Excluded$'$ is only modified by removing one vertex from it, which can be done in time $O(\log N)$. Hence, the total time spent on maintaining Excluded$'$ is $\widetilde{O}(b + w_{\mathsf{Excluded}}(v))$.

Since runtime of each iteration is dominated by the calls to ZEROTEST and/or ESTIMATE, the total running time is $\widetilde{O}(2^{w_{\mathsf{Excluded}}(v)} \cdot b^3 + |\mathsf{Found}| \cdot b^2)$. $\qquad\square$

B   Proof of Lemma 4.

Proof. Observe that the set of edges of tree $T = T(S_1 \cup S_2 \cup \{v\})$ is the union of the sets of edges of trees $T_1 = T(S_1 \cup \{v\})$ and $T_2 = T(S_2 \cup \{v\})$. Consider the set of all ancestors of v with two

children. For ancestor u, one of those children, say z, lies on the path from the v to the root, and, therefore, the edge (u, z) is a part of both of the trees T_1, T_2. Now consider the other child of u, z'. Because edge (u, z') exists in T, it also exists in one of the trees T_1 or T_2. But then u also has two children in that tree. Therefore, each ancestor of v with two children has two children in T_1 or T_2 as well. Therefore, by definition of w_T, we get that $w_T(v) \le w_{T_1}(v) + w_{T_2}(v)$, which is equivalent to the desired inequality. $\qquad\square$

References

[ABDN18] Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. More consequences of falsifying SETH and the orthogonal vectors conjecture. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 253–266. ACM, 2018.

[AGS03] A Akavia, S Goldwasser, and S Safra. Proving hard-core predicates using list decoding. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 146–157. IEEE, 2003.

[Aka10] Adi Akavia. Deterministic sparse fourier approximation via fooling arithmetic progressions. In *COLT*, pages 381–393, 2010.

[AWW14] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 39–51. Springer, 2014.

[AZKK19] Andisheh Amrollahi, Amir Zandieh, Michael Kapralov, and Andreas Krause. Efficiently Learning Fourier Sparse Set Functions. *Advances In Neural Information Processing Systems 32 (Nips 2019)*, 32(CONF), 2019.

[BCG+12] Petros Boufounos, Volkan Cevher, Anna C Gilbert, Yi Li, and Martin J Strauss. What's the Frequency, Kenneth?: Sublinear Fourier Sampling Off the Grid. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 61–72. Springer, 2012.

[BFJ+94] Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 253–262, 1994.

[BM96] Sonali Bagchi and Sanjit K Mitra. The nonuniform discrete fourier transform and its applications in filter design. i. 1-d. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(6):422–433, 1996.

[BM12] Sonali Bagchi and Sanjit K Mitra. *The nonuniform discrete Fourier transform and its applications in signal processing*, volume 463. Springer Science & Business Media, 2012.

[BOT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 301–309, 1988.

[Bou14]   Jean Bourgain. An improved estimate in the restricted isometry problem. In *Geometric aspects of functional analysis*, pages 65–70. Springer, 2014.

[CGV13]   Mahdi Cheraghchi, Venkatesan Guruswami, and Ameya Velingker. Restricted isometry of Fourier matrices and list decodability of random linear codes. *SIAM Journal on Computing*, 42(5):1888–1914, 2013.

[CI17]   Mahdi Cheraghchi and Piotr Indyk. Nearly optimal deterministic algorithm for sparse Walsh-Hadamard transform. *ACM Transactions on Algorithms (TALG)*, 13(3):1–36, 2017.

[CKPS16]   Xue Chen, Daniel M Kane, Eric Price, and Zhao Song. Fourier-sparse interpolation without a frequency gap. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 741–750. IEEE, 2016.

[CKSZ17]   Volkan Cevher, Michael Kapralov, Jonathan Scarlett, and Amir Zandieh. An adaptive sublinear-time block sparse Fourier transform. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 702–715, 2017.

[CRT06]   E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52:489–509, 2006.

[CT06]   Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006.

[Don06]   D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[FJ]   Matteo Frigo and Steven G. Johnson. FFTW: C subroutine library for computing the discrete fourier transform (DFT). `https://www.fftw.org/`.

[FR13]   Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Springer, 2013.

[Fri99]   Matteo Frigo. A fast Fourier transform compiler. In *Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation*, pages 169–180, 1999.

[FS03]   Jeffrey A Fessler and Bradley P Sutton. Nonuniform fast fourier transforms using min-max interpolation. *IEEE transactions on signal processing*, 51(2):560–574, 2003.

[GGI+02]   Anna C Gilbert, Sudipto Guha, Piotr Indyk, Shanmugavelayutham Muthukrishnan, and Martin Strauss. Near-optimal sparse Fourier representations via sampling. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 152–161, 2002.

[GHI+13]   Badih Ghazi, Haitham Hassanieh, Piotr Indyk, Dina Katabi, Eric Price, and Lixin Shi. Sample-optimal average-case sparse Fourier transform in two dimensions. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1258–1265. IEEE, 2013.

[GIKW19] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. *ACM Trans. Algorithms*, 15(2):23:1–23:35, 2019.

[GL89] Oded Goldreich and Leonid 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.

[GL04] Leslie Greengard and June-Yub Lee. Accelerating the nonuniform fast fourier transform. *SIAM review*, 46(3):443–454, 2004.

[GMS05] Anna C Gilbert, Shan Muthukrishnan, and Martin Strauss. Improved time bounds for near-optimal sparse Fourier representations. In *Wavelets XI*, volume 5914, page 59141A. International Society for Optics and Photonics, 2005.

[GR87] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[HIKP] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Sparse Fast Fourier Transform code (SFFT 1.0 and 2.0). `https://groups.csail.mit.edu/netmit/sFFT/code.html`.

[HIKP12a] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse Fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 563–578. ACM, 2012.

[HIKP12b] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse Fourier transform. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1183–1194. SIAM, 2012.

[HK15] Qingqing Huang and Sham M. Kakade. Super-resolution off the grid. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2665–2673, 2015.

[HR16] Ishay Haviv and Oded Regev. The restricted isometry property of subsampled Fourier matrices. In *27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 288–297. Association for Computing Machinery, 2016.

[HR17] Ishay Haviv and Oded Regev. The restricted isometry property of subsampled fourier matrices. In *Geometric aspects of functional analysis*, pages 163–179. Springer, 2017.

[IGS07] M. A. Iwen, A. Gilbert, and M. Strauss. Empirical Evaluation of a Sub-Linear Time Sparse DFT Algorithm. *Communications in Mathematical Sciences*, 5, 2007.

[IK14] Piotr Indyk and Michael Kapralov. Sample-optimal Fourier sampling in any constant dimension. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 514–523. IEEE, 2014.

[IKP14] Piotr Indyk, Michael Kapralov, and Eric Price. (Nearly) Sample-optimal sparse Fourier transform. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 480–499. SIAM, 2014.

[Iwe10]   Mark A Iwen. Combinatorial sublinear-time Fourier algorithms. *Foundations of Computational Mathematics*, 10(3):303–338, 2010.

[JENR15]   Nagaraj Thenkarai Janakiraman, Santosh K. Emmadi, Krishna R. Narayanan, and Kannan Ramchandran. Exploring connections between sparse fourier transform computation and decoding of product codes. In *53rd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2015, Allerton Park & Retreat Center, Monticello, IL, USA, September 29 - October 2, 2015*, pages 1366–1373. IEEE, 2015.

[JLS20]   Yaonan Jin, Daogao Liu, and Zhao Song. A robust multi-dimensional sparse Fourier transform in the continuous setting. *arXiv preprint arXiv:2005.06156*, 2020.

[Kap16]   Michael Kapralov. Sparse Fourier transform in any constant dimension with nearly-optimal sample complexity in sublinear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 264–277, 2016.

[Kap17]   Michael Kapralov. Sample efficient estimation and recovery in sparse FFT via isolation on average. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 651–662. Ieee, 2017.

[KM93]   Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[KVZ19]   Michael Kapralov, Ameya Velingker, and Amir Zandieh. Dimension-independent sparse Fourier transform. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2709–2728. SIAM, 2019.

[KY11]   Krzysztof Kazimierczuk and Vladislav YU. Accelerated nmr spectroscopy by using compressed sensing. *Angewandte Chemie International Edition*, 2011.

[LDSP08]   Michael Lustig, David L Donoho, Juan M Santos, and John M Pauly. Compressed sensing MRI. *IEEE signal processing magazine*, 25(2):72–82, 2008.

[LMN93]   N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM (JACM)*, 1993.

[Man94]   Y. Mansour. Learning Boolean Functions via the Fourier Transform. *Theoretical Advances in Neural Computation and Learning*, 1994.

[Man95]   Yishay Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM Journal on Computing*, 24(2):357–368, 1995.

[Moi15]   Ankur Moitra. Super-resolution, extremal functions and the condition number of vandermonde matrices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 821–830. ACM, 2015.

[MZIC17]   Sami Merhi, Ruochuan Zhang, Mark A Iwen, and Andrew Christlieb. A New Class of Fully Discrete Sparse Fourier Transforms: Faster Stable Implementations with Guarantees. *Journal of Fourier Analysis and Applications*, pages 1–34, 2017.

[NSW19]   Vasileios Nakos, Zhao Song, and Zhengyu Wang. (nearly) sample-optimal sparse fourier transform in any dimension; ripless and filterless. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1568–1577. IEEE, 2019.

[OHR19]   Frank Ong, Reinhard Heckel, and Kannan Ramchandran. A fast and robust paradigm for fourier compressed sensing based on coded sampling. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 5117–5121. IEEE, 2019.

[OPR15]   Frank Ong, Sameer Pawar, and Kannan Ramchandran. Fast and efficient sparse 2d discrete fourier transform using sparse-graph codes. *CoRR*, abs/1509.05849, 2015.

[PR13]   Sameer Pawar and Kannan Ramchandran. Computing a k-sparse n-length discrete fourier transform using at most 4k samples and o (k log k) complexity. In *2013 IEEE International Symposium on Information Theory*, pages 464–468. IEEE, 2013.

[PR14]   Sameer Pawar and Kannan Ramchandran. A robust R-FFAST framework for computing a k-sparse n-length DFT in o (k log n) sample complexity using sparse-graph codes. In *2014 IEEE International Symposium on Information Theory*, pages 1852–1856. IEEE, 2014.

[PS15]   Eric Price and Zhao Song. A robust sparse Fourier transform in the continuous setting. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 583–600. IEEE, 2015.

[PST01]   Daniel Potts, Gabriele Steidl, and Manfred Tasche. Fast fourier transforms for nonequispaced data: A tutorial. In *Modern sampling theory*, pages 247–270. Springer, 2001.

[Uma19]   Chris Umans. Fast generalized dfts for all finite groups. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 793–805. IEEE, 2019.

[Wil05]   Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

[Wol67]   J Wolf. Decoding of bose-chaudhuri-hocquenghem codes and prony's method for curve fitting (corresp.). *IEEE Transactions on Information Theory*, 13(4):608–608, 1967.