A Fast Hadamard Transform for Signals with Sub-linear Sparsity in the Transform Domain

Robin Scheibler, Student Member, IEEE Saeid Haghighatshoar, Student Member, IEEE Martin Vetterli, Fellow, IEEE

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 non-zero spectral values has been formulated as a belief propagation (BP) algorithm (peeling decoding) over a 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" $(\alpha \in (1,\frac{1}{3}])$ and the "less sparse" $(\alpha \in (\frac{1}{3},1))$ regime.

Index Terms—Walsh-Hadamard, Transform, sparse, sparse FFT, sub-linear, peeling decoder.

I. INTRODUCTION

THE fast Walsh-Hadamard transform (WHT) is a well-known signal processing tool with application in areas as varied as image compression and coding [1], spreading sequence for multi-user transmission in cellular networks (CDMA) [2], spectroscopy [3] as well as compressed sensing [4]. It has also nice properties studied in different areas of mathematics [5]. Its recursive structure, similar to the famous fast Fourier transform (FFT) algorithm for computing the discrete Fourier transform (DFT) of the signal, allows a fast computation with complexity $O(N\log_2(N))$ in the dimension of the signal N [6], [7].

A number of recent publications have addressed the particular problem of computing the DFT of an N dimensional signal under the assumption of K-sparsity of the signal in the frequency domain [8], [9], [10], [11], [12]. In particular, it has been shown that the well known computational complexity $O(N\log_2(N))$ of the FFT algorithm can be strictly improved. Such algorithms are generally known as sparse FFT (sFFT)

R. Scheibler, S. Haghighatshoar and Martin Vetterli are with the 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

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

A short version of this paper was presented at the 51st Annual Allerton Conference on Communication, Control, and Computing, Monticello, 2013.

algorithms. The authors in [13] by extending the results of [12], 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 [14], [15] developed a novel low complexity iterative algorithm to recover the non-zero frequency elements using ideas from sparse-graph codes [16].

In this paper, we develop a fast algorithm to compute the WHT of data sparse in the Hadamard domain. We first develop some useful properties of the WHT, specially the subsampling and the modulation property, that will later play a vital role in the development the algorithm. In particular, we show the subsampling in 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 non-zero components if more than one of them are involved in the induced linear combination. Similar to [15] and borrowing ideas from sparse-graph codes, we construct a bipartite graph by considering the non-zero 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 non-zero coefficients in the transform domain. Moreover, we give an iterative peeling decoder to recover those non-zero 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 non-zero 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, for any value of K.

Notations and Preliminaries: For m an integer, 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 $N=2^n$ a power of two, the i-th components of v is equivalently 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. Also sometimes the real value assigned to v_i is not important for

us and by v_i we simply mean the binary expansion associated to its index i, however, the distinction must be clear from the context. \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 binary components and addition of the vectors done component wise. The inner product of two n dimensional binary vectors u,v is defined by $\langle u\,,\,v\rangle = \sum_{t=0}^{n-1} u_t v_t$ with arithmetic over \mathbb{F}_2 although $\langle\,,\,\rangle$ is not an inner product in exact mathematical sense, for example, $\langle u\,,\,u\rangle = 0$ for any $u\in\mathbb{F}_2^n$.

For a signal $X \in \mathbb{R}^N$, the support of X is defined as $\operatorname{supp}(X) = \{i \in [N] : X_i \neq 0\}$. The signal X is called K-sparse if $|\operatorname{supp}(X)| = K$, 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 $K_N = \max_{X \in \mathcal{S}_N} |\operatorname{supp}(X)|$.

Definition 1. A class of signals of increasing dimension $\{S_N\}_{N=1}^{\infty}$ 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.

II. MAIN RESULTS

Let us first describe the main result of this work in the following theorem.

Theorem 1. Let $0 < \alpha < 1$, $N = 2^n$ a power of two and $K = N^{\alpha}$. Let $x \in \mathbb{R}^N$ be 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 at random 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 $C \leq (\frac{1}{\alpha} \vee \frac{1}{1-\alpha}) + 1$, 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 non-zero 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 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*.

Remark 1. To prove Theorem 1, we distinguish between the very sparse case $(0 < \alpha \le \frac{1}{3})$ and less sparse one $(\frac{1}{3} < \alpha < 1)$. Also, we implicitly assume that the algorithm knows the value of α which might not be possible in some cases. As we will see later if we know to which regime the signal belongs and some bounds on the value of the α , it is possible to design an algorithm that works for all those values of α . However, the sample and computational complexity of that algorithm might increase compared with the optimal one that knows the value of α . For example, if we know that the signal is very sparse, $\alpha \in (0, \alpha^*]$ with $\alpha^* \le \frac{1}{3}$, it is sufficient to design the algorithm for α^* and it will work for all signals with sparsity index less

that α^* . Similarly, if the signal is less sparse with a sparsity index $\alpha \in (\frac{1}{3}, \alpha^*)$, where $\alpha^* < 1$, then again it is sufficient to design the algorithm for α^* and it will automatically work for all $\alpha \in (\frac{1}{3}, \alpha^*)$.

Remark 2. In the very sparse regime $(0 < \alpha \le \frac{1}{3})$, we prove that for any value of α the success probability of the optimally designed algorithm is at least $1 - O(1/K^{3(C/2-1)})$, with $C = [\frac{1}{\alpha}]$ where for $u \in \mathbb{R}_+$, $[u] = \max\{n \in \mathbb{Z} : n \le u\}$. It is easy to show that for every value of $\alpha \in (0, \frac{1}{3})$ the success probability can be lower bounded by $1 - O(N^{-\frac{3}{8}})$.

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 $k \in \mathbb{F}_2^n$ denote the corresponding binary index of the transform domain component. Also, 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

This subsection is devoted to reviewing some of the basic properties of the WHT. Some of the properties are not directly used in the paper and we have included them for the sake of completeness. They can be of independent interest. The proofs of all the properties are provided in Appendix A.

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

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

The next property is more subtle and allows to partially permute the Hadamard spectrum in a specific way by applying a corresponding permutation in the time domain. However, the collection of all such possible permutations is limited. We give a full characterization of all those permutations. Technically, this property is equivalent to finding permutations π_1, π_2 : $[N] \rightarrow [N]$ with corresponding permutation matrices Π_1, Π_2 such that

$$\Pi_2 H_N = H_N \Pi_1,\tag{1}$$

where H_N is the Hadamard matrix of order N and where the permutation matrix corresponding to a permutation π is defined by $(\Pi)_{i,j}=1$ if and only if $\pi(i)=j$, and zero otherwise. The identity (1) is equivalent to finding a row permutation of H_N that can be equivalently obtained by a column permutation of H_N .

Property 2. All of the permutations satisfying (1) are described by the elements of

$$GL(n, \mathbb{F}_2) = \{ A \in \mathbb{F}_2^{n \times n} \mid A^{-1} \text{ exists} \},$$

the set of $n \times n$ non-singular matrices with entries in \mathbb{F}_2 .

Remark 3. The total number of possible permutations in Property 2, is $\prod_{i=0}^{n-1} (N-2^i)$, which is a negligible fraction of all N! permutation over [N].

Property 3 (Permutation). Let $\Sigma \in GL(n, \mathbb{F}_2)$. Assume that X_k is the WHT of the time domain signal x_m . Then

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

Remark 4. Notice that any $\Sigma \in \mathrm{GL}(n, \mathbb{F}_2)$ is a bijection from \mathbb{F}_2^n to \mathbb{F}_2^n , thus $x_{\Sigma m}$ is simply a vector obtained by permuting the initial vector x_m .

The last property is that of downsampling/aliasing. Notice that for a vector x of dimension $N=2^n$, we index every components by a binary vector of length n, namely, $x_{m_0,m_1,\dots,m_{n-1}}$. To subsample this vector along dimension i, we freeze the i-th component of the index to either 0 or 1. For example, $x_{0,m_1,\dots,m_{n-1}}$ is a 2^{n-1} dimensional vector obtained by subsampling the vector x_m along the first index.

Property 4 (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

$$\Psi_b = \left[\mathbf{0}_{b \times (n-b)} \ I_b \right]^T, \tag{2}$$

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

$$x_{\Psi_b m} \quad \stackrel{\mathrm{WHT}}{\longleftrightarrow} \quad \sqrt{\frac{B}{N}} \sum_{j \in \mathcal{N}\left(\Psi_b^T\right)} X_{\Psi_b k + j},$$

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

Notice that Property 4 can be simply applied for any matrix Ψ_b that subsamples any set of indices of length b not necessarily the b last ones.

Remark 5. The group \mathbb{F}_2^n can be visualized as the vertices of the n-dimensional hypercube. The downsampling property just explained 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 in the spectral domain. This is illustrated visually in Fig. 1 for dimension n=3.

Remark 6. In a general downsampling procedure, one can replace the frozen indices by an arbitrary but fixed binary pattern. The only difference is that instead of summing the aliased spectral components, one should also take into account the suitable $\{+,-\}$ sign patterns, namely, we have

$$x_{\Psi_b m + p} \stackrel{\text{WHT}}{\longleftrightarrow} \sqrt{\frac{B}{N}} \sum_{j \in \mathcal{N}(\Psi_b^T)} (-1)^{\langle p, j \rangle} X_{\Psi_b k + j}, \quad (3)$$

where p is a binary vector of length n with b zeros at the end.

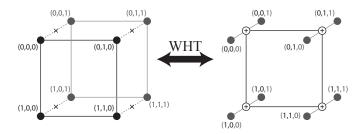


Fig. 1. Illustration of the downsampling property on a hypercube for $N=2^3$. The two cubes are the time-domain and Hadamard-domain signals on the left and right, respectively. We decide to drop all nodes whose third coordinate is '1'. We illustrate this by adding an '×' on the edges connecting these vertices through the third coordinate. This is equivalent to summing up vertices along the corresponding edges in the Hadamard domain.

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 and its WHT defined by

$$u_{\Sigma,p}(m) = \sqrt{\frac{N}{B}} x_{\Sigma \Psi_b m + p}.$$

Then, the length B Hadamard transform of $u_{\Sigma,p}$ is given by

$$U_{\Sigma,p}(k) = \sum_{j \in \mathbb{F}_2^n \mid \mathcal{H}_j = k} X_j \left(-1 \right)^{\langle p, j \rangle}, \tag{4}$$

where \mathcal{H} is the index hashing operator defined by

$$\mathcal{H} = \Psi_b^T \Sigma^T, \tag{5}$$

where Ψ_b is as in (2). Note that the index of components in the sum (4) can be explicitly written as function of the bin index k

$$j = \Sigma^{-T} \Psi_b k + q, \qquad q \in \mathcal{N}(\mathcal{H}).$$

The proof simply follows from the properties 1, 3, and 4. Based on Proposition 1, we give Algorithm 1 which computes the hashed Hadamard spectrum. Given an FFT-like fast Hadamard transform (FHT) algorithm, and picking B bins for hashing the spectrum, Algorithm 1 requires $O(B \log B)$ operations.

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

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

Ensure: U contains the hashed Hadamard spectrum of x.

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

$$U = \sqrt{\frac{N}{B}}$$
 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 an iterative peeling decoding algorithm to recover the non-zero spectral values. We explain how it is possible to identify collisions between the non-zero spectral coefficients that are hashed to the same bin and also to estimate the support of non-colliding components.

Let us 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 bin k – those whose index j satisfies $\mathcal{H}j=k$ – whereas in the latter the same variables are added together weighted by $(-1)^{\langle p\,,\,j\rangle}$. 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 at least some values of p. In fact, n-b well-chosen values of p allow to detect whether there is only one, or more than one non-zero components in the sum.

When there is only one $X_{j'} \neq 0$ hashed to the bin k $(h_{\Sigma}(j') = k)$, the result of the ratio test is precisely 1 or -1, depending on the value of the inner product between j' and p. In particular, we have

$$\langle p, j' \rangle = \mathbb{1}_{\{r_{\Sigma, n}(k) < 0\}},\tag{6}$$

where $\mathbbm{1}_{\{t<0\}}=1$ if t<0, and zero otherwise. Hence, if for n-b well-chosen values of p, the ratio test results in 1 or -1, implying that there is only one non-zero spectral coefficient in the corresponding hash bin, by some extra effort it is even possible to identify the position of this non-zero component. We formalize this result in the following proposition proved in Appendix B.

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

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

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

the index of the unique non-zero coefficient is given by

$$j = \Sigma^{-T}(\Psi_b k + \hat{v}). \tag{7}$$

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 non-zero coefficient.

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 non-zero 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, where $K = O(N^{\alpha})$ scales sub-linearly with N with index α . 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 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. For example, the first row of the Hadamard matrix is the all-one vector which implies that the sum of all of the components of X must be equal to the first time domain sample. A similar interpretation holds for the other rows. Thus, the WHT can be imagined as a code over a bipartite graph. With this picture in mind, one can consider the recovery of the non-zero spectral values as a decoding problem over this bipartite graph. If we consider the WHT, it is easy to see that the induced bipartite graph on the non-zero spectral values is a complete (dense) bipartite graph because any variable node is connected to all of the check nodes as has been depicted in the left part of Fig. 2, where $\{X_1, X_8, X_{11}\}$ are the only non-zero variables in the spectral domain and each check constraint correspond to the value of a time domain sample. It is also implicitly assumed that one knows the support of X, $\{1, 8, 11\}$ in our case. At the moment, it is not clear how one can obtain the position of the non-zero variables. As we will explain, in the final version of the algorithm this can be done by using Proposition 2.

For codes on bipartite graphs, there is a collection of low complexity belief propagation algorithms to recover the variable nodes given the value of check nodes. Most of these algorithms perform well assuming 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 components of the 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 operation must be clear from this picture. The idea is that hashing 'sparsifies' the underlying bipartite graph. It is also important to notice that in the bipartite graph induced by hashing, one can obtain all of the values of parity checks (hash outputs) by using low complexity operations on a small set of time domain samples as explained in Proposition 1.

We propose the following iterative algorithm to recover the non-zero spectral variables over the bipartite graph induced by hashing. The algorithm first tries to find a degree one check node. Using the terminology of [15], we call such a check node a *singleton*. Using Proposition 2, the algorithm is able

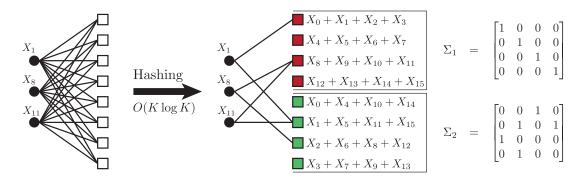


Fig. 2. On the left, bipartite graph representation of the WHT for N=8 and K=3. On the right, the underlying bipartite graph after applying C=2 different hashing produced by plugging Σ_1 , Σ_2 in (5) with B=4. The variable nodes (\bullet) are the non-zero spectral values to be recovered. The white check nodes (\square) are the original time-domain samples. The colored squares are new check nodes after applying Algorithm 1.

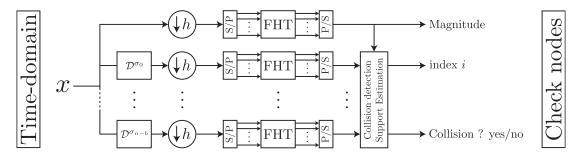


Fig. 3. A block diagram of the SparseFHT algorithm in the time domain. The downsampling plus small size low complexity FHT blocks compute different hash outputs. Delay blocks denote an index shift by σ_i before hashing. The S/P and P/S are serial-parallel and parallel-serial blocks to emphasize that the FHT operates on the whole signal at once. The collision detection/support estimation block implements Proposition 2 to identify if there is a collision. Index i is not valid when there is a collision.

to find the position and the value of the corresponding nonzero component and, 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, namely, it is satisfied. 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 which we call a *zeroton*. Also 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 that another singleton be found.

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*) and the algorithm fails to identify all of the non-zero spectral values.

A more detailed pseudo-code of the proposed iterative algorithm is given in Algorithm 2.

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 proof of the last part of Theorem 1, regarding the success probability of the algorithm, is the subject of Sections VI and VII for the very and less sparse regimes, respectively.

Computational Complexity: As will be further clarified in Sections VI and VII, depending on the sparsity index of the signal α , we will use C different hash blocks, where the signal α , we will use C underent 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 number of non-zero 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 $B \log_2(B)$ operations which assuming K = B, has a computational complexity $K \log_2(K)$. 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 for 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 for 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. PERFORMANCE ANALYSIS OF THE VERY SPARSE REGIME

In this section, we consider the very sparse regime, where $0 < \alpha \le \frac{1}{3}$. In this regime, we show that assuming a random support model for non-zero spectral components and a careful

Algorithm 2 SparseFHT (x, N, K, C, L, Σ) Require: Input signal x of length $N = 2^n$ Sparsity K

```
Require: Input signal x of length N=2^n. Sparsity K. Hash
   count C. Number of iterations of decoder L. Array \Sigma of C
   matrices in GL(n, \mathbb{F}_2), \Sigma_c = [\sigma_{c,1} \mid \cdots \mid \sigma_{c,n}], \ \sigma_{c,i} \in \mathbb{F}_2^n.
Ensure: X contains the sparse Hadamard spectrum of x.
   B = O(K)
   D = n - b + 1
   for c = 1, \ldots, C do
      U_{c,0} = \text{FastHadamardHashing}(x, N, \Sigma_c, 0, B)
      for d=1,\ldots,D do
          U_{c,d} = \text{FastHadamardHashing}(x, N, \Sigma_c, \sigma_{c,d}, B)
      end for
   end for
   for l=1,\ldots,L do
      for c = 1, \ldots, C do
         for k = 0, ..., B - 1 do
             if U_{c,0,k}=0 then
                continue to next k
             end if
             \hat{v} \leftarrow 0
             for d = 1, \ldots, D do
                if U_{c,d,k}/U_{c,0,k} = -1 then
                   \hat{v}_{d-1} \leftarrow 1
                else if U_{c,d,k}/U_{c,0,k} \neq 1 then
                   continue to next k
                end if
             end for
            i \leftarrow \Sigma_c^{-T}(\Psi_b \, k + \hat{v})
             X_i \leftarrow U_{c,0,k}
             for c'=1,\ldots,C do
                j \leftarrow \Psi_h^T \Sigma_{c'}^T i
                U_{c',0,j} \leftarrow \overset{c}{U}_{c',0,j} - X_i
for d' = 1, \dots, D do
                   U_{c',d',j} \leftarrow U_{c',d',j} - X_i(-1)^{\langle \sigma_{c',d'},i \rangle}
             end for
         end for
      end for
```

design of hash functions, it is possible to obtain a random bipartite graph with variable nodes corresponding to non-zero spectral components and with check nodes corresponding to outputs of hash functions. We explicitly prove that asymptotically this random graph behaves similarly to the ensemble of LDPC bipartite graph. Running the peeling decoder to recover the spectral components is also equivalent to the belief propagation (BP) decoding for a binary erasure channel (BEC). Fortunately, there is a rich literature in coding theory about asymptotic performance of the BP decoder. Specially, it is possible to 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.

end for

We use the following steps to rigorously analyze the performance of the decoder in the very sparse regime:

- 1) We explain construction of suitable hash functions depending on the value of $\alpha \in (0, \frac{1}{2}]$.
- 2) We rigorously analyze the structure of the induced bipartite graph obtained by treating the non-zero spectral components as variable nodes and the output of hash functions as check nodes. In particular, we prove that the resulting graph is a fully random left regular bipartite graph similar to the regular LDPC ensemble. We also obtain variable and check degrees distribution polynomials for this graph.
- 3) At every stage, the peeling decoder recovers some of the variable nodes, removing all the edges incident to those variable nodes. We use Wormald's method given in [17] to prove the concentration of the number of unpeeled edges around its expected value, which we also characterize. Wormald's method as exploited in [18], uses the differential equation approach to track the evolution of the number of edges in the underlying bipartite graph. Specifically, it shows that the number of edges at every step of the algorithm is very well concentrated around the solution of the associated differential equations.
- 4) Wormald's method gives a concentration bound to the number of remaining edges as far as their count is a fixed ratio $\gamma \in (0,1)$ of the initial edges in the graph. Another expander argument as in [18] is necessary to show that if the peeling decoder peels $1-\gamma$ ratio of the edges successfully, it can continue to peel off all the remaining edges with very high probability.

A. Hash Construction

For the very sparse regime, $0 < \alpha \leq \frac{1}{3}$, consider those values of α equal to $\frac{1}{C}$ for some positive integer $C \geq 3$. We will explain later how to cover the intermediate values. 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 . We design C different subsampling operator, where the i-th one returns all of the variables with indices i b up to (i+1)b-1 kept and the other indices set to zero. Using the terminology of Proposition 1, we let Σ_i be the identity matrix with columns circularly shifted by (i+1)b to the left. Then, the hash operator given by (5) is

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

where I_b is the identity matrix of order b and Φ_b is defined in (2). To give further intuition about hash construction, let us label the elements of the vector x with their binary representation $x_0^{n-1} \in \mathbb{F}_2^n$. 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. In particular, labeling the transform domain components by $X_0^{n-1} \in \mathbb{F}_2^n$ and ignoring the multiplicative constants, it is seen from Equation (4) that 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.

In terms of complexity, to obtain the output of each hash bin, we only need to compute the WHT of a smaller subsampled signal of dimension B. Note that by hash construction, K=B which implies that all of the hash functions can be computed in $CK\log_2(K)$ operations. As we will explain later, we need at least C=3 hashes for the peeling algorithm to work successfully and that is the main reason why this construction works for $\alpha \leq \frac{1}{3}$. For intermediate values of α , those not equal to $\frac{1}{C}$ for some integer C, one can construct $\left[\frac{1}{\alpha}\right]$ hashes with $B=2^{[n\alpha]}$ output bins and one hash with smaller number of output bins, thus obtaining a computational complexity of order $(1+\left[\frac{1}{\alpha}\right])K\log_2(K)$.

B. Random Bipartite Graph Construction

1) Random Support Model: For an N dimensional signal $x \in \mathbb{R}^N$, the support of x is defined as $supp(x) = \{i \in [N] : x \in \mathbb{R}^N : x$ $x_i \neq 0$. The signal x is called K sparse if $|\operatorname{supp}(x)| =$ K, where for $A \subset [N]$, |A| denotes the cardinality of A. For a given (K, N), RS1(K, N) is the class of all stochastic signals whose support is uniformly at random selected from the set of all $\binom{N}{K}$ possible supports of size K. We do not put any constraint on the non-zero components; they can be deterministic or random. Model RS1 is equivalent to selecting K out of N objects at random without replacement. If we assume that the selection of the indices for the support is done independently but with replacement, we obtain another model that we call RS2(K, N). In particular, if $V_i, i \in [K]$ are i.i.d. random variables uniformly distributed over [N], a random support in RS2(K, N) is given by the random set $\{V_i : i \in$ [K]. Obviously, the size of a random support in RS2(K, N)is not necessarily fixed but it is at most K. The following proposition, proved in Appendix C, shows that in the sublinear sparsity regime, RS1 and RS2 are essentially equivalent.

Proposition 3. Consider the random support model RS2(K,N), where $K=N^{\alpha}$ for some fixed $0<\alpha<1$ and let H be the random size of the support set. Asymptotically as N tends to infinity $\frac{H}{K}$ converges to 1 in probability.

2) 'Balls and Bins' Model $\mathcal{G}(K,B,C)$: Consider C disjoint sets of check nodes S_1,S_2,\ldots,S_C of the same size $|S_i|=B$. A graph in the ensemble of random bipartite graphs \mathcal{G} with K variable nodes at the left and $C\times B$ check nodes $\cup_{i=1}^C S_i$ at the right is generated as follows. Each variable node v in \mathcal{G} , independently from other variable nodes, is connected to check nodes $\{s_1,s_2,\ldots,s_C\}$ where $s_i\in S_i$ is uniformly at random selected from S_i and selection of s_i 's are independent of one another. Every edge e in \mathcal{G} can be labelled as (v,c), where $v\in [K]$ is a variable node and c is a check node in one of S_1,S_2,\ldots,S_C . For a variable node v, the neighbors of v denoted by $\mathcal{N}(v)$ consists of C different check nodes connected to v, each of them from a different S_i . Similarly, for a check node $c\in \cup_{i=1}^C S_i$, $\mathcal{N}(c)$ is the set of all variable nodes connected to c.

By construction, all of the resulting bipartite graphs in the ensemble are left regular with variable degree C but the check node degree is not fixed. During the construction, it might happen that two variable nodes have exactly the same

neighborhood. In that case, we consider them as equivalent variables and keep only one of them and remove the other, thus the number of variable nodes in a graph from the ensemble $\mathcal{G}(K,B,C)$ might be less than K.

This model is a variation of the Balls and Bins model, where we have K balls, C buckets of different color each containing B bins and every ball selects one bin from each bucket at random independent of the other balls.

Here we also recall some terminology from graph theory that we will use later. A walk of size ℓ in graph $\mathcal G$ starting from a node $v \in [K]$ is a set of ℓ edges e_1, e_2, \ldots, e_ℓ , where $v \in e_1$ and where consecutive edges are different, $e_i \neq e_{i+1}$, but incident with each other $e_i \cap e_{i+1} \neq \emptyset$. A directed neighborhood of an edge e = (v, c) of depth ℓ is the induced subgraph in $\mathcal G$ consisting of all edges and associated check and variable nodes in all walks of size $\ell+1$ starting from v with the first edge being $e_1 = (v, c)$. An edge e is said to have a tree neighborhood of depth ℓ if the directed neighborhood of e of depth ℓ is a tree.

3) Ensemble of Graphs Generated by Hashing: In the very sparse regime $(0 < \alpha < \frac{1}{3})$, in order to keep the computational complexity of the hashing algorithm around $O(K\log_2(K))$, we constructed $C=\frac{1}{\alpha}$ different surjective hash functions $h_i: \mathbb{F}_2^n \to \mathbb{F}_2^b, i \in [C]$, where $b \approx n\alpha$ and where for an $x \in \mathbb{F}_2^n$ with binary representation $x_0^{n-1}, h_i(x_0^{n-1}) = (x_{ib}, x_{ib+1}, \ldots, x_{ib+b-1})$. We also explained that in the spectral domain, this operation is equivalent to hashing spectral the component labeled with $X_0^{n-1} \in \mathbb{F}_2^n$ into the bin labelled with $h_i(X_0^{n-1})$. Notice that by this hashing scheme there is a one-to-one relation between a spectral element X and its bin indices in different hashes $(h_0(X), h_1(X), \ldots, h_{C-1}(X))$.

Let V be a uniformly distributed random variable over \mathbb{F}_2^n . It is easy to check that in the binary representation of V, V_0^{n-1} are like i.i.d. unbiased bits. This implies that $h_0(V), h_1(V), \ldots, h_{C-1}(V)$ will be independent from one another because they depend on disjoint subsets of V_0^{n-1} . Moreover, $h_i(V)$ is also uniformly distributed over \mathbb{F}_2^b .

Assume that X_1, X_2, \ldots, X_K are K different variables in \mathbb{F}_2^n denoting the position of non-zero 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 $h_i(X_i) \in S_i$, $j \in [C]$.

Proposition 4. Let $h_i: \mathbb{F}_2^n \to \mathbb{F}_2^b$, $i \in [C]$ be as explained before. Let V_1, V_2, \ldots, V_K be a set of variables generated from the ensemble RS2(K, N), $N = 2^n$ denoting the position of non-zero components. The bipartite graph associated with variables V_1^K and hash functions h_i is a graph from ensemble $\mathcal{G}(K, B, C)$, where $B = 2^b$.

Proof: As V_1^K belong to the ensemble RS2(N,K), they are i.i.d. variables uniformly distributed in [N]. This implies that for a specific V_i , $h_j(V_i)$, $j \in [C]$ are independent from one another. Thus, every variable node selects its neighbor

checks in $S_0, S_1, \ldots, S_{C-1}$ completely at random. Moreover, for any $j \in [C]$, the variables $h_j(V_1), \ldots, h_j(V_K)$ are also independent, thus each variable selects its neighbor checks in S_j independent of all other variables. This implies that in the corresponding bipartite graph, every variable node selects its C check neighbors completely at random independent of other variable nodes, thus it belongs to $\mathcal{G}(K,B,C)$.

In Section V, we explained the peeling decoder over the bipartite graph induced by the non-zero spectral components. It is easy to see that the performance of the algorithm always improves if we remove some of the variable nodes from the graph because it potentially reduces the number of colliding variables in the graph and there is more chance for the peeling decoder to succeed decoding.

Proposition 5. Let α , C, K, h_i , $i \in [C]$ be as explained before. Let \mathcal{G} be the bipartite graph induced by the random support set V_1^K generated from RS1 and hash functions h_i . For any $\epsilon > 0$, asymptotically as N tends to infinity, the average failure probability of the peeling decoder over \mathcal{G} is upper bounded by its average failure probability over the ensemble $\mathcal{G}(K(1+\epsilon),B,C)$.

Proof: Let \mathcal{G}_{ϵ} be a graph from ensemble $\mathcal{G}(K(1+\epsilon),B,C)$. From Proposition 3, asymptotically the number of variable nodes in \mathcal{G}_{ϵ} is greater than K. If we drop some of the variable nodes at random from \mathcal{G}_{ϵ} to keep only K of them we obtain a graph from ensemble \mathcal{G} . From the explanation of the peeling decoder, it is easy to see that the performance of the decoder improves by removing some of the variable nodes because in that case less variables are collided together in different bins and there is more chance to peel them off. This implies that peeling decoder performs strictly better over \mathcal{G} compared with \mathcal{G}_{ϵ} .

Remark 7. If we consider the graph induced by V_1^K from RS1 and hash functions h_i , the edge connection between variable nodes and check nodes is not completely random thus it is not compatible with Balls-and-Bins model explained before. Proposition 5 implies that asymptotically the failure probability for this model can be upper bounded by the failure probability of the peeling decoder for Balls-and-Bins model of slightly higher number of edges $K(1+\epsilon)$.

4) Edge Degree Distribution Polynomial: As we explained in the previous section, assuming a random support model for non-zero spectral components in the very sparse regime $0<\alpha<\frac{1}{3}$, we obtained a random graph from ensemble $\mathcal{G}(K,B,C)$. We also assumed that $n\alpha\in\mathbb{N}$ and we selected $b=n\alpha$, thus K=B. Let us call $\beta=\frac{K}{B}$ to be the average number of non-zero components per a hash bin. In our case, we designed hashes so that $\beta=1$. 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 6. Let $\mathcal{G}(K,B,C)$ be the random graph ensemble as before with $\beta = \frac{K}{B}$ fixed. Then asymptotically as N tends to infinity the check degree converges to a Poisson random variable with parameter β .

Proof: Construction of the ensemble $\mathcal G$ shows that any variable node has a probability of $\frac{1}{B}$ to be connected to a specific check node, c, independent of all other variable nodes. Let $Z_i \in \{0,1\}$ be a Bernoulli random variable where $Z_i = 1$ if and only if variable i is connected to check node c. It is easy to check that the degree of c will be $Z = \sum_{i=1}^K Z_i$. The Characteristic function of Z can be easily obtained:

$$\Phi_Z(\omega) = \mathbb{E}\left[e^{j\omega Z}\right] = \prod_{i=1}^K \mathbb{E}\left[e^{j\omega Z_i}\right]$$
$$= \left(1 + \frac{1}{B}(e^{j\omega} - 1)\right)^{\beta B} \to e^{\beta(e^{j\omega} - 1)},$$

showing the convergence of Z to a Poisson distribution with parameter 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. Notice that we have i-1 instead of i in the formula. This choice makes the analysis to be written in a more compact form as we will see.

Proposition 7. 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.

Proof: From left regularity of a graph from ensemble \mathcal{G} , it results that all of the edges are connected to variable nodes of degree C, thus $\lambda(\alpha)=\alpha^{C-1}$ and the number of edges is equal to CK. By symmetry of hash construction, it is sufficient to obtain the edge degree distribution polynomial for check nodes of the first hash. The total number of edges that are connected to the check nodes of the first hash is equal to K. Let N_i be the number of check nodes in this hash with degree i. By definition of ρ_i , it results that

$$\rho_i = \frac{iN_i}{K} = \frac{iN_i/B}{K/B}.$$

Let Z be the random variable as in the proof of Proposition 6 denoting the degree of a specific check node. Then, as N tends to infinity one can show that

$$\lim_{N\to\infty}\frac{N_i}{B}=\lim_{N\to\infty}\mathbb{P}\left\{Z=i\right\}=\frac{e^{-\beta}\beta^i}{i!} \text{ a.s.}$$

Thus ρ_i converges almost surely to $\frac{e^{-\beta}\beta^{i-1}}{(i-1)!}$. As $\rho_i \leq 1$, for any $\alpha : |\alpha| < 1 - \epsilon$, $|\rho_i\alpha^{i-1}| \leq (1 - \epsilon)^{i-1}$ and applying the Dominated Convergence Theorem, $\rho(\alpha)$ converges to $\sum_{i=1}^{\infty} \frac{e^{-\beta}\beta^{i-1}}{(i-1)!} \alpha^{i-1} = e^{-\beta(1-\alpha)}.$

5) Average Check Degree Parameter β : In the very sparse regime, as we explained assuming that $b=n\alpha$ is an integer we designed independent hashes with $B=2^b$ output bins so that $\beta=\frac{K}{B}=1$. As we will see the performance of the peeling decoder (described later by the DE equation in (8)) depends on the parameter β . The less β the better the performance of the peeling decoder. Also notice that decreasing β via increasing β increases the time complexity $O(B\log_2(B))$ of computing the hash functions. For the general case, one can select β such that $\beta \in [1,2)$ or at the cost of increasing the computational

complexity make β smaller for example $\beta \in [\frac{1}{2},1)$ to obtain a better performance.

C. Performance Analysis of the Peeling Decoder

Assume that \mathcal{G} is the random bipartite graph resulting from applying C hashes to signal spectrum. As we explained in Section V, the iterative peeling algorithm starts by finding a singleton (check node of degree 1 which contains only one variable node or non-zero spectral components). The decoder peels off this variable node and removes all of the edges connected to it from the graph. The algorithm continues by peeling off a singleton at each step until all of the check nodes are zeroton; all of the non-zero variable nodes are decoded, or all of the remaining unpeeled check nodes are multiton in which case the algorithm fails to completely decode all the spectral variables.

1) Wormald's Method: In order to analyze the behavior of the resulting random graphs under the peeling decoding, the authors in [18] applied Wormald's method to track the ratio of edges in the graph connected to check nodes of degree 1 (singleton). The essence of Wormald's method is to approximate the behavior of a stochastic system (here the random bipartite graph), after applying suitable time normalization, by a deterministic differential equation. The idea is that asymptotically as the size of the system becomes large (thermodynamic limit), the random state of the system is, uniformly for all times during the run of the algorithm, well concentrated around the solution of the differential equation. In [18], this method was applied to analyze the performance of the peeling decoder for bipartite graph codes over the BEC. We briefly explain the problem setting in [18] and how it can be used in our case.

Assume that we have a bipartite graph \mathcal{G} with k variable nodes at the left, ck check nodes at the right and with edge degree polynomials $\lambda(x)$ and $\rho(x)$. We can define a channel code $\mathcal{C}(\mathcal{G})$ over this graph as follows. We assign k independent message bits to k input variable nodes. The output of each check node is the module 2 summation (XOR or summation over \mathbb{F}_2) of the all of the message bits that are connected to it. Thus, the resulting code will be a systematic code with kmessage bits along with ck parity check bits. To communicate a k bit message over the channel, we send k message bits and all of the check bits associated with them. While passing through the BEC, some of the message bits or check bits are erased independently. Assume a specific case in which the message bits and check bits are erased independently with probability δ and δ' respectively. Those message bits that pass perfectly through the channel are successfully transmitted, thus, the decoder tries to recover the erased message bits from the redundant information received via check bits. If we consider the induced graph after removing all variable nodes and check nodes corresponding to the erased ones from \mathcal{G} , we end up with another bipartite graph \mathcal{G}' . It is easy to see that over the new graph \mathcal{G}' , one can apply the peeling decoder to recover the erased bits.

In [18], this problem was fully analyzed for the case of $\delta' = 0$, where all of the check bits are received perfectly but

 δ ratio of the message bits are erased independently from one another. In other words, the final graph \mathcal{G}' has on average $k\delta$ variable nodes to be decoded. Therefore, the analysis can be simply applied to our case, by assuming that $\delta \to 1$, where all of the variable nodes are erased (they are all unknown and need to by identified). Notice that from the assumption $\delta'=0$ no check bit is erased as is the case in our problem. In particular, Proposition 2 in [18] states that

Proposition 2 in [18]: Let \mathcal{G} be a bipartite graph with edge degrees specified by $\lambda(x)$ and $\rho(x)$ and with k message bits chosen at random. Let δ be fixed so that

$$\rho(1 - \delta\lambda(x)) > 1 - x$$
, for $x \in (0, 1]$.

For any $\eta > 0$, there is some k_0 such that for all $k > k_0$, if the message bits of $\mathcal{C}(\mathcal{G})$ are erased independently with probability δ , then with probability at least $1 - k^{\frac{2}{3}} \exp(-\sqrt[3]{k}/2)$ the recovery algorithm terminates with at most ηk message bits erased

Replacing $\delta = 1$ in the proposition above, we obtain the following performance guarantee for the peeling decoder.

Proposition 8. Let \mathcal{G} be a bipartite graph from the ensemble $\mathcal{G}(K,B,C)$ induced by hashing functions $h_i, i \in [C]$ as explained before with $\beta = \frac{K}{B}$ and edge degree polynomials $\lambda(x) = x^{C-1}$ and $\rho(x) = e^{-\beta(1-x)}$ such that

$$\rho(1 - \lambda(x)) > 1 - x$$
, for $x \in (0, 1]$.

Given any $\epsilon \in (0,1)$, there is a K_0 such that for any $K > K_0$ with probability at least $1 - K^{\frac{2}{3}} \exp(-\sqrt[3]{K}/2)$ the peeling decoder terminates with at most ϵK unrecovered non-zero spectral components.

Proposition 8 does not guarantee the success of the peeling decoder. It only implies that with very high probability, it can peel off any ratio $\eta \in (0,1)$ of non-zero components but not necessarily all of them. However, using a combinatorial argument, it is possible to prove that with very high probability any graph in the ensemble $\mathcal G$ is an expander graph, namely, every small enough subset of left nodes has many check neighbors. This implies that if the peeling decoder can decode a specific ratio of variable nodes, it can proceed to decode all of them. A slight modification of Lemma 1 in [18] gives the following result proved in Appendix D.

Proposition 9. Let \mathcal{G} be a graph from the ensemble $\mathcal{G}(K,B,C)$ with $C\geq 3$. There is some $\eta>0$ 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.

Proof of Part 3 of Theorem 1 for $\alpha \in (0,\frac{1}{3}]$: In the very sparse regime $\alpha \in (0,\frac{1}{3}]$, we construct $C = \left[\frac{1}{\alpha}\right] \geq 3$ hashes each containing $2^{n\alpha}$ output bins. Combining Proposition 8 and 9, we obtain that the success probability of the peeling decoder is lower bounded by $1 - O(\frac{1}{K^{3(C/2-1)}})$ as mentioned in Remark 2.

2) Analysis based on Belief Propagation over Sparse Graphs: In this section, we give another method of analysis and further intuition about the performance of the peeling

decoder and why it works 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 to recover non-zero frequency components in [15]. Consider a specific edge e=(v,c) in a graph from ensemble $\mathcal{G}(K,B,C)$. Consider a directed neighborhood of this edge of depth ℓ as explained is VI-B2. At the first stage, it is easy to see that this edge is peeled off from the graph assuming that all of 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. This pictorially shown in Figure 4.

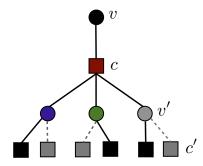


Fig. 4. Tree-like neighborhood an an edge e=(v,c). Dashed lines show the edges that have been removed before iteration t. The edge e is peeled off at iteration t because all the variable nodes v' connected to c are already decoded, thus c is a singleton check.

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 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 neighborhood of depth ℓ assuming a tree up to depth ℓ . A simple analysis similar to [15], gives the following recursion

$$p_{j+1} = \lambda(1 - \rho(1 - p_j)), \quad j \in [\ell],$$
 (8)

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 [15], 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. Specifically, if we start 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} . Moreover, a martingale argument was applied in [15] to show that not only the average of unpeeled edges is approximately KCp_{ℓ} but also with very high probability the number of those edges is well concentrated around KCp_{ℓ} .

Equation (8) is in general known as density evolution equation. Starting from $p_0=1$, this equation fully predicts the behavior 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 $\beta=\frac{K}{B}$.

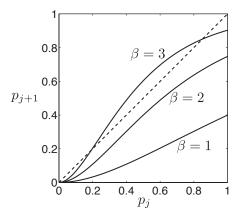


Fig. 5. Density Evolution equation for C=3 and different values of $\beta=\frac{K}{B}$

For very small values of β , this equation has only a fixed point 0 which implies that asymptotically the peeling decoder can recover a ratio of variables very close to 1. 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. It is easy to see that the only fixed point is 0 provided that for any $p \in (0,1]$, $p > \lambda(1-\rho(1-p))$. As $\lambda : [0,1] \to [0,1]$, $\lambda(x) = x^{C-1}$ is an increasing function of x, by change of variable $x = \lambda^{-1}(p)$, one obtains that $x > 1 - \rho(1 - \lambda(x))$ or equivalently

$$\rho(1 - \lambda(x)) > 1 - x$$
, for $x \in (0, 1]$.

This is exactly the same result that we obtained by applying Wormald's method as in [18]. In particular, this analysis clarifies the role of x in Wormald's method.

Similar to Wormald's method, this analysis only guaranties that for any $\epsilon \in (0,1)$, asymptotically as N tends to infinity, $1-\epsilon$ ratio of the variable nodes can be recovered. An expander argument is again necessary to guarantee the full recovery of all the remaining variables.

VII. PERFORMANCE ANALYSIS OF THE LESS SPARSE REGIME

For the less sparse regime $(\frac{1}{3} < \alpha < 1)$, similar to the very sparse case, we will first construct suitable hash functions which guarantee a low computational complexity of order $O(K\log_2(K)\log_2(\frac{N}{K}))$ for the recovery of non-zero spectral values. Assuming a uniformly random support model in the spectral domain, similar to the very sparse case, we can represent the hashes by a regular bipartite graph. Over this graph, the peeling algorithm proceeds to find singleton checks and peel the associated variables from the graph until no singleton remains. The recovery is successful if all of the variables are peeled off, thus, all of the remaining checks are zeroton otherwise some of the non-zero spectral values are not recovered and the perfect recovery fails.

As we will explain, the structure of the induced bipartite graph in this regime is a bit different than the very sparse one. The following steps are used to analyze the performance of the peeling decoder:

- 1) Constructing suitable hash functions
- 2) Representing hashing of non-zero 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 $C\geq 3$. We will explain how to deal with arbitrary values of C and α , especially those in the range $(\frac{1}{3},\frac{2}{3})$, in Section VII-D.

A. Hash Construction

Assume that $\alpha=1-\frac{1}{C}$ for some integer $C\geq 3$. Let x be an N dimensional signal with $N=2^n$ and let X denote its WHT. For simplicity, we label the components of X by a binary vector $X_0^{n-1}\in\mathbb{F}_2^n$. 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 $r_i=X_{i\,t}^{(i+1)t-1}$. It is clear that there is a one-to-one relation between each 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 different subsets of (r_0,r_1,\ldots,r_{C-1}) of size C-1 and appending them together. For example

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

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 . Moreover, if we denote the null space of h_i by $\mathcal{N}(h_i)$, it is easy to show that for any $i, j \in [C], i \neq j$, $\mathcal{N}(h_i) \cap \mathcal{N}(h_j) = \mathbf{0} \in \mathbb{F}_2^n$.

Using the subsampling property of the WHT and similar to the hash construction that we had in Subsection VI-A, it is seen that subsampling the time domain signal and taking WHT of the subsampled signal is equivalent to hashing the spectral components of the signal. In particular, all of the spectral components X_0^{n-1} with the same $h_i(X_0^{n-1})$ are mapped into the same bin in hash i, thus, different bins of the hash can be labelled with B different elements of \mathbb{F}_2^b .

It is easy to see that, with this construction the average number of non-zero 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 left nodes (variable nodes) corresponding to non-zero spectral components and with CB right 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 non-zero spectral component labelled with X_0^{n-1} is connected to nodes $s_i \in S_i$ if and only if the binary label

assigned to s_i is $h_i(X_0^{n-1})$. In the very sparse regime, we showed that if the support of the signal is generated according to the RS2(K,N), where K random positions are selected uniformly at random independent from one another from [N], then the resulting graph is a random left regular bipartite graph, where each variable nodes select its C neighbors in S_1, S_2, \ldots, S_C completely independently. However, in the less sparse regime, the selection of the neighbor checks in different hashes is not completely random. To explain more, let us assume that $\alpha = \frac{2}{3}$, thus C = 3. Also assume that for a non-zero spectral variable labelled with X_0^{n-1} , r_i denotes $X_{i\,t}^{(i+1)t-1}$, 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. This has been pictorially shown in Figure 6.

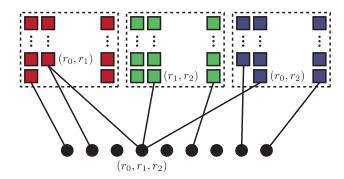


Fig. 6. Bipartite graph representation for the less sparse case $\alpha = \frac{2}{3}$, C = 3

If we assume that X_0^{n-1} is selected uniformly at random from \mathbb{F}_2^n then the bin numbers is each hash, i.e. (r_0, r_1) in the first hash, are individually selected uniformly at random among all possible bins. 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 variable V_1^K are selected independently as the position of non-zero spectral variables, the bin association for different variables V_i is still done independently.

C. Performance Analysis of the Peeling Decoder

As the resulting bipartite graph is not a completely random graph, it is not possible to directly apply Wormald's method as we did for the very sparse case as in [18]. However, an analysis based on the DE for the BP algorithm can still be applied. In other words, setting $p_0=1$ and

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

as in (8) 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 [15] can be applied to show that the number of remaining edges is also well concentrated around its average. Similar to the very sparse case, this argument asymptotically guarantees the recovery of any ratio of the variables between 0 and 1. 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 use the concept of trapping sets 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.

Notice that a trapping set can not be decoded because all of its neighbor check nodes are multiton. We first analyze the structure of the trapping set and find the probability that a specific set of variables build a trapping set. Let X be a spectral variable in the trapping set with the corresponding binary representation X_0^{n-1} and assume that C=3. As we explained, 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, there is a simple interpretation for a set T to be a trapping set, namely, 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 axis. Notice that in this case each hash is equivalent to projecting (r_0, r_1, r_2) onto two dimensional planes spanned by different coordinates, for example, $h_1(r_0, r_1, r_2) = (r_0, r_1)$ is a projection on the plane spanned by the first and second coordinate axes of the lattice. A similar argument holds for other values of C > 3, thus, larger values of α .

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

$$L(R_i = r_i) = \{ (r_0, \dots, r_{i-1}, r_{i+1}, \dots, r_{C-1}) :$$

$$(r_0, r_1, \dots, r_{i-1}, r_i, r_{i+1}, \dots, r_{C-1}) \in L \}.$$

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

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

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 10. Assume that T is a trapping set for the C dimensional lattice representation L of the non-zero spectral domain variables as explained before. Then for any r_i on the i-th 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 11. The size of the trapping set for a C dimensional lattice is at least 2^C .

Proof: We use 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 10, $T(R_0 = r_0)$ and $T(R_0 = r'_0)$ are two (C-1) dimensional trapping sets each consisting of at least 2^{C-1} elements by induction hypothesis. Thus, T has at least 2^C elements.

Remark 8. The bound $|T| \ge 2^C$ on the size of the trapping set is actually tight. For example, for $i \in [C]$ consider r_i, r'_i where $r_i \ne r'_i$ and let

$$T = \{(a_0, a_1, \dots, a_{C-1}) : a_i \in \{r_i, r_i'\}, i \in [C]\}.$$

It is easy to see that T is a trapping set with 2^C elements corresponding to the vertices of a C dimensional cube.

We now prove the following proposition which implies that if the peeling decoder can decode all of the variable nodes except a fixed number of them, with high probability it can continue to decode all of them.

Proposition 12. 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 very high probability.

Proof: The proof in very similar to [15]. Let T be a trapping set of size s. By Proposition 11, we have $s \geq 2^C$. Let p_i be the number of distinct values taken by elements of T along the R_i axis and let $p_{\max} = \max_{i \in [C]} p_i$. Without loss of generality, let us assume that the R_0 axis is the one having the maximum p_i . Consider $T(R_0 = r_0)$ for those $p_{\rm max}$ values of r_0 along the R_0 axis. Proposition 10 implies that each $T(R_0 = r_0)$ is a trapping set which has at least 2^{C-1} elements according to Proposition 11. This implies that $s \geq 2^{C-1}p_{\max}$ or $p_{\max} \leq \frac{s}{2^{C-1}}$. Moreover, T being the trapping set implies that there are subsets T_i consisting of elements from axes R_i and all of the elements of T are restricted to take their i-th coordinate values along R_i from the set T_i . Considering the way that we generate the position of non-zero variables X_0^{n-1} with the equivalent representation $(r_0, r_1, \dots, r_{C-1})$, the coordinate of any variable is selected uniformly and completely independently from on another and from the coordinates of the other variables. This implies that

$$\begin{split} \mathbb{P}\left\{F_{s}\right\} &\leq \mathbb{P}\left\{\text{For any variables in } T,\, r_{i} \in T_{i}, i \in [C]\right\} \\ &\leq \prod_{i=0}^{C-1} \binom{\mathcal{P}i}{p_{i}} (\frac{p_{i}}{\mathcal{P}_{i}})^{s} \leq \prod_{i=0}^{C-1} \binom{\mathcal{P}i}{s/2^{C-1}} (\frac{s}{2^{C-1}\mathcal{P}_{i}})^{s}, \end{split}$$

where F_s is the event that the peeling decoder fails to decode a specific subset of variables of size s and where \mathcal{P}_i denotes the number of all possible values for the i-th coordinate of a variable. By our construction all \mathcal{P}_i are equal to $P = 2^{n/C} =$ $2^{n(1-\alpha)} = N^{(1-\alpha)}$, thus we obtain that

$$\mathbb{P}\left\{F_{s}\right\} \leq \binom{P}{s/2^{C-1}}^{C} \left(\frac{s}{2^{C-1}P}\right)^{sC} \\
\leq \left(\frac{2^{C-1}Pe}{s}\right)^{sC/2^{C-1}} \left(\frac{s}{2^{C-1}P}\right)^{sC} \\
\leq \left(\frac{se^{1/(2^{C-1}-1)}}{2^{C-1}P}\right)^{sC(1-1/2^{C-1})} .$$

Taking the union bound over all $\binom{K}{s}$ possible ways of selection of s variables out of K variables, we obtain that

$$\begin{split} \mathbb{P}\left\{F\right\} &\leq \binom{K}{s} \mathbb{P}\left\{F_{s}\right\} \\ &\leq \left(\frac{eP^{C-1}}{s}\right)^{s} \left(\frac{se^{1/(2^{C-1}-1)}}{2^{C-1}P}\right)^{sC(1-1/2^{C-1})} \\ &= O(1/P^{s(1-\frac{C}{2^{C-1}})}) \\ &\leq O(1/P^{(2^{C}-2C)}) = O(1/N^{\frac{2^{C}}{C}-2}). \end{split}$$

For $C \geq 3$, this gives an upper bound of $O(N^{-\frac{2}{3}})$.

D. Generalized Hash Construction

The hash construction that we explained only covers values of $\alpha=1-\frac{1}{C}$ for $C\geq 3$ which belongs to the region $\alpha\in [\frac{2}{3},1)$. We will explain a hash construction that extends to any value of C and $\alpha\in (0,1)$, which is not necessarily of the form $1-\frac{1}{C}$. This construction reduces to the very and less sparse regimes hash constructions when $\alpha=\frac{1}{C}$, $\alpha\in (0,1/3]$, and $\alpha=1-\frac{1}{C}$, $\alpha\in [2/3,1)$, respectively.

In the very sparse regime $\alpha=\frac{1}{3}$, we have C=3 different hashes and for a non-zero spectral variable X with index $X_0^{n-1}=(r_0,r_1,r_2),\ h_i(X_0^{n-1})=r_i$ thus the output of different hashes depend on non overlapping parts of the binary index of X whereas for $\alpha=\frac{2}{3}$ the hash outputs are $(r_0,r_1),$ (r_1,r_2) and (r_0,r_2) which overlap on a portion of binary indices of length $\frac{n}{3}$. Intuitively, it is clear that in order to construct different hashes for $\alpha\in(\frac{1}{3},\frac{2}{3})$, we should start increasing the overlapping size of different hashes from 0 for $\alpha=\frac{1}{3}$ to $\frac{n}{3}$ for $\alpha=\frac{2}{3}$. We give the following construction for the hash functions

$$h_i(X_0^{n-1}) = X_{i\,t}^{i\,t+b}, i \in [C],$$

where $t=\frac{n}{C}$ and the values of the indices are computed modulo n, for example $X_n=X_0$. In the terminology of Section IV, we pick $\mathcal{H}_i=\Psi_b^T\Sigma_i^T\in\mathbb{F}_2^{k\times n}$, where $\Sigma_i\in\mathbb{F}_2^{n\times n}$ is the identity matrix with columns circularly shifted by (i+1)b to the left. It is clear that each hash is a surjective map from \mathbb{F}_2^n into $\mathbb{F}_2^{n\alpha}$. Therefore, if we pick $b=n\alpha$, the number of output bins in each hash is $B=2^{n\alpha}=N^{\alpha}=K$, thus the average number of non-zero variables per bin in every hash is equal to $\beta=\frac{K}{B}=1$. In terms of decoding performance for the intermediate values of $\alpha\in(\frac{1}{3},\frac{2}{3})$, one expects that the performance of the peeling decoder for this regime is between the very sparse regime $\alpha=\frac{1}{3}$ and the less sparse one $\alpha=\frac{2}{3}$.

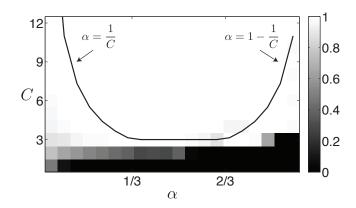


Fig. 7. Probability of success of the algorithm in the very sparse regime as a function of α and C. The dimension of the signal is $N=2^{22}$. The black line corresponds to $\alpha=\frac{1}{C}$ and $\alpha=1-\frac{1}{C}$ in the very and less sparse regimes, respectively. We fix $\beta=1$. The hashing matrices are deterministically picked as described in Section VII-D.

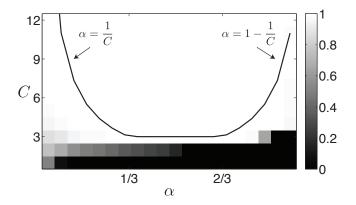


Fig. 8. Probability of success of the algorithm in the very sparse regime as a function of α and C. The dimension of the signal is $N=2^{22}$. The black line corresponds to $\alpha=\frac{1}{C}$ and $\alpha=1-\frac{1}{C}$ in the very and less sparse regimes, respectively. We fix $\beta=1$. The hashing matrices are picked at random for every trial.

VIII. EXPERIMENTAL RESULTS

In this section, we empirically evaluate 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 trials. We also provide a comparison of the run time of our algorithm and the standard Hadamard transform.

- Experiment 1: We fix the signal size to $N=2^{22}$ and run the algorithm 1000 times to estimate the success probability for $\alpha \in (0,\frac{1}{3}]$ and $1 \leq C \leq 12$. The hashing scheme used is as described in Section VII-D. Fig. 7 shows the simulation result. Albeit the asymptotic behavior of the error probability is only guaranteed for $C=(\frac{1}{\alpha}\vee\frac{1}{1-\alpha})$, we observe much better results in practice. Indeed, C=4 already gives a probability of success very close to one over a large range of α , and only up to C=6 seems to be required for the largest values of α .
- Experiment 2: We repeat here experiment 1, but instead of deterministic hashing matrices, we now pick Σ_i , $i \in [C]$,

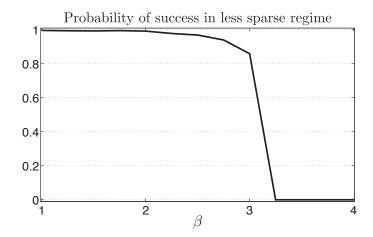


Fig. 9. Probability of success of the algorithm in the less sparse regime as a function of $\beta=K/B$. We fix $N=2^{22}$, $B=2^{17}$, C=4, and vary α in the range 0.7 to 0.9.

uniformly at random from $\mathrm{GL}(n,\mathbb{F}_2)$. The result is shown in Fig. 8. We observer that this scheme performs at least as well as the deterministic one.

- Experiment 3: In this experiment, we investigate the sensitivity of the algorithm to the value of the parameter $\beta=K/B$; the average number of non-zero coefficients per bin. As we explained, in our hash design we use $\beta\approx 1$. However, using larger values of β is appealing from a computational complexity point of view. For the simulation, we fix $N=2^{22}$, $B=2^{17}$, C=4, and vary α between 0.7 and 0.9, thus changing K and as a result β . Fig. 9 show the simulation results. For $\beta\approx 0.324$, the algorithm succeeds with probability very close to one. Moreover, for values of β larger than 3, success probability sharply goes to 0.
- Runtime measurement: We compare the runtime of the SparseFHT algorithm with a straightforward implementation of the fast Hadamard transform. The result is shown in Fig. 10 for $N=2^{15}$. SparseFHT performs much faster for $0<\alpha<2/3$.

It is also intersting to identify the range of α for which SparseFHT has a lower runtime than the conventional FHT. We define α^* , the largest value of α such that SparseFHT is faster than FHT for any lower value of α . That is

$$\alpha^* = \sup_{\alpha \in (0,1)} \{ \alpha : \forall \alpha' \le \alpha, \ T_{FHT}(n) > T_{SFHT}(\alpha',n) \},$$

where T_{FHT} and T_{SFHT} are the runtimes of the conventional FHT and SparseFHT, respectively. We plot α^* as a function of $n = \log_2 N$ in Fig. 11.

Remark 9. In the computation of the complexity in Section V-B, we have assumed that matrix-vector multiplications in \mathbb{F}_2^n can be done in O(1). In general, it is not true. However, the deterministic hashing scheme of the algorithm is nothing but a circular bit shift that can be implemented in a constant number of operations, independent of the vector size n.

If one is given Σ , some matrix from $\mathbb{F}_2^{n \times n}$, and its inverse transpose Σ^{-T} , the overall complexity of the algorithm would

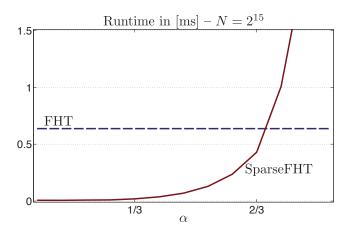


Fig. 10. Comparison of the Median runtime in ms of the SparseFHT and conventional FHT for $N=2^{15}$ and for different values of α . Confidence interval where found to be negligible and are omitted here. Lower runtime is better.

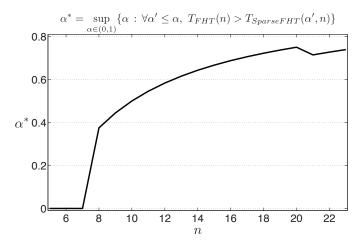


Fig. 11. In this figure, we plot $n=\log_2 N$ against α^* , the largest value of α such that SparseFHT runs faster than the conventional FHT for all values of α smaller or equal. When FHT is always faster, we simply set $\alpha^*=0$. Larger values are better.

nonetheless be unchanged. First, we observe that it is possible to compute the inner product of two vectors in constant time using bitwise operations and a small look-up table¹. Now, given the structure of Ψ_b , computing $\Sigma\Psi_b m$ in Algorithm 1 only requires $\log_2 K$ inner products. Thus the complexity of Algorithm 1 is unchanged. Finally, (7) can be split into pre-computing $\Sigma^{-T}\Psi_b k$ at the same time as we subsample the signal (in $O(\log_2 K)$), and computing the inner product between \hat{v} and the n-b first columns of Σ when doing the decoding $(O(\log_2 \frac{N}{K}))$.

IX. CONCLUSION

We presented a new algorithm to compute the Hadamard transform of a signal of length N which is K-sparse in the Hadamard domain. The algorithm presented has complexity $O(K\log_2K\log_2\frac{N}{K})$ and only requires $O(K\log_2\frac{N}{K})$ timedomain samples. We show that the algorithm correctly reconstructs the Hadamard transform of the signal with high probability asymptotically going to one.

¹http://graphics.stanford.edu/~seander/bithacks.html#ParityLookupTable

The performance of the algorithm is also evaluated empirically through simulation, and its speed is compared to that of the conventional fast Hadamard transform. We find that considerable speed-up can be obtained, even for moderate signal length (e.g. $N=2^{10}$) with reasonnable sparsity assumptions.

However, from the statement of Proposition 2, it will be apparent to the reader that the algorithm is absolutely not robust to noise. In fact, at very large signal size, the machine noise, using double-precision floating point arithmetic, proved to be problematic in the simulation. To make the algorithm fully practical, a robust estimator is needed to replace Proposition 2, and is, so far, left for future work.

REFERENCES

- W. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," in *Proceedings of the IEEE*, 1969, pp. 58–68.
- [2] 3GPP TS 25.213 V11.4.0 Release 11, "Spreading and modulation (fdd)," 2013.
- [3] K. J. Horadam, Hadamard Matrices and Their Applications. Princeton University Press, 2007.
- [4] S. Haghighatshoar and E. Abbe, "Polarization of the Rényi information dimension for single and multi terminal analog compression," arXiv preprint arXiv:1301.6388, 2013.
- [5] A. Hedayat and W. Wallis, "Hadamard matrices and their applications," The Annals of Statistics, pp. 1184–1238, 1978.
- [6] 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.
- [7] 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.
- [8] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss, "Near-optimal sparse fourier representations via sampling," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, 2002, pp. 152–161.
- [9] 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.
- [10] D. Lawlor, Y. Wang, and A. Christlieb, "Adaptive sub-linear Fourier algorithms," arXiv.org, Jul. 2012.
- [11] 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.
- [12] —, "Nearly optimal sparse Fourier transform," *Proceedings of the* 44th symposium on Theory of Computing, pp. 563–578, 2012.
- [13] 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.
- [14] S. Pawar and K. Ramchandran, "A hybrid DFT-LDPC framework for fast, efficient and robust compressive sensing," in *Communication, Con*trol, and Computing (Allerton), 2012 50th Annual Allerton Conference on, 2012, pp. 1943–1950.
- [15] —, "Computing a k-sparse n-length Discrete Fourier Transform using at most 4k samples and O(k log k) complexity," arXiv.org, May 2013.
- [16] T. Richardson and R. L. Urbanke, Modern coding theory. Cambridge University Press, 2008.
- [17] N. C. Wormald, "Differential Equations for Random Processes and Random Graphs," *The Annals of Applied Probability*, vol. 5, no. 4, pp. 1217–1235, Nov. 1995.
- [18] 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.

APPENDIX A

PROOF OF THE PROPERTIES OF THE WHT

A. Proof of Property 1

$$\sum_{m \in \mathbb{F}_2^n} (-1)^{\langle k, m \rangle} x_{m+p} = \sum_{m \in \mathbb{F}_2^n} (-1)^{\langle k, m+p \rangle} x_m.$$

And the proof follows by taking $(-1)^{\langle k, p \rangle}$ out of the sum and recognizing the Hadamard transform of x_m .

B. Proof of Property 2

As we explained, it is possible to assign an $N \times N$ matrix Π to the permutation π as follows

$$(\Pi)_{i,j} = \begin{cases} 1 & \text{if } j = \pi(i) \Leftrightarrow i = \pi^{-1}(j) \\ 0 & \text{otherwise.} \end{cases}.$$

Let π_1 and π_2 be the permutations associated with Π_1 and Π_2 . Since $(H_N)_{i,j} = (-1)^{\langle i,j \rangle}$, the identity (1) implies that

$$(-1)^{\langle \pi_2(i), j \rangle} = (-1)^{\langle i, \pi_1^{-1}(j) \rangle}.$$

Therefore, for any $i, j \in \mathbb{F}_2^n$, π_1, π_2 must satisfy $\langle \pi_2(i), j \rangle = \langle i, \pi_1^{-1}(j) \rangle$. By linearity of the inner product, one obtains that

$$\langle \pi_2(i+k), j \rangle = \langle i+k, \pi_1^{-1}(j) \rangle$$

= $\langle i, \pi_1^{-1}(j) \rangle + \langle k, \pi_1^{-1}(j) \rangle$
= $\langle \pi_2(i), j \rangle + \langle \pi_2(k), j \rangle$.

As $i, j \in \mathbb{F}_2^n$ are arbitrary, this implies that π_2 , and by symmetry π_1 , are both linear operators. Hence, all the permutations satisfying (1) are in one-to-one correspondence with the elements of $GL(n, \mathbb{F}_2)$.

C. Proof of Property 3

Since Σ is non-singular, then Σ^{-1} exists. It follows from the definition of the WHT that

$$\begin{split} \sum_{m \in \mathbb{F}_2^n} (-1)^{\langle k \,,\, m \rangle} x_{\Sigma m} &= \sum_{m \in \mathbb{F}_2^n} (-1)^{\langle k \,,\, \Sigma^{-1} m \rangle} x_m \\ &= \sum_{m \in \mathbb{F}_2^n} (-1)^{\langle \Sigma^{-T} k \,,\, m \rangle} x_m. \end{split}$$

This completes the proof.

D. Proof of Property 4

$$\begin{split} \sum_{m \in \mathbb{F}_2^b} (-1)^{\langle k \,,\, m \rangle} x_{\Psi_b m} \\ &= \quad \frac{1}{\sqrt{N}} \sum_{m \in \mathbb{F}_2^b} (-1)^{\langle k \,,\, m \rangle} \sum_{p \in \mathbb{F}_2^n} (-1)^{\langle \Psi_b m \,,\, p \rangle} X_p \\ &= \quad \frac{1}{\sqrt{N}} \sum_{p \in \mathbb{F}_2^n} X_p \sum_{m \in \mathbb{F}_2^b} (-1)^{\langle m \,,\, k + \Psi_b^T p \rangle}. \end{split}$$

In the last expression, if $p = \Psi_b k + i$ with $i \in \mathcal{N}(\Psi_b^T)$ then it is easy to check that the inner sum is equal to B, otherwise it is equal to zero. Thus, by proper renormalization of the sums one obtains the proof.

APPENDIX B PROOF OF PROPOSITION 2

We first show that if multiple coefficients fall in the same bin, it is very unlikely that 1) is fulfilled. Let $\mathcal{I}_k = \{j \mid \mathcal{H}j = k\}$ be the set of variable indices hashed to bin k. This set is finite and its element can be enumerated as $\mathcal{I}_k = \{j_1, \ldots, j_{\frac{N}{B}}\}$. We show that a set $\{X_j\}_{j\in\mathcal{I}_k}$ is very unlikely, unless it contains only one non-zero element. Without loss of generality, we consider $\sum_{j\in\mathcal{I}_k} X_j = 1$. Such $\{X_j\}_{j\in\mathcal{I}_k}$ is a solution of

$$\begin{bmatrix} 1 & \cdots & 1 \\ (-1)^{\langle \sigma_1, j_1 \rangle} & \cdots & (-1)^{\langle \sigma_1, j_{\frac{N}{B}} \rangle} \\ \vdots & \ddots & \vdots \\ (-1)^{\langle \sigma_{n-b}, j_1 \rangle} & \cdots & (-1)^{\langle \sigma_{n-b}, j_{\frac{N}{B}} \rangle} \end{bmatrix} \begin{bmatrix} X_{j_1} \\ \vdots \\ X_{j_{\frac{N}{B}}} \end{bmatrix} = \begin{bmatrix} 1 \\ \pm 1 \\ \vdots \\ \pm 1 \end{bmatrix},$$

where $\sigma_i, i \in \{1, \dots, n\}$ denotes the i-th column of the matrix Σ . The left hand side matrix in the expression above, is $(n-b+1) \times 2^{n-b}$. As $\sigma_1, \dots, \sigma_{n-b}$ form a basis for \mathcal{I}_k , all the columns are different and are (omitting the top row) the exhaustive list of all 2^{n-b} possible ± 1 vectors. Thus the right vector is always one of the columns of the matrix and there is a solution with only one non-zero component (1-sparse solution) to this system whose support can be uniquely identified. Adding any vector from the null space of the matrix to this initial solution yields another solution. However, as we will show, due to its structure this matrix is full rank and thus its null space has dimension $2^{n-b}-n+b-1$. Assuming a continuous distribution on the non-zero components X_i , the probability that $\{X_i\}_{i\in\mathcal{I}_k}$ falls in this null space is zero.

To prove that the matrix is indeed full rank, let us first focus on the rank of the sub-matrix obtained by removing the first row. This submatrix itself always contains $M = -2I + 11^T$, where I is the identity matrix of order n-b and 1 is the allone vector of dimension (n-b). One can simply check that M is a symmetric matrix, thus by spectral decomposition, it has n-b orthogonal eigen-vectors $v_i, i \in [n-b]$. It is also easy to see that the normalized all-one vector $v_0 = \frac{1}{\sqrt{n-b}}$ of dimension n-b is an eigen-vector of M with eigen-value $\lambda_0 = n - b - 2$. Moreover, assuming the orthonormality of the eigen-vectors, it results that $v_i^T M v_i = \lambda_i = -2$, where we used $v_i^T \mathbb{1} = v_i^T v_0 = 0$ for $i \neq 0$. Thus, for $n - b \neq 2$ all of the eigen-vlaues are non-zero and M is invertible, which implies that the sub-matrix resulted after removing the first row is full rank. In the case where n - b = 2, one can notice that the Hadamard matrix of size 2 will be contained as a submatrix, and thus the matrix will be full rank.

Now it remains to prove that initial matrix is also full rank with a rank of n-b+1. Assume that the columns of the matrix are arranged in the lexicographical order such that neglecting the first row, the first and the last column are all 1 and all -1. If we consider any linear combination of the rows except the first one, it is easy to see that the first and the last element in the resulting row vector have identical magnitudes but opposite signs. This implies that the all-one row cannot be written as a linear combination of the other rows of the matrix. Therefore, the rank of the matrix must be n-b+1.

To prove (7), let Σ_L and Σ_R be the matrices containing respectively the first n-b and the last b columns of Σ , such that $\Sigma = [\Sigma_L \Sigma_R]$. If there is only one coefficient in the bin, then (6) implies that $\hat{v} = [(j^T \Sigma_L) \ 0]^T$. Using definitions (2) and (5), we obtain that $\Psi_b \mathcal{H} j = [0 \ (j^T \Sigma_R)]^T$. We observe that they sum to $\Sigma^T j$ and the proof follows.

APPENDIX C PROOF OF PROPOSITION 3

For $t \in [K]$, let H_t denote the size of the random set obtained by picking t objects from [N] independently and uniformly at random with replacement. Let a_t and v_t denote the average and the variance of H_t for $t \in [K]$. It is easy to see that $\{H_t\}_{t \in [K]}$ is a Markov process. Thus, we have

$$\mathbb{E}[H_{t+1} - H_t | H_t] = (1 - H_t / N),$$

because the size of the random set increases if an only if we choose an element from $[N]\backslash H_t$. This implies that $a_{t+1}=1+\gamma a_t$, where $\gamma=1-\frac{1}{N}$. Solving this equation we obtain that

$$a_t = \sum_{r=0}^{t} \gamma^r = \frac{1 - \gamma^{t+1}}{1 - \gamma} = N(1 - \gamma^{t+1}).$$
 (9)

In particular, $a_K = N(1-(1-\frac{1}{N})^K)$, which implies that $\mathbb{E}\left[\frac{H_K}{K}\right] = \frac{N}{K}(1-(1-\frac{1}{N})^K)$. One can check that for $K=N^\alpha$, $0<\alpha<1$, as N tends to infinity $\mathbb{E}\left[\frac{H_K}{K}\right]$ converges to 1. To find the variance of H_t , we use the formula

$$Var(H_{t+1}) = Var(H_{t+1}|H_t) + Var(\mathbb{E}[H_{t+1}|H_t)]).$$
 (10)

Therefore, we obtain that

$$\operatorname{Var}(\mathbb{E}\left[H_{t+1}|H_t|\right]) = \operatorname{Var}(1+\gamma H_t) = \gamma^2 v_t. \tag{11}$$

Moreover, for the first part in (10), we have

$$\operatorname{Var}(H_{t+1}|H_t) = \mathbb{E}_{H_t} \left\{ \operatorname{Var}(H_{t+1}|H_t = h_t) \right\}$$

$$= \mathbb{E}_{H_t} \left\{ \operatorname{Var}(H_{t+1} - H_t|H_t = h_t) \right\}$$

$$\stackrel{(I)}{=} \mathbb{E} \left[\frac{H_t}{N} \left(1 - \frac{H_t}{N} \right) \right]$$

$$= \frac{a_t}{N} + \frac{a_t^2 + v_t}{N^2}, \tag{12}$$

where in (I) we used the fact that given H_t , $H_{t+1} - H_t$ is a Bernoulli random variable with probability $\frac{H_t}{N}$, thus its variance in equal to $\frac{H_t}{N}(1-\frac{H_t}{N})$. Combining (11) and (12), we obtain that

$$v_{t+1} = \left(\gamma^2 + \frac{1}{N^2}\right)v_t + \frac{a_t}{N}\left(1 + \frac{a_t}{N}\right).$$
 (13)

From (9), it is easy to see that a_t is increasing in t. Moreover, from (13) it is seen that v_{t+1} is increasing function of a_t , thus, if we consider the following recursion

$$w_{t+1} = \left(\gamma^2 + \frac{1}{N^2}\right) w_t + \frac{a_K}{N} \left(1 + \frac{a_K}{N}l\right),$$

then for any $t \in [K]$, $v_t \leq w_t$. As w_t is also an increasing sequence of t, we obtain that

$$v_K \le w_K \le w_\infty = \frac{a_K}{N} \left(1 + \frac{a_K}{N} \right) / \left(1 - \gamma^2 - \frac{1}{N^2} \right)$$
$$= \frac{a_K}{2} \left(1 + \frac{a_K}{N} \right) / \left(1 - \frac{1}{N} \right).$$

Using Chebyshev's inequality, we obtain that for any $\epsilon > 0$

$$\mathbb{P}\left\{\frac{H_K}{K} \geq (1+\epsilon)\right\} \leq \frac{v_K}{K^2(\epsilon+1-\frac{a_K}{K})^2} = \Theta\left(\frac{1}{\epsilon^2 K}\right).$$

Obviously, $\frac{H_K}{K} \le 1$, thus $\frac{H_K}{K}$ converges to 1 in probability as N and as a result K tend to infinity.

APPENDIX D PROOF OF PROPOSITION 9

Let S be any set of variable nodes of size at most ηK , where we will choose η later, the average degree of variable nodes in S is C. Let $\mathcal{N}_i(S), i \in [C]$ be the check neighbors of \mathcal{G} in hash i. If for at least one of the hashes $i \in [C]$, $|\mathcal{N}_i(S)| > \frac{|S|}{2}$, it results that there is at least one check node of degree 1 (a singleton) among the neighbors, which implies that the peeling decoder can still proceed to decode further variable nodes.

Let \mathcal{E}_s^i denote the event that a specific subset A of size s of variable nodes has at most $\frac{s}{2}$ check neighbors in hash i. Also let $\mathcal{E}_s = \bigcap_{i=1}^C \mathcal{E}_s^i$. By the construction of \mathcal{G} , it is easy

to see that $\mathbb{P}\{\mathcal{E}_s\} = \prod_{i=1}^C \mathbb{P}\left\{\mathcal{E}_s^i\right\}$. Let T be any subset of check nodes in hash i of size $\frac{s}{2}$. The probability that all the neighbors of A in hash i belong to a specific set T of size $\frac{s}{2}$ is equal to $(\frac{s}{2B})^s$. Taking a union bound over $\binom{B}{s/2}$ of all such sets, it is seen that $\mathbb{P}\left\{\mathcal{E}_s\right\} \leq \binom{B}{s/2}(\frac{s}{2B})^s$, which implies that $\mathbb{P}\left\{\mathcal{E}_s^i\right\} \leq \left(\binom{B}{s/2}(\frac{s}{2B})^s\right)^C$. Taking a union bound over all possible subsets of size s of variables, we obtain that

$$\mathbb{P}\left\{F_{s}\right\} \leq \binom{K}{s} \mathbb{P}\left\{\mathcal{E}_{s}\right\} \leq \binom{K}{s} \left(\binom{B}{s/2} \left(\frac{s}{2B}\right)^{s}\right)^{C}$$
$$\leq \left(\frac{eK}{s}\right)^{s} \left(\frac{2eB}{s}\right)^{sC/2} \left(\frac{s}{2B}\right)^{sC} \leq \frac{u^{s}s^{s(C/2-1)}}{K^{s(C/2-1)}},$$

where $u=e^{C/2+1}(\frac{\beta}{2})^{C/2}$ and where F_s denotes the event that the peeling decoder fail to decode a set of variables of size s. We also used the fact that for $n\geq m$, $\binom{n}{m}\leq (\frac{n\,e}{m})^m$ and $\mathbb{P}\left\{F_1\right\}=\mathbb{P}\left\{F_2\right\}=0$. Selecting $\eta=\frac{1}{2u^{2/(C-2)}}$ and applying the union bound, we obtain that

$$\begin{split} \mathbb{P}\left\{F\right\} & \leq \sum_{s=1}^{\eta K} \mathbb{P}\left\{F_{s}\right\} = \sum_{s=3}^{\eta K} \mathbb{P}\left\{F_{s}\right\} = \sum_{s=3}^{\eta K} \frac{u^{s} s^{s(C/2-1)}}{K^{s(C/2-1)}} \\ & = O\left(\frac{1}{K^{3(C/2-1)}}\right) + \sum_{s=4}^{\eta K} \left(\frac{1}{2}\right)^{s} = O\left(\frac{1}{K^{3(C/2-1)}}\right), \end{split}$$

where F is the event that the peeling decoder fails to decode all the variables. This completes the proof.