A Fast Hadamard Transform for Signals with Sub-linear Sparsity

Robin Scheibler, Saeid Haghighatshoar and Martin Vetterli
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
Email: {robin.scheibler, saeid.haghighatshoar, martin.vetterli}@epfl.ch

***Abstract*—A new iterative low complexity algorithm has been presented for computing the Walsh-Hadamard transform (WHT) of an N dimensional signal with a K-sparse WHT, where N is a power of two and $K = O(N^\alpha)$, scales sub-linearly in N for some $0 < \alpha < 1$. Assuming a random support model for the nonzero transform domain components, the algorithm reconstructs the WHT of the signal with a sample complexity $O(K \log_2(\frac{N}{K}))$, a computational complexity $O(K \log_2(K) \log_2(\frac{N}{K}))$ and with a very high probability asymptotically tending to 1.**

The approach is based on the subsampling (aliasing) property of the WHT, where by a carefully designed subsampling of the time domain signal, one can induce a suitable aliasing pattern in the transform domain. By treating the aliasing patterns as parity-check constraints and borrowing ideas from erasure correcting sparse-graph codes, the recovery of the nonzero spectral values has been formulated as a belief propagation (BP) algorithm (peeling decoding) over an sparse-graph code for the binary erasure channel (BEC). Tools from coding theory are used to analyze the asymptotic performance of the algorithm in the *"very sparse"* ($\alpha \in (0, \frac{1}{3}]$) and the *"less sparse"* regime ($\alpha \in (\frac{1}{3}, 1)$).

I. Introduction

The discrete Walsh-Hadamard transform (WHT) is a well-known signal processing tool with application in areas as disparate as image compression, designing spreading sequences in multi-user transmission in cellular networks (CDMA), coding, spectroscopy as well as compressed sensing [1]. Its recursive structure, similar to the fast Fourier transform (FFT) algorithm for computing the discrete Fourier transform (DFT), allows a fast computation with a complexity $O(N \log_2(N))$ in the dimension of the signal N [2], [3].

A number of recent publications have addressed the particular problem of computing the DFT of an N dimensional signal which is K-sparse in the frequency domain [4], [5], [6]. In particular, it has been shown that the already known computational complexity $O(N \log_2(N))$ belonging to the FFT algorithm can be strictly improved. Such algorithms are generally known as *sparse* FFT (sFFT) algorithms. The authors in [7] by extending the results of [6], gave a very low complexity algorithm for computing the 2D-DFT of a $\sqrt{N} \times \sqrt{N}$ signal. In a similar line of work, based on the subsampling property of the DFT in the time domain resulting in aliasing in the frequency domain, the authors in [8], [9] developed a novel low complexity iterative algorithm to recover the non-zero frequency elements using ideas from sparse-graph codes.

In this paper, we first develop some of the useful properties of the WHT, specially the subsampling and the modulation property that are of vital importance for developing the recovery algorithm. In particular, we show that subsampling in the time domain allows to induce a well-designed aliasing pattern over the transform domain components. In other words, it is possible to obtain a linear combination of a controlled collection of transform domain components (aliasing), which creates interference between the nonzero components if more than one of them are involved in the induced linear combination. Similar to [9] and borrowing ideas from sparse-graph codes, we construct a bipartite graph by considering the nonzero values in the transform domain as variable nodes and interpreting any induced aliasing pattern as a parity check constraint over the variables in the graph. We analyze the structure of the resulting graph assuming a random support model for the nonzero coefficients in the transform domain. Moreover, we give an iterative peeling decoder to recover those nonzero components. In a nutshell, our proposed sparse fast Hadamard transform (SparseFHT) consists of a set of deterministic linear hash functions (explicitly constructed) and an iterative peeling decoder that uses the hash outputs to recover the nonzero transform domain variables. It recovers the K-sparse WHT of the signal in sample complexity (number of time domain samples used) $O(K \log_2(\frac{N}{K}))$, total computational complexity $O(K \log_2(K) \log_2(\frac{N}{K}))$ and with a high probability approaching 1 asymptotically.

Notations and Preliminaries: For an integer m, the set of all integers $\{0, 1, \ldots, m-1\}$ is denoted by $[m]$. We use the small letter x for the time domain and the capital letter X for the transform domain signal. For an N dimensional real-valued vector v, with $N = 2^n$, the i-th component of v is interchangeably represented by v_i or $v_{i_0, i_1, \ldots, i_{n-1}}$, where $i_0, i_1, \ldots, i_{n-1}$ denotes the binary expansion of i with i_0 and i_{n-1} being the least and the most significant bits. $\mathbb{F}_2$ denotes the binary field consisting of $\{0, 1\}$ with summation and multiplication modulo 2. We also denote by $\mathbb{F}_2^n$ the space of all n dimensional vectors with the addition of the vectors done component wise over $\mathbb{F}_2$. The inner product of two binary vectors $u, v \in \mathbb{F}_2^n$ is defined by $\langle u\,,\, v \rangle = \sum_{t=0}^{n-1} u_t v_t$ with arithmetic over $\mathbb{F}_2$.

The research of Robin Scheibler was supported by ERC Advanced Investigators Grant: Sparse Sampling: Theory, Algorithms and Applications SPARSAM no. 247006

II. Main Results

For a signal $X \in \mathbb{R}^N$, the support of X is defined as $\mathrm{supp}(X) = \{i \in [N] : X_i \neq 0\}$. The signal X is called K-sparse if $|\mathrm{supp}(X)| = K$, where for a set $A \subset [N]$, $|A|$ denotes the cardinality or the number of elements of A. For a collection of N dimensional signals $\mathcal{S}_N \subset \mathbb{R}^N$, the sparsity of $\mathcal{S}_N$ is defined as $K_N = \max_{X \in \mathcal{S}_N} |\mathrm{supp}(X)|$.

Definition 1. A class of signals of increasing dimension $\{\mathcal{S}_N\}_{N=1}^{\infty}$ has sub-linear sparsity if there is $0 < \alpha < 1$ and some $N_0 \in \mathbb{N}$ such that for all $N > N_0$, $K_N \leq N^{\alpha}$. The value α is called the sparsity index of the class.

Theorem 1. Let $0 < \alpha < 1$, $N = 2^n$ and $K = N^{\alpha}$. Suppose $x \in \mathbb{R}^N$ is a time domain signal with a WHT $X \in \mathbb{R}^N$. Assume that X is a K-sparse signal in a class of signals with sparsity index α whose support is uniformly randomly selected among all possible $\binom{N}{K}$ subsets of $[N]$ of size K. For any value of α, there is an algorithm that can compute X and has the following properties:

1) **Sample complexity:** The algorithm uses $CK \log_2(\frac{N}{K})$ time domain samples of the signal x. C is a function of α and $C \leq (\frac{1}{\alpha} \vee \frac{1}{1-\alpha}) + 1$ for all $\alpha \in (0,1)$, where for $a, b \in \mathbb{R}_+$, $a \vee b$ denotes the maximum of a and b.
2) **Computational complexity:** The total number of operations in order to successfully decode all the nonzero spectral components or announce a decoding failure is $O(CK \log_2(K) \log_2(\frac{N}{K}))$.
3) **Success probability:** The algorithm correctly computes the K-sparse WHT X with a very high probability asymptotically approaching 1 as N tends to infinity, where the probability is taken over all random selections of the support of X.

III. Walsh-Hadamard Transform and its Properties

Let x be an $N = 2^n$ dimensional signal indexed with elements $m \in \mathbb{F}_2^n$. The N dimensional WHT of the signal x is defined by

$$X_k = \frac{1}{\sqrt{N}} \sum_{m \in \mathbb{F}_2^n} (-1)^{\langle k\,,\, m \rangle} x_m,$$

where $k \in \mathbb{F}_2^n$ denotes the corresponding binary index of the transform domain component. Throughout the paper, borrowing some terminology from the DFT, we call transform domain samples $X_k, k \in \mathbb{F}_2^n$ frequency or spectral domain components of the time domain signal x.

A. Basic Properties

In this section, we give some of the basic properties of the WHT without a proof.

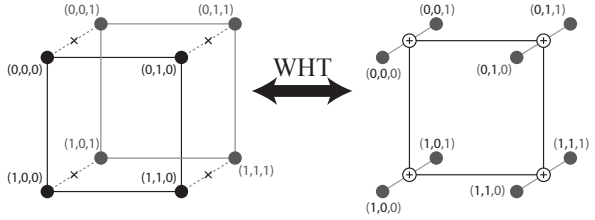

Fig. 1: Illustration of the Downsampling property on a hypercube.

Property 1 (Shift/Modulation)**.** Let X_k be the WHT of the signal x_m and let $p \in \mathbb{F}_2^n$. Then

$$x_{m+p} \quad \overset{\text{WHT}}{\longleftrightarrow} \quad (-1)^{\langle p\,,\, k \rangle} X_k.$$

Property 2 (Permutation)**.** Let $\Sigma \in \mathrm{GL}(n, \mathbb{F}_2)$. Assume that X_k is the WHT of the time domain signal x_m. Then

$$x_{\Sigma m} \quad \overset{\text{WHT}}{\longleftrightarrow} \quad X_{\Sigma^{-T} k}.$$

For an $N = 2^n$ dimensional signal x with the binary indexing $x_{m_0, m_1, \ldots, m_{n-1}}$, the subsampling along dimension $i \in [n]$ is defined by freezing the i-th component of the index to either 0 or 1. For example, $x_{0, m_1, \ldots, m_{n-1}}$ is a 2^{n-1} dimensional signal obtained by subsampling the signal x_m along the first index.

Property 3 (Downsampling/Aliasing)**.** Suppose that x is a vector of dimension $N = 2^n$ indexed by the elements of $\mathbb{F}_2^n$ and assume that $B = 2^b$, where $b \in \mathbb{N}$ and $b < n$. Let

$$\Psi_b = \begin{bmatrix} \mathbf{0}_{b \times (n-b)} & I_{b \times b} \end{bmatrix}^T, \tag{1}$$

be the subsampling matrix freezing the first $n-b$ first indices to 0. If X_k is the WHT of x, then

$$x_{\Psi_b m} \quad \overset{\text{WHT}}{\longleftrightarrow} \quad \sqrt{\frac{B}{N}} \sum_{i \in \mathcal{N}(\Psi_b^T)} X_{\Psi_b k + i}, \tag{2}$$

where $x_{\Psi_b m}$ is a B dimensional signal labelled with $m \in \mathbb{F}_2^b$.

Remark 1. $\mathbb{F}_2^n$ can be visualized as the vertices of the n-dimensional hypercube. The downsampling property implies that downsampling along some of the dimensions in the time domain is equivalent to summing up all of the spectral components along *the same dimensions*. This is illustrated in Fig. 1 for dimension $n = 3$.

IV. Hadamard Hashing Algorithm

By applying the basic properties of the WHT, one can design suitable hash functions in the spectral domain. The main idea is that one does not need to have access to the spectral values and the output of all hash functions can be simply computed by low complexity operations on the time domain samples of the signal.

Proposition 1 (Hashing)**.** Assume that $\Sigma \in \mathrm{GL}(n, \mathbb{F}_2)$ and $p \in \mathbb{F}_2^n$. Let $N = 2^n$, $b \in \mathbb{N}$, $B = 2^b$ and let $m, k \in \mathbb{F}_2^b$ denote the time and frequency indices of a B dimensional signal $u_{\Sigma,p}(m) = \sqrt{\frac{N}{B}}\, x_{\Sigma \Psi_b m + p}$ and its WHT. Then, the length B WHT of $u_{\Sigma,p}$ is given by

$$U_{\Sigma,p}(k) = \sum_{i \in \mathbb{F}_2^n \,|\, h_\Sigma(i) = k} (-1)^{\langle p\,,\, i \rangle} X_i,$$

where Ψ_b is as in (1) and h_Σ is the index hashing function defined by

$$h_\Sigma(i) = \Psi_b^T \Sigma^T\, i. \tag{3}$$

Proof simply follows from the Property 1, 2, and 3. Based on Proposition 1, we give Algorithm 1 which computes the hashed Hadamard spectrum.

Algorithm 1 FastHadamardHashing(x, N, Σ, p, B)

Require: Signal x of dimension $N = 2^n$, Σ, p and given number of output bins $B = 2^b$ in a hash.

Ensure: U contains the hashed Hadamard spectrum of x.

$u_m = x_{\Sigma \Psi_b m + p}$, for $m \in \mathbb{F}_2^b$.

$U = \sqrt{\frac{N}{B}}\, \mathrm{FastHadamard}(u_m, B)$.

A. Properties of Hadamard Hashing

In this part, we review some of the nice properties of the hashing algorithm that are crucial for developing the iterative peeling decoding algorithm to recover the nonzero spectral values. We explain how it is possible to identify collision between the nonzero spectral coefficients that are hashed to the same bin and also to estimate the support of non-colliding components.

Consider $U_{\Sigma,p}(k)$ for two cases: $p = 0$ and some $p \neq 0$. It is easy to see that in the former $U_{\Sigma,p}(k)$ is obtained by summing all of the spectral variables hashed to the bin k whereas in the latter the same variables are added together with a weight $(-1)^{\langle p\,,\, i \rangle}$. Let us define the following ratio test $r_{\Sigma,p}(k) = \frac{U_{\Sigma,p}(k)}{U_{\Sigma,0}(k)}$. When the sum in $U_{\Sigma,p}(k)$ contains only one non-zero component, it is easy to see that $|r_{\Sigma,p}(k)| = 1$ for *'any value'* of p. However, if there is more than one component in the sum, under a very mild assumption on the the non-zero coefficients of the spectrum (i.e. they are jointly sampled from a continuous distribution), one can show that $|r_{\Sigma,p}(k)| \neq 1$ for some of the values of p. In fact, $n - b$ well-chosen values of p is sufficient to detect whether there is one, or more than one non-zero components in the sum.

When there is only one $X_{i'} \neq 0$ hashed to the bin k ($h_\Sigma(i') = k$), the result of the ratio test is precisely 1 or -1, depending on the value of the inner product between i' and p. In particular, we have $\langle p\,,\, i' \rangle = \mathbb{1}_{\{r_{\Sigma,p}(k) < 0\}}$, where $\mathbb{1}_{\{t<0\}} = 1$ if $t < 0$, and zero otherwise. Moreover, if for $n - b$ well-chosen values of p, the ratio test results in 1 or -1, by some extra effort, it is even possible to identify the position of the non-zero component. We formalize this result in the following proposition.

Proposition 2 (Collision detection / Support estimation)**.** Let $\Sigma \in \mathrm{GL}(n, \mathbb{F}_2)$ and let $\sigma_i, i \in [n]$ denote its columns.

1) If for all $d \in [n - b]$, $|r_{\Sigma,\sigma_d}(k)| = 1$ then almost surely there is only one nonzero spectral value in the bin indexed by k. Moreover, if we define

$$\hat{v}_d = \begin{cases} \mathbb{1}_{\{r_{\Sigma,\sigma_d}(k) < 0\}} & d \in [n-b], \\ 0 & \text{otherwise}, \end{cases}$$

then the index of the unique nonzero coefficient is given by

$$i = \Sigma^{-T}(\Psi_b\, k + \hat{v}). \tag{4}$$

2) If there exists a $d \in [n - b]$ such that $|r_{\Sigma,\sigma_d}(k)| \neq 1$ then the bin k contains more than one nonzero coefficient.

Proof: We only give a sketch of proof. Let $\mathcal{I}_k = \{i \,|\, h_\Sigma(i) = k\}$ be the set of variable indices hashed to the bin k and without loss of generality, assume that $\sum_{i \in \mathcal{I}_k} X_i = 1$. Such $\{X_i\}_{i \in \mathcal{I}_k}$ is a solution of

$$\begin{bmatrix} 1 & \cdots & 1 \\ (-1)^{\langle \sigma_1\,,\, i_1 \rangle} & \cdots & (-1)^{\left\langle \sigma_1\,,\, i_{\frac{N}{B}} \right\rangle} \\ \vdots & \ddots & \vdots \\ (-1)^{\langle \sigma_{n-b}\,,\, i_1 \rangle} & \cdots & (-1)^{\left\langle \sigma_{n-b}\,,\, i_{\frac{N}{B}} \right\rangle} \end{bmatrix} \begin{bmatrix} X_{i_1} \\ \vdots \\ X_{i_{\frac{N}{B}}} \end{bmatrix} = \begin{bmatrix} 1 \\ \pm 1 \\ \vdots \\ \pm 1 \end{bmatrix},$$

where the specific $\pm$ sign on the right hand side vector is obtained from the ratio test. The left hand side matrix in the expression above has dimension $(n - b + 1) \times 2^{n-b}$. As $\sigma_1, \ldots, \sigma_{n-b}$ form a basis for $\mathcal{I}_k$, all the columns are different and omitting the top row, they contain all exhaustive list of 2^{n-b} possible vectors with ± 1 elements. Thus, the right vector is always one of the columns of the matrix, and the position of the nonzero element can be uniquely identified. ∎

V. Sparse Fast Hadamard Transform

In this section, we give a brief overview of the main idea of Sparse Fast Hadamard Transform (SparseFHT). In particular, we explain the peeling decoder which recovers the nonzero spectral components and analyze its computational complexity.

A. Explanation of the Algorithm

Assume that x is an $N = 2^n$ dimensional signal with a K-sparse WHT X. As $H_N^{-1} = H_N$, taking the inner product of the vector X with the i-th row of the Hadamard matrix H_N gives the time domain sample x_i. Using the terminology of Coding theory, it is possible to consider the

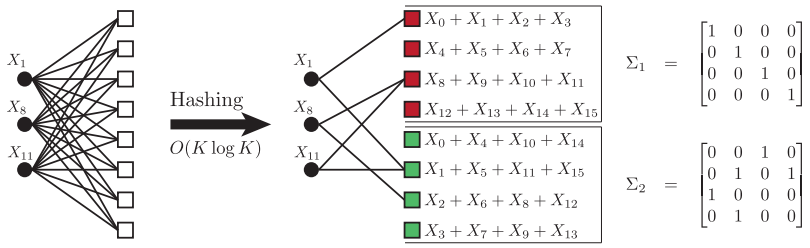

Fig. 2: On the left, bipartite graph representation of the WHT for $N = 8$ and $K = 3$. On the right, the underlying bipartite graph after applying $C = 2$ different hashing produced by plugging Σ_1, Σ_2 in (3) with $B = 4$. The variable nodes (•) are the non-zero spectral values to be recovered. The white check nodes (□) are the original time-domain samples. The colored squares are new check nodes after applying Algorithm 1.

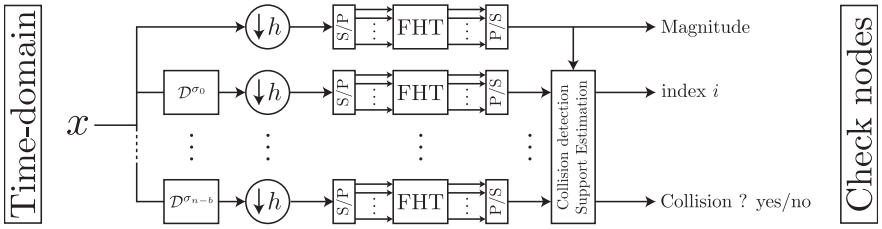

Fig. 3: A block diagram of the SparseFHT algorithm in the time domain. The downsampling plus small size low complexity FHT blocks compute different hash outputs. Delay blocks denote an index shift by σ_i before hashing. The collision detection/support estimation block implements Proposition 2 to identify if there is a collision. Index i is the position of the only nonzero spectral variable in a hash bin and it is not not valid when there is a collision.

spectral components X as variables nodes (information bits in coding theory), where the inner product of the i-th row of H_N is like a parity check constraint on X. Thus, WHT can be imagined as a code over a bipartite graph and the recovery of the nonzero spectral values can be interpreted as a decoding problem over this bipartite graph. The bipartite graph for WHT is a complete graph because any variable node is connected to all of the check nodes as depicted in the left part of Fig. 2 for the nonzero spectral components $\{X_1, X_8, X_{11}\}$.

For codes on bipartite graphs, there is a collection of low complexity belief propagation algorithms performing well under the sparsity of the underlying bipartite graph. Unfortunately, the graph corresponding to WHT is dense, and probably not suitable for any of these belief propagation algorithms to succeed. As explained in Section IV, by subsampling the time domain signal, it is possible to hash the spectral components in different bins as depicted for the same signal X in the right part of Fig. 2. The advantage of the hashing must be clear from this picture. The idea is that hashing *'sparsifies'* the underlying graph. It is also important to notice that in the bipartite graph induced by hashing, one can obtain all of parity check values (hash outputs) by using low complexity operations on a small subset of time domain samples as explained in Proposition 1 and Algorithm 1.

We propose the following iterative algorithm to recover the nonzero spectral variables over the bipartite graph induced by hashing. The algorithm first tries to find a degree one check node. Using the terminology of [9], we call such a check node a *singleton*. Notice that from Proposition 2, this is doable. Further, the algorithm is able to find the position and the value of the corresponding nonzero component, thus the algorithm can subtract (peel off) this variable from all other check nodes that are connected to it. In particular, after this operation the corresponding singleton check node gets value zero. Equivalently, we can update the underlying graph by removing the mentioned variable node from the graph along with all the edges connected to it. This creates an isolated (degree zero) check node called a *zeroton*. Notice that by removing some of the edges from the graph, the degree of the associated checks decreases by one, thus there is a chance to find another singleton.

The algorithm proceeds to peel off a singleton at a time until all of the check nodes are *zeroton* (decoding succeeds) or all of the remaining check nodes have degree greater than one (we call them *multiton*), where the algorithm fails to identify all of the nonzero spectral values.

B. Complexity Analysis

Figure 3 shows a full block diagram of the SparseFHT algorithm. Using this block diagram, it is possible to prove part 1 and 2 of Theorem 1 about the sample and the computational complexity of the SparseFHT algorithm. The last part of Theorem 1, regarding the success probability of

the algorithm, will be proved in Section VI and VII.

Computational Complexity: As we will explain in Section VI and VII, depending on the sparsity index of the signal α, we will use C different hash blocks, where $C \leq (\frac{1}{\alpha} \vee \frac{1}{1-\alpha}) + 1$, each with $B = 2^b$ different output bins. We always select $B = K$ to keep the average nonzero components per bin $\beta = \frac{K}{B}$ equal to 1. This implies that computing the hash outputs via an FHT block of size B needs $B\log_2(B) = K\log_2(K)$ operations. Moreover, we need to compute any hash output with $n - b = \log_2(\frac{N}{B})$ different shifts in order to do collision detection/support estimation, thus the computational cost per each hash is $K\log_2(K)\log_2(\frac{N}{K})$. As we need to compute C different hash blocks, the total computational complexity per each iteration will be $CK\log_2(K)\log_2(\frac{N}{K})$. We will explain later that the algorithm terminates in a fixed number of iterations independent of the value of α and the dimension of the signal N. Therefore, the total computational complexity of the algorithm will be $O(CK\log_2(K)\log_2(\frac{N}{K}))$.

Sample Complexity: Assuming $K = B$, computing each hash with $n-b$ different shifts needs $K\log_2(\frac{N}{K})$ time domain samples. Therefore, the total sample complexity will be $CK\log_2(\frac{N}{K})$.

VI. Analysis of the very Sparse Regime

For the very sparse regime $\alpha \in (0, \frac{1}{3}]$, we show that assuming a random support model for nonzero spectral components and a careful design of hash functions, it is possible to obtain a random bipartite graph with variable nodes corresponding to nonzero spectral components and with check nodes corresponding to output of hash functions. Running the peeling decoder to recover the spectral components is also equivalent to the belief propagation (BP) decoding over a binary erasure channel (BEC). We also show that the error (decoding failure) probability can be asymptotically characterized by a *'Density Evolution'* (DE) equation allowing a perfect analysis of the peeling decoder. We use the following steps to rigorously analyze the performance of the decoder in this regime:

1) We explain construction of suitable hash functions depending on the value of $\alpha \in (0, \frac{1}{3}]$.
2) We rigorously analyze the structure of the induced bipartite graph and prove that it is a fully random left regular bipartite graph.
3) We use a Density Evolution equation to asymptotically track the ratio of unpeeled edges during the runtime of the algorithm.
4) An expander argument is applied to show that if the decoder peels a ratio of the edges very close to 1, it can continue to peel off all the remaining edges with very high probability.

A. Hash Construction

Consider those values of $\alpha \in (0, \frac{1}{3}]$ equal to $\frac{1}{C}$ for some integer $C \geq 3$. For $\alpha = \frac{1}{C}$, we will consider C different hash functions as follows. Let x be an N dimensional time domain signal with a WHT X, where $N = 2^n$ and let $b = \frac{n}{C}$. As we explained before, the components of the vector X can be labelled by n dimensional binary vector from $\mathbb{F}_2^n$. Let

$$\Psi_b^{(i)} = [\mathbf{0}_{b\times ib}\ I_{b\times b}\ \mathbf{0}_{b\times(n-(i+1)b)}]^T,$$

where $I_{b\times b}$ is the identity matrix of order b and let $p \in \mathbb{F}_2^n$ be an arbitrary vector. Then the i-th subsampled signal in the time domain is a $B = 2^b$ dimensional vector given by $x_{{\Psi_b^{(i)}}^T m+p}$, where $m \in \mathbb{F}_2^b$. The WHT of the subsampled signal, similar to Equation (2), is given by

$$x_{\Psi_b^{(i)}m+p} \quad \overset{\text{WHT}}{\longleftrightarrow} \quad \sqrt{\frac{B}{N}} \sum_{j\in\mathcal{N}\left({\Psi_b^{(i)}}^T\right)} X_{\Psi_b^{(i)}k+j}, \qquad (5)$$

which shows that for a given $k \in \mathbb{F}_2^b$, all of the spectral components with indices $\Psi_b^{(i)}k+j$ with $j \in \mathcal{N}\left({\Psi_b^{(i)}}^T\right)$ are mapped to the bin number k in the resulting hash function. Equivalent to the C different subsampling operators, we can consider functions $h_i, i \in [C]$ where $h_i(X_0^{n-1}) = (X_{i\,b}, X_{i\,b+1}, \ldots, X_{i\,b+b-1})$. The important point is that with this construction, the outputs of different h_i depend on non overlapping portions of the labeling binary indices. Moreover, from Equation (5) every spectral component X_0^{n-1} is hashed to the bin labelled with $h_i(X_0^{n-1}) \in \mathbb{F}_2^b$ in hash i.

B. Ensemble of Graphs Generated by Hashing

By the hashing scheme that we explained, there is a one-to-one relation between an spectral element X and its bin indices in different hashes $(h_0(X), h_1(X), \ldots, h_{C-1}(X))$. Assume that $X_1, X_2, \ldots, X_K$ are K independent uniformly distributed variables in $\mathbb{F}_2^n$ denoting the position of nonzero spectral components. For these K variables and hash functions h_i, we can associate a bipartite graph as follows. We consider K variable nodes corresponding to X_1^K and C different set of check nodes $S_0, S_1, \ldots, S_{C-1}$ each of size $B = 2^b$. The check nodes in each S_i are labelled by elements of $\mathbb{F}_2^b$. For each variable X_i, we consider C different edges connecting X_i to check nodes labelled with $h_j(X_i) \in S_j$, $j \in [C]$. One can simply check that in the resulting bipartite graph, every variable node selects its hash bins uniformly among all possible bins, independent of the bins selected in the other hashes and all the hash bins of the other variables. We denote by $\mathcal{G}(K, B, C)$ the ensemble of all graphs generated in this way.

1) Edge Degree Distribution Polynomial: For a graph from the ensemble $\mathcal{G}(K, B, C)$, let $\beta = \frac{K}{B}$ denote the average number of nonzero components per a hash bin ($\beta = 1$ in our case). As the resulting bipartite graph is left regular, all of the variable nodes have degree C whereas for a specific check node the degree is random and depends on the graph realization.

Proposition 3. Let $\mathcal{G}(K, B, C)$ be the random graph ensemble as before with $\beta = \frac{K}{B}$ fixed. Asymptotically as N tends to infinity the check degree converges to a Poisson random variable with average β.

For a bipartite graph, the edge degree distribution polynomial is defined by $\rho(\alpha) = \sum_{i=1}^{\infty} \rho_i \alpha^{i-1}$ and $\lambda(\alpha) = \sum_{i=1}^{\infty} \lambda_i \alpha^{i-1}$, where ρ_i (λ_i) is the ratio of all edges that are connected to a check node (variable node) of degree i.

Proposition 4. Let $\mathcal{G}$ be a random bipartite graph from the ensemble $\mathcal{G}(K, B, C)$ with $\beta = \frac{K}{B}$. Then, $\lambda(\alpha) = \alpha^{C-1}$ and $\rho(\alpha)$ converges to $e^{-\beta(1-\alpha)}$ as N tends to infinity.

C. Performance Analysis of the Peeling Decoder

In this section, we analyze the the performance of the peeling decoder and give further intuition about why it must work very well in the very sparse regime. This method is based on the analysis of BP decoder over sparse locally tree-like graphs. The analysis is very similar to the analysis of the peeling decoder for the recovery of the nonzero frequency components in [9]. Consider a specific edge $e = (v, c)$ in a graph from the ensemble $\mathcal{G}(K, B, C)$. Consider a directed neighborhood of this edge of depth ℓ consisting of all edges, check and variable nodes that can be reached from this edge up to depth 2ℓ. At the first stage, it is easy to see that this edge is peeled off from the graph if all the edges (c, v') connected to the check node c are peeled off because in that case check c will be a singleton allowing to decode the variable v and to peel off the edge e. This has been depicted in Figure 4.

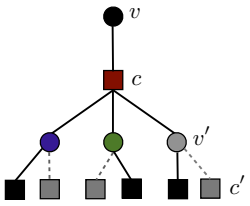

Fig. 4: Tree-like neighborhood an an edge $e = (v, c)$. Dashed lines show the edges removed before iteration t.

One can proceed in this way in the directed neighborhood to find the condition under which the variable v' connected to c can be peeled off and so on. Assuming that the directed neighborhood is a tree, all of the messages that are passed from the leaves up to the the head edge e are independent from one another. Let p_ℓ be the probability that edge e is peeled off depending on the information received from the directed tree neighborhood up to depth ℓ. A simple analysis similar to [9], gives the following recursion

$$p_{j+1} = \lambda(1 - \rho(1 - p_j)), \quad j \in [\ell], \tag{6}$$

where λ and ρ are the edge degree polynomials of the ensemble $\mathcal{G}$. This iteration shows the progress of the peeling decoder in recovering unknown variable nodes. In [9], it was proved that for any specific edge e, asymptotically with very high probability the directed neighborhood of e up to any fixed depth ℓ is a tree. In particular, starting from a left regular graph $\mathcal{G}$ from $\mathcal{G}(K, B, C)$ with KC edges, after ℓ steps of decoding, the average number of unpeeled edges is concentrated around KCp_ℓ. A martingale argument was applied in [9] to show that not only the average of the unpeeled edges is approximately KCp_ℓ but also with very high probability the number of those edges is well concentrated around KCp_ℓ.

Equation (6) is known as density evolution equation. Starting from $p_0 = 1$, this equation fully predicts the performance of the peeling decoding over the ensemble $\mathcal{G}$. Figure 5 shows a typical behavior of this iterative equation for different values of the parameter $\beta = \frac{K}{B}$.

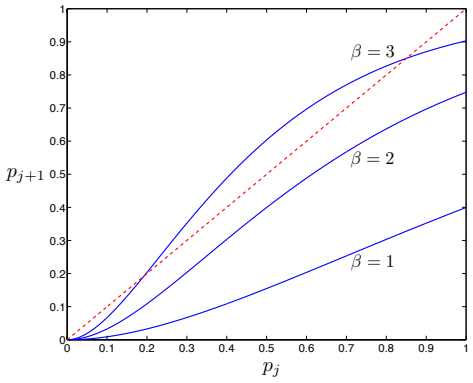

Fig. 5: Density Evolution equation for $C = 3$.

For small values of β, 0 is the only fixed point of this equation which implies that asymptotically the decoder can recover a very close to 1 ratio of the variables. However, for large values of β, i.e. $\beta \gtrsim 2.44$ for $C = 3$, this equation has a fixed point greater than 0. The largest fixed point is the place where the peeling decoder stops and can not proceed to decode the remaining variables. This analysis only guarantees that for any $\epsilon \in (0, 1)$, asymptotically as N tends to infinity, $1 - \epsilon$ ratio of the edges are peeled off. We use a combinatorial argument that guarantees the full recovery of all the remaining variables with high probability. The proof follows from a slight variation of Lemma 1 in [10].

Proposition 5. Let $\mathcal{G}$ be a graph from the ensemble $\mathcal{G}(K, B, C)$ with $C \geq 3$. There is some $\eta \in (0, 1)$ such that with probability at least $1 - O(\frac{1}{K^{3(C/2-1)}})$, the recovery process restricted to the subgraph induced by any η-fraction of the left nodes terminates successfully.

VII. Analysis of the Less Sparse Regime

Similar to the very sparse regime $\alpha \in (0, \frac{1}{3}]$, we use the following steps to analyze the performance of the algorithm:

1) Constructing suitable hash functions,
2) Representing hashing of nonzero spectral values by an equivalent bipartite graph,
3) Analyzing the performance of the peeling decoder over the resulting bipartite graph.

For simplicity, we consider the case where $\alpha = 1 - \frac{1}{C}$ for some integer $C \geq 3$.

A. Hash Construction

Let $\alpha = 1 - \frac{1}{C}$ for some integer $C \geq 3$. Suppose x is an $N = 2^n$ dimensional signal and let X denote its WHT. Assume that the components of X are labelled by binary indices $X_0^{n-1} \in \mathbb{F}_2^n$. Let $t = \frac{n}{C}$ and let us divide the set of n binary indices X_0^{n-1} into C non-intersecting subsets $r_0, r_1, \ldots, r_{C-1}$, where $r_i = X_{it}^{(i+1)t-1}$. It is clear that there is a one-to-one relation between every binary vector $X_0^{n-1} \in \mathbb{F}_2^n$ and its representation $(r_0, r_1, \ldots, r_{C-1})$. We construct C different hash function $h_i, i \in [C]$ by selecting C different subset of $(r_0, r_1, \ldots, r_{C-1})$ each of size $C-1$ and appending them together. For example,

$$h_1(X_0^{n-1}) = (r_0, r_1, \ldots, r_{C-2}) = X_0^{(C-1)t-1},$$

and the hash output is obtained by appending $C-1$ first $r_i, i \in [C]$. One can simply check that $h_i, i \in [C]$ are linear surjective functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2^b$, where $b = (C-1)t$. In particular, the range of each hash consists of $B = 2^b$ different elements of $\mathbb{F}_2^b$. With this construction, the average number of nonzero elements per bin in every hash is kept at $\beta = \frac{K}{B} = 1$ and the complexity of computing all the hashes along with their $n - b$ shifts, which are necessary for collision detection/support estimation, is $CK \log_2(K) \log_2(\frac{N}{K})$. The sample complexity can also be easily checked to be $CK \log_2(\frac{N}{K})$.

B. Bipartite Graph Representation

Similar to the very sparse regime, we can assign a bipartite graph with the K variable nodes corresponding to nonzero spectral components and with CB check nodes corresponding to different bins of all the hashes. In particular, we consider C different set of check nodes $S_1, S_2, \ldots, S_C$ each containing B nodes labelled with the elements of $\mathbb{F}_2^b$ and a specific nonzero spectral component labelled with X_0^{n-1} is connected to nodes $s_i \in S_i$ whose binary label is $h_i(X_0^{n-1})$. The difference with the less sparse case is that the selection of the neighbor checks in different hashes is not completely random anymore. To explain more, let us assume that $\alpha = \frac{2}{3}$, thus $C = 3$. Assume that for a nonzero spectral variable labelled with X_0^{n-1}, r_i denotes $X_{it}^{(i+1)t-1}$, where $t = \frac{n}{C}$. In this case, this variable is connected to bins labelled with (r_0, r_1), (r_1, r_2) and (r_0, r_2) in 3 different hashes as depicted in Figure 6.

Assuming that X_0^{n-1} is selected uniformly randomly from $\mathbb{F}_2^n$, the bin number is each hash, i.e. (r_0, r_1) in the first hash, is selected uniformly randomly among all possible bins.

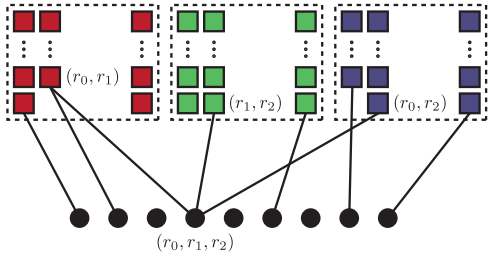

Fig. 6: Bipartite graph representation for $\alpha = \frac{2}{3}$, $C = 3$.

However, it is easily seen that the joint selection of bins is not completely random among different hashes. In other words, the associated bins in different hashes are not independent from one another. However, assuming the random support model, where K variables V_1^K are selected independently as the position of nonzero spectral variables, the bin association for different variables V_i is still done independently.

C. Performance Analysis of the Peeling Decoder

An analysis based on the DE for the BP algorithm can still be applied to track the ratio of the unpeeled edges during the algorithm run. In other words, setting $p_0 = 1$ and

$$p_{j+1} = \lambda(1 - \rho(1 - p_j)), \quad j \in [\ell],$$

as in (6) with λ and ρ being the edge degree polynomials of the underlying bipartite graph, it is still possible to show that after ℓ steps of decoding the average number of unpeeled edges is approximately KCp_ℓ. A martingale argument similar to [9] can be applied to show that the number of remaining edges is also well concentrated around its average. Another argument is necessary to show that if the peeling decoder decodes a majority of the variables, it can proceed to decode all of them with very high probability. To formulate this, we define the concept of a trapping set for the peeling decoder.

Definition 2. Let $\alpha = 1 - \frac{1}{C}$ for some integer $C \geq 3$ and let $h_i, i \in [C]$ be a set of hash functions as explained before. A subset of variables $T \subset \mathbb{F}_2^n$ is called a trapping set for the peeling decoder if for any $v \in T$ and for any $i \in [C]$, there is another $v_i \in T$, $v \neq v_i$ such that $h_i(v) = h_i(v_i)$, thus colliding with v in the i-th hash.

Let X be a spectral variable in the trapping set with the corresponding binary representation X_0^{n-1} and assume that $C = 3$. We can equivalently represent this variable with (r_0, r_1, r_2), where $r_i = X_{it}^{(i+1)t-1}$ with $t = \frac{n}{C}$. We can consider a three dimensional lattice whose i-th axis is labelled by all possible values of r_i. In this space, A set T is a trapping set if and only if for any $(r_0, r_1, r_2) \in T$ there are three other elements (r_0', r_1, r_2), (r_0, r_1', r_2) and (r_0, r_1, r_2') in T that can be reached from (r_0, r_1, r_2) by moving along exactly one of the axes. In general, for $C \geq 3$, the set of all C-tuples $(r_0, r_1, \ldots, r_{C-1})$ is a C dimensional lattice.

We denote this lattice by L. The intersection of this lattice by the hyperplane $R_i = r_i$ is a $(C-1)$ dimensional lattice defined by

$$L(R_i = r_i) = \{(r_0, \ldots, r_{i-1}, r_{i+1}, \ldots, r_{C-1}) : (r_0, r_1, \ldots, r_{i-1}, r_i, r_{i+1}, \ldots, r_{C-1}) \in L\}.$$

Similarly, for $S \subset L$, we have the following definition

$$S(R_i = r_i) = \{(r_0, \ldots, r_{i-1}, r_{i+1}, \ldots, r_{C-1}) : (r_0, r_1, \ldots, r_{i-1}, r_i, r_{i+1}, \ldots, r_{C-1}) \in S\}.$$

Obviously, $S(R_i = r_i) \subset L(R_i = r_i)$. We have the following proposition whose proof simply follows from the definition of the trapping set.

Proposition 6. Assume that T is a trapping set for the C dimensional lattice representation L of the nonzero spectral domain variables as explained before. Then for any r_i on the R_i axis, $T(R_i = r_i)$ is either empty or a trapping set for the $(C-1)$ dimensional lattice $L(R_i = r_i)$.

Proposition 7. The size of the trapping set for a C dimensional lattice is at least 2^C.

Proof: We give a simple proof using the induction on C. For $C = 1$, we have a one dimensional lattice along a line labelled with r_0. In this case, there must be at least two variables on the line to build a trapping set. Consider a trapping set T of dimension C. There are at least two points $(r_0, r_1, \ldots, r_{C-1})$ and $(r'_0, r_1, \ldots, r_{C-1})$ in T. By Proposition 6, $T(R_0 = r_0)$ and $T(R_0 = r'_0)$ are two $(C-1)$ dimensional trapping sets each containing at least 2^{C-1} elements by induction hypothesis. Thus, T has at least 2^C elements. ∎

Proposition 8. Let s be a fixed positive integer. Assume that $\alpha = 1 - \frac{1}{C}$ for some integer $C \geq 3$ and consider a hash structure with C different hashes as explained before. If the peeling decoder decodes all except a set of variables of size s, it can decode all of the variables with a probability at least $1 - O(1/N^{\frac{2C}{C}-2})$.

The proof follows from a similar proof in [9].

VIII. Experimental Results

In this section, we have empirically evaluated the performance of the SparseFHT algorithm for a variety of design parameters. The simulations are implemented in C programming language and the success probability of the algorithm has been estimated via sufficient number of simulations.

Experiment 1: We fix the signal size to $N = 2^{22}$ and run the algorithm 1200 times to estimate the success probability for $\alpha \in (0, \frac{1}{3}]$ and $C = \frac{1}{\alpha}$. Fig. 7 shows the simulation result.

Experiment 2: Fig. 8 shows a comparison of the runtime of the SparseFHT algorithm with a straightforward implementation of the fast Hadamard transform for $N = 2^{15}$.

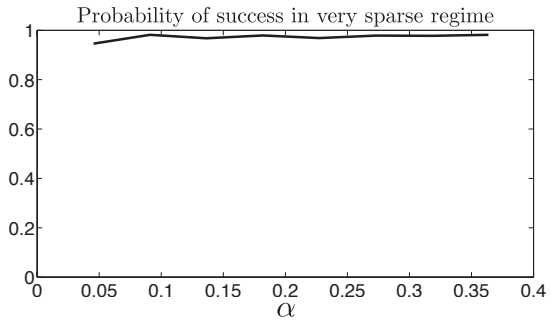

Fig. 7: Performance of SparseFHT for $\alpha \in (0, \frac{1}{3}]$, $N = 2^{22}$ and $C = \frac{1}{\alpha}$.

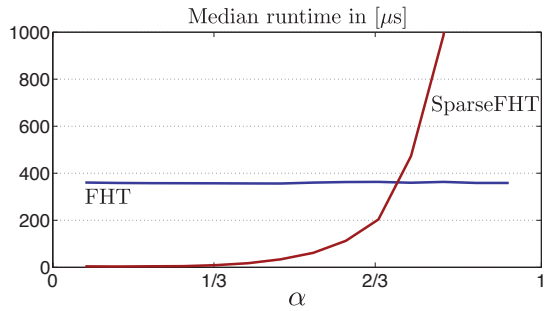

Fig. 8: Comparison of the Median runtime of the SparseFHT with the standard Hadamard transform for $N = 2^{15}$ and for different values of α.

References

[1] S. Haghighatshoar and E. Abbe, "Polarization of the Rényi information dimension for single and multi terminal analog compression," *arXiv preprint arXiv:1301.6388*, 2013.

[2] M. H. Lee and M. Kaveh, "Fast Hadamard transform based on a simple matrix factorization," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 6, pp. 1666–1667, 1986.

[3] J. R. Johnson and M. Pueschel, "In search of the optimal Walsh-Hadamard transform," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, 2000, pp. 3347–3350.

[4] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A Tutorial on Fast Fourier Sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 57–66, 2008.

[5] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse Fourier transform," *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1183–1194, 2012.

[6] ——, "Nearly optimal sparse Fourier transform," *Proceedings of the 44th symposium on Theory of Computing*, pp. 563–578, 2012.

[7] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi, "Sample-Optimal Average-Case Sparse Fourier Transform in Two Dimensions," *arXiv.org*, Mar. 2013.

[8] S. Pawar and K. Ramchandran, "A hybrid DFT-LDPC framework for fast, efficient and robust compressive sensing," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, 2012, pp. 1943–1950.

[9] ——, "Computing a k-sparse n-length Discrete Fourier Transform using at most 4k samples and O(k log k) complexity," *arXiv.org*, May 2013.

[10] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 569–584, 2001.