A Fast Hadamard Transform for Signals With Sublinear Sparsity in the Transform Domain

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

***Abstract*— In this paper, we design a new iterative low-complexity algorithm for computing the Walsh–Hadamard transform (WHT) of an N dimensional signal with a K-sparse WHT. We suppose that N is a power of two and $K = O(N^{\alpha})$, scales sublinearly in N for some $\alpha \in (0, 1)$. Assuming a random support model for the nonzero transform-domain components, our algorithm reconstructs the WHT of the signal with a sample complexity $O(K \log_2(N/K))$ and a computational complexity $O(K \log_2(K) \log_2(N/K))$. Moreover, the algorithm succeeds with a high probability approaching 1 for large dimension N. Our approach is mainly based on the subsampling (aliasing) property of the WHT, where by a carefully designed subsampling of the time-domain signal, a suitable aliasing pattern is induced in the transform domain. We treat the resulting aliasing patterns as parity-check constraints and represent them by a bipartite graph. We analyze the properties of the resulting bipartite graphs and borrow ideas from codes defined over sparse bipartite graphs to formulate the recovery of the nonzero spectral values as a peeling decoding algorithm for a specific sparse-graph code transmitted over a binary erasure channel. This enables us to use tools from coding theory (belief-propagation analysis) to characterize the asymptotic performance of our algorithm in the very sparse ($\alpha \in (0, 1/3]$) and the less sparse ($\alpha \in (1/3, 1)$) regime. Comprehensive simulation results are provided to assess the empirical performance of the proposed algorithm.**

***Index Terms*— Walsh-Hadamard transform, sparse FFT, sub-linear sparsity, peeling decoder, density evolution.**

I. Introduction

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

A number of recent publications have addressed the problem of computing the DFT of an N dimensional signal when it is sparse in the frequency domain [8]–[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 under the sparsity assumption. Such algorithms are generally known as *sparse* FFT (sFFT) algorithms. In [13], Ghazi et al. developed a very low-complexity algorithm for computing the 2D-DFT of a $\sqrt{N} \times \sqrt{N}$ signal by extending the results of [12]. In a similar line of work, Pawar and Ramchandran [14], [15] used the subsampling property of the DFT to develop a low-complexity algorithm for recovering the nonzero frequency-domain components of a signal by using ideas from sparse-graph codes [16].

In this paper, we develop a novel algorithm for computing the WHT of an N dimensional signal with a sub-linear sparsity in the Hadamard domain. More precisely, we assume that the number of nonzero Hadamard-domain components $K = O(N^{\alpha})$ scales sub-linearly in N for some $\alpha \in (0, 1)$. We first develop some useful properties of the WHT, specially the subsampling and the modulation property, which plays a vital role in the development of the algorithm. In particular, we show that 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 nonzero components if more than one of them are involved in the resulting linear combination. Similar to [15] and by borrowing ideas from sparse-graph codes, we construct a bipartite graph by considering the nonzero values in the transform domain as variable nodes and, by interpreting every induced aliasing pattern as a parity check constraint over the variables in the graph. We analyze the structure of the resulting graph assuming a random support model for the nonzero coefficients in the transform domain. Moreover, we give an iterative peeling decoder to recover the nonzero components. In short, our proposed sparse fast Hadamard transform (SparseFHT) consists of a set of deterministic linear hash functions (explicitly constructed) and an iterative peeling decoder that uses the hash outputs to recover the nonzero transform-domain variables. It recovers the K-sparse WHT of the signal with a sample complexity (number

Manuscript received December 30, 2013; accepted December 27, 2014. Date of publication February 16, 2015; date of current version March 13, 2015. This work was supported by the ERC Advanced Investigators Grant Sparse Sampling Theory, Algorithms and Applications under Grant 247006. This paper was presented at the 51st Annual Allerton Conference on Communication, Control, and Computing in 2013.

The authors are with the School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Lausanne 1015, Switzerland (e-mail: robin.scheibler@epfl.ch; saeid.haghighatshoar@tu-berlin.de; martin.vetterli@epfl.ch).

Communicated by Y. Ma, Associate Editor for Signal Processing.

The implementation of the algorithm used in the numerical experiments is available online at http://lcav.github.io/SparseFHT/.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2015.2404441

of required time-domain samples) $O(K\log_2(\frac{N}{K}))$, total computational complexity $O(K\log_2(K)\log_2(\frac{N}{K}))$ and with a high probability approaching 1 for large dimension N.

A closely related work, in spirit, to ours is the work on one-way functions by Goldreich and Levin [17] and Goldreich [18] in theoretical computer science, where it was shown that the support recovery of the nonzero transform-domain coefficients is reduced to recovering the value of inner products of boolean vectors. In particular, an efficient algorithm was developed to speed up this computation [18]. Although our signal model and proposed recovery algorithm is completely different, we believe that our results would be of interest in these areas as well.

Notations and Preliminaries: For an integer m, the set of all integers $\{0, 1, \ldots, m-1\}$ is denoted by $[m]$. We use the small letter x for the time domain and the capital letter X for the transform-domain signal. For an N dimensional real-valued vector v, with $N = 2^n$ 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 to 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, where the addition of two vectors is 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 the exact mathematical sense: for example, if $u = [1, 1, 1, 1]^T$, then $\langle u\,,\, u\rangle = 0$, but $u \neq 0$. We often use Σ for a matrix with entries in $\mathbb{F}_2$. We denote by Σ^T the transpose of the matrix Σ. When Σ is non-singular, we use the shorthand notation Σ^{-T} for $(\Sigma^T)^{-1}$. The null space of a matrix Σ is denoted by $\mathcal{N}(\Sigma) = \{u \mid \Sigma u = 0\}$.

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

Definition 1: Let $\mathcal{S}$ be a class of signals $\mathcal{S} = \cup_{N=1}^{\infty}\mathcal{S}_N$, where $\mathcal{S}_N$ denotes the subclass of all N-dimensional signals. $\mathcal{S}$ is said to have a sub-linear sparsity if there is some $\alpha \in (0, 1)$ such that $K_N = O(N^\alpha)$, where K_N denotes the sparsity of the collection $\mathcal{S}_N$. We call α the sparsity index of class $\mathcal{S}$.

II. Main Results

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

Theorem 1: Let $\alpha \in (0, 1)$ be a fixed number. Suppose $N = 2^n$ is a power of two and assume $K = N^\alpha$. 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 selected uniformly at random among all possible $\binom{N}{K}$ subsets of $[N]$ of size K. In addition, let the magnitude of the nonzero components of X be independently sampled from some arbitrary continuous distribution (that does not need to be known). Then, 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 needed in order to successfully decode all the nonzero spectral components or announce a decoding failure is $O(CK\log_2(K)\log_2(\frac{N}{K}))$.
3) *Success Probability:* The algorithm correctly computes the K-sparse WHT X with a very high probability asymptotically approaching 1 as the dimension of the signal N tends to infinity, where the probability is taken over all random selections of the support of X. More importantly, the algorithm can find out whether the recovery succeeds or fails.

To prove Theorem 1, we distinguish between the very sparse case ($\alpha \in (0, \frac{1}{3}]$) and less sparse one ($\alpha \in (\frac{1}{3}, 1)$), where we implicitly assume that the algorithm knows the value of α, which might not be possible in some applications.

Fortunately, there is some underlying monotonicity in our algorithm that helps to solve this problem. More precisely, the algorithm designed for a specific value of $\alpha = \alpha^*$ can successfully recover all the transform-domain signals with sparsity index less that α^*. Thus, even if the value of α is unknown, we only need to design the algorithm for the largest possible value of α. However, the drawback is that the resulting sample and computational complexity might get much higher than the optimal algorithm that knows the exact value of α.

Fortunately, this problem can also be solved to obtain an optimal algorithm that does not need to know the value of α. The main observation is that in the sub-linear sparsity regime, where the number of nonzero components scales like $K = O(N^\alpha)$, the resulting sample and time complexity are on the order of $O(N^\alpha\log_2(N))$ and $O(N^\alpha\log_2(N)^2)$ respectively. This implies that for $\alpha_1 < \alpha_2$, and for a sufficiently large signal dimension N, the algorithm designed for α_1 has negligible sample and computational complexity compared with the one designed for α_2. Hence, we can use an adaptive strategy. We design our algorithm for small values of α. The algorithm will find out if the recovery was successful. If not, we increase the value of α and run a new algorithm for the new value of α, and we continue until the recovery is successful. In this way, we get a sample and computational complexity comparable with the algorithm that knows the exact value of α. The only drawback is that, in the adaptive scheme, the required number of time-domain samples gradually increases as we try larger value of α, thus it might not be useful in applications in which the number of samples should be a priori fixed.

Remark 1: In the very sparse regime ($\alpha \in (0, \frac{1}{3})$), we prove that for any value of α the success probability of the optimally designed algorithm is at least $1 - O(1/N^{3\alpha(C/2-1)})$, with $C = \lfloor\frac{1}{\alpha}\rfloor$ where for $u \in \mathbb{R}_+$, $\lfloor u\rfloor = \max\{n \in \mathbb{Z} : n \leq u\}$. 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

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

where $k \in \mathbb{F}_2^n$ denotes the corresponding binary index of the transform-domain component. Also, throughout the paper, borrowing some terminology from the DFT, we call x the time-domain signal and we refer to X as the transform-domain, Hadamard-domain, frequency-domain, or spectral-domain signal. Also, note that the WHT is invertible and its inverse is given by the same expression as in Eq. (1) except that the roles of x_m and X_k are interchanged.

Notice that with our notation both the time-domain signal $x : \mathbb{F}_2^n \to \mathbb{R}$ and the transform-domain signal $X : \mathbb{F}_2^n \to \mathbb{R}$ are functions from the index set $\mathbb{F}_2^n$ to reals. Therefore, the WHT given by the Eq. (1) maps the function x (time-domain signal) onto the function X (transform-domain signal). For simplicity of notation, we will use x_m for the time-domain and X_k for the frequency-domain functions with the convention that both m and k belong to the index set $\mathbb{F}_2^n$.

A. Basic Properties

In this section, we review some of the basic properties of the WHT. Some of the properties are not directly used in the paper, but we include 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

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

The next property is more subtle and enables us 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 $\pi_1, \pi_2 : [N] \to [N]$ with corresponding permutation matrices Π_1, Π_2 such that

$$\Pi_2 H_N = H_N \Pi_1, \tag{2}$$

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 (2) 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 (2) are described by the elements of

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

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

Remark 2: 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 $N!$ permutations over $[N]$.

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

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

Notice that $y_m = x_{\Sigma m}$ is the function given by the composition of the function x and the index transformation i_Σ, i.e., $y = x \circ i_\Sigma$, where for $m \in \mathbb{F}_2^n$, $i_\Sigma(m) = \Sigma m$ is the multiplication of the matrix Σ with the index vector m. Moreover, 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 signal obtained by permuting the components of the signal x_m.

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

$$\Psi_b = \begin{bmatrix} \mathbf{0}_{b \times (n-b)} & I_b \end{bmatrix}^T, \tag{3}$$

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

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

where $m, k \in \mathbb{F}_2^b$ denote the corresponding binary indices of the time and frequency components, and $x_{\Psi_b m}$ is a $B = 2^b$ dimensional signal labelled with $m \in \mathbb{F}_2^b$.

Recall that by our notation, $y_m = x_{\Psi_b m}$ is a function $y : \mathbb{F}_2^b \to \mathbb{R}$ given by $y = x \circ i_{\Psi_b}$, where $i_{\Psi_b} : \mathbb{F}_2^b \to \mathbb{F}_2^n$ is the index transformation given by $i_{\Psi_b}(m) = \Psi_b m$. It is not difficult to check that $\Psi_b m$, which is the multiplication of $m \in \mathbb{F}_2^b$ with the matrix Ψ_b of dimension $n \times b$, gives an index in $\mathbb{F}_2^n$ which is the right argument for the function x. Also, index $\Psi_b k + j$ with $j \in \mathcal{N}\left(\Psi_b^T\right)$ gives the suitable index for the function X. Notice that Property 4 can be simply applied for any matrix Ψ_b that subsamples any set of indices of length b but not necessarily the b last ones.

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

In a general downsampling procedure, we can replace the frozen indices by an arbitrary but fixed binary pattern. The only difference is that, instead of summing the aliased spectral components, we should also take into account the suitable

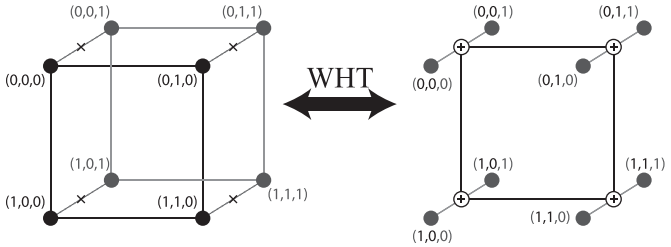

Fig. 1. Illustration of the downsampling property on a hypercube for $N = 2^3$. The two cubes represent the time-domain (left) and the Hadamard-domain (right) signal. We decide to drop all the nodes whose third coordinate is '1'. We illustrate this by adding a '×' on the edges connecting those vertices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

$\{+, -\}$ sign patterns, i.e., we have

$$x_{\Psi_b m+p} \quad \overset{\text{WHT}}{\longleftrightarrow} \quad \sqrt{\frac{B}{N}} \sum_{j \in \mathcal{N}(\Psi_b^T)} (-1)^{\langle p\,,\, j\rangle} X_{\Psi_b k+j}, \qquad (4)$$

where p is a binary vector of length n with b zeros at the end. To visualize this property, consider Fig. 1, where we have a signal over a 3-dimensional cube and we subsample it along the third dimension, i.e., we keep only 4 variables with the third index equal to 0. Notice that these variables lie on a 2-dimensional (square) face of the cube that corresponds to a subsampling with $p = 000$. Instead, we can use $p = 001$ for subsampling and this value of p will select all 4 variables on the other face of the cube corresponding to those variables with the third index equal to 1. This face of the cube is a square parallel to the square corresponding to $p = 000$.

IV. Hadamard Hashing Algorithm

By applying the basic properties of the WHT, we can design suitable hash functions in the spectral domain. The main idea is that, to compute the output of a hash function, we do not need to have access to the spectral components because it can be computed by low-complexity operations that are directly applied to 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

$$U_{\Sigma,p}(k) = \sum_{j \in \mathbb{F}_2^n \,|\, \mathcal{H} j = k} X_j\, (-1)^{\langle p\,,\, j\rangle}, \qquad (5)$$

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

$$\mathcal{H} = \Psi_b^T \Sigma^T, \qquad (6)$$

where Ψ_b is as in (3). Note that the index of components in the sum (5) can be explicitly written as a function of the bin index k as follows:

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

The proof simply follows from the properties 1, 3, and 4.

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.

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

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

Using Proposition 1, we give Algorithm 1 for computing the hashed Hadamard spectrum. It chooses B bins for hashing the spectrum, chooses B samples of the time-domain signal according to the subsampling parameters Σ and p, and uses an FFT-like fast Hadamard transform (FHT) to compute the hash output with $O(B \log_2 B)$ operations.

A. Properties of Hadamard Hashing

In this part, we review some properties of the hashing algorithm that are crucial for developing an iterative peeling decoding algorithm that recovers the nonzero spectral values. We mainly explain how it is possible to identify collisions between the nonzero spectral coefficients that are hashed to the same bin and 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 the spectral variables hashed to bin k (those whose index j satisfies $h_\Sigma(j) = \mathcal{H} j = k$); 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 nonzero component, it is easy to see that $|r_{\Sigma,p}(k)| = 1$ for *'any value'* of p. However, if there is more than one nonzero component in the sum, and assuming that those nonzero values are jointly sampled from a continuous distribution, we 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 enable us to detect whether there is only one, or more than one nonzero component 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

$$\left\langle p\,,\, j' \right\rangle = \mathbb{1}_{\{r_{\Sigma,p}(k) < 0\}}, \qquad (7)$$

where $\mathbb{1}_{\{t<0\}} = 1$ 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 nonzero spectral coefficient in the corresponding hash bin, it is even possible to identify the position of this nonzero component. We formalize this result in the following proposition proved in Appendix B.

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

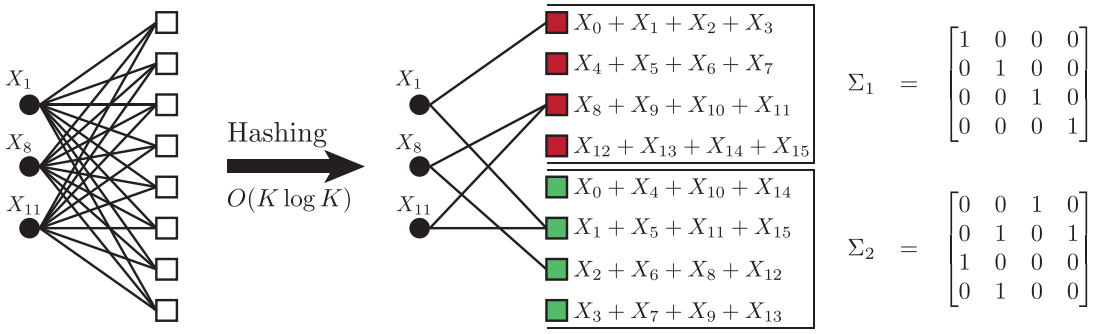

Fig. 2. On the left, bipartite graph representation of the WHT for $N = 8$ and $K = 3$. On the right, the underlying bipartite graph after applying $C = 2$ different hashing produced by plugging Σ_1, Σ_2 in (6) with $B = 4$. The variable nodes ($\bullet$) are the nonzero spectral values to be recovered. The white check nodes ($\square$) are the original time-domain samples. The colored squares are new check nodes after applying Algorithm 1.

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

$$\hat{v}_d = \begin{cases} \mathbb{1}_{\{r_{\Sigma,\sigma_d}(k)<0\}} & d \in [n-b], \\ 0 & \text{otherwise}, \end{cases} \tag{8}$$

the index of the unique nonzero value is given by

$$j = \Sigma^{-T}(\Psi_b\, k + \hat{v}). \tag{9}$$

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

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 that recovers the nonzero spectral components, and analyze its computational complexity.

A. Explanation of the Algorithm

In coding theory, it is common to represent a code over a bipartite graph with variable and check nodes. There is exactly one variable node per code bit and one check node per parity check constraint in the code. A variable node is connected to a check node if and only if it appears in the parity check equation corresponding to that check node. Such a graph is called *sparse* if the number of edges is in the order of the number of nodes.

We can slightly extend the bipartite graph representation to include the WHT. To explain this further, let us consider Eq. (1). From the symmetry of WHT, we have

$$x_m = \frac{1}{\sqrt{N}} \sum_{k \in \mathbb{F}_2^n} (-1)^{\langle m,\, k \rangle} X_k. \tag{10}$$

Eq. (10) states that knowing a time-domain sample x_m puts a linear constraint on the spectral-domain components X_k. Using the terminology of coding theory, we interpret these linear constraints as parity check constraints over X_k. For example, for $m = 0$, the resulting parity check equation implies that the sum of all the components of X must be equal to the first time-domain sample x_0 multiplied by $\sqrt{N}$. We associate a bipartite graph representation as follows: we associate variable nodes to the components of X (code bits in coding theory), and corresponding to every known time-domain sample, we add a check node to represent the resulting linear constraint. In particular, if we only keep the nonzero spectral variables, we obtain the induced bipartite graph over these variables. With this picture in mind, we can formulate the recovery of the nonzero spectral values as a decoding problem over this induced bipartite graph.

It is not difficult to see that for the WHT, the induced bipartite graph on the nonzero spectral values is a complete (dense) bipartite graph because any variable node is connected to all the check nodes. This has been depicted in the left part of Fig. 2, where $\{X_1, X_8, X_{11}\}$ are the only nonzero variables in the spectral domain and each check constraint corresponds to the value of a specific time-domain sample. It is also implicitly assumed that the support (position of nonzero variables) of X is known, e.g., $\{1, 8, 11\}$ in Fig. 2. At the moment, it is not clear how we can obtain the position of these nonzero variables. In the final version of the algorithm, this is done by applying Proposition 2.

For codes on sparse bipartite graphs, there exists a collection of low-complexity belief propagation algorithms to efficiently recover the code bits given the value of check nodes observed via a noisy channel [16]. Unfortunately, the graph corresponding to WHT is dense, and probably not suitable for any of these belief propagation algorithms.

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 must be clear from this picture. The idea is that via hashing, we can obtain a new representation with a *sparse* bipartite graph. In some sense, hashing *'sparsifies'* the underlying bipartite graph. It is also important to note that in the former representation, the output value of every parity check equation is an already known time-domain sample of the signal, thus no extra effort is necessary to compute them. For the latter representation obtained via hashing, the value of parity checks (hash outputs) are not a priori known but fortunately, they can be computed by using low-complexity operations on a small subset of time-domain samples as explained in Proposition 1.

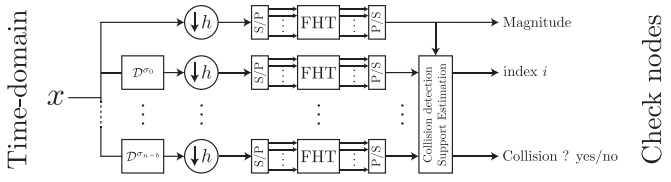

Fig. 3. A block diagram of the SparseFHT algorithm in the time domain. The downsampling plus small size low-complexity FHT blocks compute different hash outputs. Delay blocks denote an index shift by σ_i before hashing. The S/P and P/S are serial-parallel and parallel-serial blocks to emphasize that 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.

We propose the following iterative algorithm to recover the nonzero spectral variables over the bipartite graph induced by hashing. The algorithm first tries to find a degree-one check node. Using the terminology of [15], we call such a check node a *singleton*. This singleton check is connected to only one nonzero variable. Using Proposition 2, we can find the position and the value of this nonzero variable. This enables us to subtract (peel off) this variable from all the other check nodes that are connected to it, thus we can remove this variable node from the graph along with all the edges that are connected to it. Consequently, the degree of all the check nodes that are connected to this variable node decreases by one, thus there is a chance that another singleton be found. Also, removing the edges, creates an isolated (degree zero) check node that we call a *zeroton*.

The algorithm proceeds to peel off one singleton at a time until all the check nodes are zerotons (decoding succeeds) or all the remaining check nodes have degrees greater than one (we call them *multiton*) and the algorithm fails to completely recover all the nonzero spectral values. A more detailed pseudo-code of the proposed iterative algorithm is given in Algorithm 2.

Algorithm 2 SparseFHT(x, N, K, C, L, $\mathbf{\Sigma}$)

Require: Input signal x of length $N = 2^n$. Sparsity K. Hash count C. Number of iterations of decoder L. Array $\mathbf{\Sigma}$ of C matrices in $\mathrm{GL}(n, \mathbb{F}_2)$, $\Sigma_c = [\sigma_{c,1} \mid \cdots \mid \sigma_{c,n}]$, $\sigma_{c,i} \in \mathbb{F}_2^n$.
Ensure: X contains the sparse Hadamard spectrum of x.

```
B = O(K)
D = n − b + 1
for c = 1, ..., C do
  U_{c,0} = FastHadamardHashing(x, N, Σ_c, 0, B)
  for d = 1, ..., D do
    U_{c,d} = FastHadamardHashing(x, N, Σ_c, σ_{c,d}, B)
  end for
end for
for l = 1, ..., L do
  for c = 1, ..., C do
    for k = 0, ..., B − 1 do
      if U_{c,0,k} = 0 then
        continue to next k
      end if
      v̂ ← 0
      for d = 1, ..., D do
        if U_{c,d,k} / U_{c,0,k} = −1 then
          v̂_{d−1} ← 1
        else if U_{c,d,k} / U_{c,0,k} ≠ 1 then
          continue to next k
        end if
      end for
      i ← Σ_c^{−T} (Ψ_b k + v̂)
      X_i ← U_{c,0,k}
      for c' = 1, ..., C do
        j ← Ψ_b^T Σ_{c'}^T i
        U_{c',0,j} ← U_{c',0,j} − X_i
        for d' = 1, ..., D do
          U_{c',d',j} ← U_{c',d',j} − X_i (−1)^{⟨σ_{c',d'}, i⟩}
        end for
      end for
    end for
  end for
end for
```

B. Complexity Analysis

Fig. 3 shows a full block diagram of the SparseFHT algorithm. Using this block diagram, we prove part 1 and 2 of Theorem 1 regarding 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.

1) Computational Complexity: As we further explain in Sections VI and VII, depending on the sparsity index of the signal α, we use C different hash blocks, where $C \leq (\frac{1}{\alpha} \vee \frac{1}{1-\alpha}) + 1$, and where each hash has $B = 2^b$ different output bins. We always select B to be equal to the number of nonzero spectral values K. This keeps the average number of nonzero components per bin $\beta = \frac{K}{B}$ equal to 1. As computing the hash outputs via an FHT block of size B needs $O(B \log_2(B))$ operations, selecting $K = B$, the resulting computational complexity is $O(K \log_2(K))$. Moreover, in our algorithm, we need to compute any hash output with $n - b = \log_2(\frac{N}{B})$ different shifts in order to make a collision detection/support estimation. Thus, the computational cost for each hash is $O(K \log_2(K) \log_2(\frac{N}{K}))$. As we need to compute C different hash blocks, the total computational complexity of computing hash outputs for each iteration will be $O(CK \log_2(K) \log_2(\frac{N}{K}))$.

After computing the hash outputs (output of hash bins), we do a ratio test to find hashes with only one nonzero component. In total, we have $CK \log_2(\frac{N}{K})$ output hash bins, thus this step needs $O(CK \log_2(\frac{N}{K}))$ operations. If the ratio test is successful for a specific bin indexed with k, we compute the binary vector $\hat{v}$ for this bin according to Eq. (8). The next step is to find the location of the nonzero component j using Eq. (9), i.e., $j = \Sigma^{-T}(\Psi_b k + \hat{v})$. This can be split in two parts: finding $\Sigma^{-T}\Psi_b k$ for the bin index k, and computing $\Sigma^{-T}\hat{v}$ for the binary index $\hat{v}$. The former can be calculated offline for every hash bin k, thus we focus on the latter. From Eq. (8), it is seen that the last b entries of $\hat{v}$ are zero, thus we have $\Sigma^{-T}\hat{v} = \sum_{i \in [n-b]} \hat{v}_i \bar{\sigma}_i$, where $\bar{\sigma}_i$, $i \in [n]$ are the columns of Σ^{-T}. This sum can be computed using at most $(n - b)$

multiplications and bitwise XOR operations. During the whole runtime of the algorithm, this calculation is done only K times corresponding to K nonzero variables to be peeled off, thus the resulting complexity is $O(K\log_2(\frac{N}{K}))$ (recall that $n-b=O(\log_2(\frac{N}{K}))$. Hence, the total computational complexity of every iteration is of the order $O(CK\log_2(K)\log_2(\frac{N}{K}))$. We will explain in Sections VI and VII how 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 is $O(CK\log_2(K)\log_2(\frac{N}{K}))$.

2) 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 is $CK\log_2(\frac{N}{K})$.

VI. Performance Analysis of the Very Sparse Regime

In Section V-A, we explained how by applying Proposition 1, we can hash the spectral-domain components in a collection of bins and how this can be represented by a bipartite graph. In this section, we consider the very sparse regime, where $\alpha\in(0,\frac{1}{3}]$. We show that by assuming a random support model for nonzero spectral components and for a careful design of hash functions, we obtain a random bipartite graph that behaves similarly to the ensemble of LDPC bipartite graphs. We also show that running the peeling decoder to recover the nonzero spectral components is equivalent to the belief propagation (BP) decoding for a binary erasure channel (BEC). In particular, we prove that that the error (decoding failure) probability can be asymptotically characterized by a *'density evolution'* (DE) equation, thus enabling for a perfect analysis of the peeling decoder. We use the following steps to rigorously analyze the performance of the decoder in the very sparse regime:

1) We explain the construction of suitable hash functions depending on the value of $\alpha\in(0,\frac{1}{3}]$.
2) We rigorously analyze the structure of the induced bipartite graph obtained by treating the nonzero 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-degree 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 [19], to prove the concentration of the number of unpeeled edges around its expected value, which we also characterize. Wormald's method, as exploited in [20], 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 equation.
4) Wormald's method gives a concentration bound on 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 [20], is necessary to show that if the peeling decoder peels $1-\gamma$ fraction of the edges successfully, it can continue to peel off all the remaining edges with a very high probability.

A. Hash Construction

For the very sparse regime, $\alpha\in(0,\frac{1}{3}]$, consider those values of α that are equal to $\frac{1}{C}$ for some positive integer $C\geq 3$. For $\alpha=\frac{1}{C}$, we build 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 $b=\frac{n}{C}$. We label the components of the vector X by n dimensional binary vector from $\mathbb{F}_2^n$. We design C different subsampling operators, where the i-th one keeps all the indices $i\,b$ up to $(i+1)b-1$ and sets the other indices equal to zero. More precisely, using the terminology of Proposition 1, the i-th hashing operator is given by

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

where I_b is the identity matrix of order b. Let $x_0^{n-1}\in\mathbb{F}_2^n$ be the binary labeling of the elements of the signal x. Equivalent to the C different subsampling operators, we can consider functions $h_i, i\in[C]$, where

$$h_i(x_0^{n-1})=(x_{i\,b},x_{i\,b+1},\ldots,x_{i\,b+b-1}).$$

Ignoring the multiplicative constants, we can see from Eq. (5) that the spectral component labelled with X_0^{n-1} is hashed to the bin labelled with $h_i(X_0^{n-1})\in\mathbb{F}_2^b$ in the i-th hash.

As we explain in Section VI-C (Proposition 9), we need at least $C=3$ hashes for the peeling algorithm to work successfully and that is the main reason 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, we can construct $\lfloor\frac{1}{\alpha}\rfloor$ hashes with $B=2^{\lfloor n\alpha\rfloor}$ output bins and one hash with a smaller number of output bins. Thus, the required number of hashes is at most $1+\lfloor\frac{1}{\alpha}\rfloor$.

B. Random Bipartite Graph Construction

1) Random Support Models: In the very sparse regime, the number of nonzero spectral components is $K=O(N^\alpha)$ for some $\alpha\in(0,\frac{1}{3}]$. For a given (K,N), we define RS1(K,N) as the class of Hadamard-domain signals whose support is selected uniformly at random from the set of all $\binom{N}{K}$ possible supports of size K. Model RS1 is equivalent to selecting K out of N objects (locations of nonzero values) at random without replacement. Assuming that the indices for the support are selected independently but with replacement, we obtain another model that we call RS2(K,N). The size of a random support in RS2(K,N) is itself a random variable less than or equal to K. The following proposition, proved in Appendix C, shows that in the sub-linear 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 $\mathcal{G}(K, B, C)$: Let us consider C disjoint sets of check nodes $S_1, S_2, \ldots, S_C$ of the same size $|S_i| = B$. A graph G in the ensemble $\mathcal{G}(K, B, C)$ is a bipartite graph with K variable nodes on the left and $C \times B$ check nodes $\cup_{i=1}^{C} S_i$ on the right. Each variable node v in G, independently from other variable nodes, is connected to check nodes $\{s_1, s_2, \ldots, s_C\}$, where every $s_i \in S_i$ is selected uniformly at random from S_i, independent of the the other s_j. Hence, every edge e in G can be labelled as (v, c), where $v \in [K]$ is a variable node and c is a check node in one of $S_1, S_2, \ldots, S_C$. If two different variable nodes are connected to exactly the same check nodes, we consider them equivalent and we keep only one of them. By construction, all the resulting bipartite graphs in the ensemble are left regular with the variable degree C but the check node degree is not fixed.

3) Ensemble of Graphs Generated by Hashing: In Section VI-A, we explained the subsampling operator and the hash construction for the very-sparse regime. As we described in Section V-A, we can represent the hashing operation by a bipartite graph. In this section, our aim is to study the resulting bipartite graph for the proposed hash construction.

Recall that, the subsampling operator h_i is given by

$$h_i(x_0^{n-1}) = (x_{i\,b}, x_{i\,b+1}, \ldots, x_{i\,b+b-1}),$$

which maps the spectral component labeled with $X_0^{n-1} \in \mathbb{F}_2^n$ into the bin labelled with $h_i(X_0^{n-1}) \in \mathbb{F}_2^b$. Notice that by this hashing scheme there is a one-to-one relation between a spectral element labelled with X_0^{n-1} and its bin indices in different hashes $(h_0(X_0^{n-1}), h_1(X_0^{n-1}), \ldots, h_{C-1}(X_0^{n-1}))$.

Now, suppose $\{X_1, X_2, \ldots, X_{\mathrm{K}}\}$ is a subset of binary indices in $\mathbb{F}_2^n$ that is selected uniformly at random from all the subsets of $\mathbb{F}_2^n$ of size K, and denotes the position of nonzero spectral components. For these K variables and hash functions h_i, we can associate a bipartite graph as follows. We consider K variable nodes corresponding to X_i, $i \in [K]$, and C different set of check nodes $S_0, S_1, \ldots, S_{C-1}$ each of size $B = 2^b$. The check nodes in each S_i are labelled by elements of $\mathbb{F}_2^b$. For each variable X_i we consider C different edges connecting X_i to check nodes labelled with $h_j(X_i) \in S_j$, $j \in [C]$.

As X_i are selected at random *without* replacement (according to RS1), they are not independent and the resulting bipartite graph is not compatible with Balls and Bins model explained in Section VI-B2. This makes the analysis difficult. We solve this problem in two steps. First, in Proposition 4, we prove that we can still obtain a graph compatible with Balls and Bins model $\mathcal{G}(K, B, C)$ if we use RS2 instead of RS1. This is equivalent to sampling the indices randomly and independently but with replacement. Second, in Proposition 5, we prove that for a large dimension N, the failure probability of our proposed algorithm over RS1 model is upper bounded by the failure probability over $\mathcal{G}(K(1+\epsilon), B, C)$, i.e., the Balls and Bins model with a slightly higher number of variables.

Proposition 4: Let $h_i : \mathbb{F}_2^n \to \mathbb{F}_2^b$, $i \in [C]$ be as explained before. Let $\{V_j : j \in [K]\}$ be a random set from the ensemble $\mathrm{RS2}(K, N)$ for $N = 2^n$. The bipartite graph associated with variables V_j and hash functions h_i is a graph from ensemble $\mathcal{G}(K, B, C)$, where $B = 2^b$.

Proof: Recall that in the bipartite graph representation, we assign a variable node to each V_j and consider a set of check node $\cup_{i\in[C]} S_i$, where S_i is the set of all check nodes corresponding to all hash outputs in hash i. As $\{V_j : j \in [K]\}$ belongs to the ensemble $\mathrm{RS2}(N, K)$, all the variables V_j are independent from each other. Hence, for a fixed hash function h_i, the variables $h_i(V_j)$, $j \in [K]$, are also independent of each other. This implies that in the resulting bipartite graph, different variable nodes select their corresponding check node in S_i independent of each other.

Now consider a specific variable node V_k. This variable node is connected to hash bin $h_i(V_k)$ in the i-th hash. As V_k is a uniformly distributed random variable over $\mathbb{F}_2^n$, we can represent it by a binary vector B_0^{n-1} whose components B_i are like i.i.d. unbiased bits. For the constructed hash functions, we can see that the corresponding hash indices

$$h_0(B_0^{n-1}), h_1(B_0^{n-1}), \ldots, h_{C-1}(B_0^{n-1})$$

are independent from one another because they depend on disjoint subsets of B_0^{n-1}. Moreover, each $h_i(B_0^{n-1})$ is uniformly distributed over $\mathbb{F}_2^b$. This implies that every variable node V_k selects its neighbor check (hash bin) in each S_i, $i \in [C]$ uniformly and independently of the other $S_{i'}, i' \neq i$. Thus, the resulting graph belongs to $\mathcal{G}(K, B, C)$. ∎

In Section V, we explained the peeling decoder for recovering the nonzero spectral components. It is not difficult 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. This helps the peeling decoder to succeed decoding. With this explanation, we can prove the following proposition.

Proposition 5: Let α, C, K, $h_i, i \in [C]$ be as explained before. Let $\mathcal{G}_1$ be the ensemble of bipartite graphs induced by the random support model $\mathrm{RS1}(K, N)$ and hash functions h_i. For any $\epsilon > 0$ and for large dimension N, the average failure probability of the peeling decoder over $\mathcal{G}_1$ is upper bounded by its average failure probability over the ensemble $\mathcal{G}(K(1+\epsilon), B, C)$.

Proof: Let G_ϵ be a graph from ensemble $\mathcal{G}(K(1+\epsilon), B, C)$. From Proposition 3, for large dimension N, the number of variable nodes in G_ϵ is greater than K with a very high probability. If we drop some of the variable nodes at random from G_ϵ, to keep only K of them, we obtain a graph G_1 from ensemble $\mathcal{G}_1$. As the performance of the peeling decoder improves by removing some of the variable nodes, it performs strictly better over G_1 compared with G_ϵ. ∎

Proposition 4 and Proposition 5 enable us to restrict the analysis of the performance of the peeling decoder to graphs from ensemble $\mathcal{G}(K, B, C)$.

4) Edge Degree Distribution Polynomial: In this section, we restrict ourselves to the graphs from the ensemble $\mathcal{G}(K, B, C)$ for the very sparse regime $\alpha \in (0, \frac{1}{3}]$. We assume that $n\alpha \in \mathbb{N}$

and select $b = n\alpha$. Hence, we have $K = B$. We call $\beta = \frac{K}{B}$ the average number of nonzero components per hash bin. We design the hash functions so that $\beta = 1$. All the graphs from the ensemble $\mathcal{G}(K, B, C)$ are left regular, i.e., all the variable nodes have degree C, whereas the degree of a check node 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, as N tends to infinity, the check degree converges to a Poisson random variable with parameter β.

Proof: The 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 enables us to write analysis in a compact form.

Proposition 7: Let 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 the edges are connected to variable nodes of degree C, thus $\lambda(\alpha) = \alpha^{C-1}$. To find $\rho(\alpha)$, we need to find the fraction of edges that are connected to check nodes of a specific degree. From the symmetry of hash construction, it is sufficient to find 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 $i \geq 1$ and let N_i be the number of check nodes in the first hash with degree i. By definition of ρ_i, we obtain that

$$\rho_i = \frac{i N_i}{K} = \frac{i N_i / B}{K / B}.$$

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, we can show that

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

Thus, ρ_i converges almost surely to $\frac{e^{-\beta}\beta^{i-1}}{(i-1)!}$. As $\rho_i \leq 1$, for any α with $|\alpha| < 1 - \epsilon$, we have $|\rho_i \alpha^{i-1}| \leq (1 - \epsilon)^{i-1}$. Applying the *dominated convergence* theorem, we can prove that $\rho(\alpha)$ converges to $\sum_{i=1}^{\infty} \frac{e^{-\beta}\beta^{i-1}}{(i-1)!}\alpha^{i-1} = e^{-\beta(1-\alpha)}$. ∎

As we will explain in Section VI-C, the performance of the peeling decoder highly depends on the parameter β; the lower β the better the performance of the peeling decoder. The drawback is that decreasing β, via increasing B, increases the time complexity $O(B \log_2(B))$ of computing the hash functions. Generally, we can select B 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 for the peeling decoding.

C. Performance Analysis of the Peeling Decoder

Consider a random bipartite graph resulting from applying C hashes to the signal spectrum. As we explained in Section V, the iterative peeling algorithm starts by finding a singleton, i.e., a check node of degree 1 that is connected to only one variable node. The decoder peels off this variable node and removes all the edges connected to it from the graph. The algorithm continues by peeling off a singleton at each step until all the check nodes are zerotons; all the nonzero variable nodes are decoded, or all the remaining unpeeled check nodes are multitons, in which case the algorithm fails to completely decode all the nonzero spectral variables.

1) Wormald's Method: In order to analyze the behavior of the resulting random graphs under the peeling decoding, the authors in [20] 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 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 [20], this method is applied to analyze the performance of the peeling decoder for bipartite graph codes over the BEC. We briefly explain the problem setting in [20] and how it can be used in our case.

Assume that we have a bipartite graph G with K variable nodes at the left, $C\,K$ check nodes at the right and with edge degree polynomials $\lambda(x)$ and $\rho(x)$. We can define a channel code $\mathcal{C}(G)$ 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 all the message bits that are connected to it. Thus, the resulting code will be a systematic code with K message bits, along with $C\,K$ parity check bits. To communicate a K bit message over the channel, we send K message bits and all the check bits associated with them. While passing through the BEC, some of the message bits or check bits are erased independently. Consider 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 G, we end up with another bipartite graph G'. It is easy to see that over the

new graph G', we can apply the peeling decoder to recover the erased bits.

In [20], this problem was fully analyzed for the case of $\delta' = 0$, where all 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 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 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, we use [20, Proposition 2], which states the following.

[20, Proposition 2]: Let 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

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

For any $\eta > 0$, there is some K_0 such that for all $K > K_0$, if the message bits of $\mathcal{C}(G)$ 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 in our algorithm.

Proposition 8: Let $K = O(N^\alpha)$ for some $\alpha \in (0, \frac{1}{3}]$ and let G be a bipartite graph from the ensemble $\mathcal{G}(K, B, C)$ induced by hashing functions $h_i, i \in [C]$, with $\beta = \frac{K}{B}$, and with edge degree polynomials $\lambda(x) = x^{C-1}$ and $\rho(x) = e^{-\beta(1-x)}$. Suppose that

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

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 nonzero spectral components.

Proposition 8 does not guarantee the success of the peeling decoder. It only implies that with a very high probability, it can peel off any fraction $\eta \in (0, 1)$ of nonzero components, but not necessarily all of them. Using a combinatorial argument, however, it is possible to prove that with very high probability any graph in the ensemble $\mathcal{G}$ is an expander graph, specifically, 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 in [20, Lemma 1] gives the following result proved in Appendix D.

Proposition 9: Let $K = O(N^\alpha)$ for some $\alpha \in (0, \frac{1}{3}]$ and let 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 $1 - O(\frac{1}{N^{3\alpha(C/2-1)}})$, 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 $C = \lfloor \frac{1}{\alpha} \rfloor \geq 3$ hashes each containing $2^{n\alpha}$ output bins. Combining Proposition 8 and Proposition 9, we obtain that the success probability of the peeling decoder is lower bounded by $1 - O(\frac{1}{N^{3\alpha(C/2-1)}})$ as mentioned in Remark 1.

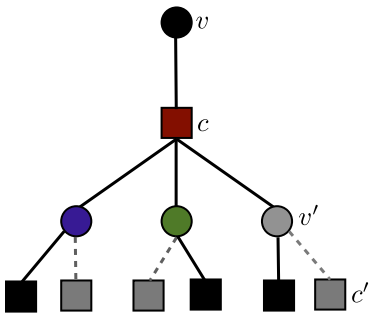

Fig. 4. Tree-like neighborhood of an edge $e = (v, c)$. Dashed lines show the edges that have been removed before iteration t. The edge e is peeled off at iteration t because all the variable nodes v' connected to c are already decoded, thus c is a singleton check.

2) Analysis Based on Belief Propagation Over Sparse Graphs: In this section, we give another method of analysis and provide 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 belief-propagation (BP) decoder over sparse locally tree-like graphs. The analysis is very similar to the analysis of the peeling decoder for recovering nonzero frequency components in [15]. We first need some terminology from graph theory. A walk of size ℓ in graph G starting from a node $v \in [K]$ is a set of ℓ edges $e_1, e_2, \ldots, e_\ell$, where v is one of the vertices of the edge e_1 and where consecutive edges are different, $e_i \neq e_{i+1}$, but incident with each other. A directed neighborhood of an edge $e = (v, c)$ of depth ℓ is the induced subgraph in 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 $e_1 = (v, c)$. A node e is said to have a tree-like neighborhood of depth ℓ if the directed neighborhood of e of depth ℓ is a tree.

Let $e = (v, c)$ be an edge in a graph from ensemble $\mathcal{G}(K, B, C)$ and consider a directed neighborhood of this edge of depth ℓ. At the first stage, it is easy to see that this edge is peeled off from the graph assuming that all the edges (c, v') connected to the check node c are peeled off, because in that case check node c will be a singleton enabling us to decode the variable v. This is shown in Fig. 4.

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 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

$$p_{j+1} = \lambda(1 - \rho(1 - p_j)), \quad j \in [\ell], \tag{11}$$

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 is proved that for any specific edge e, asymptotically with

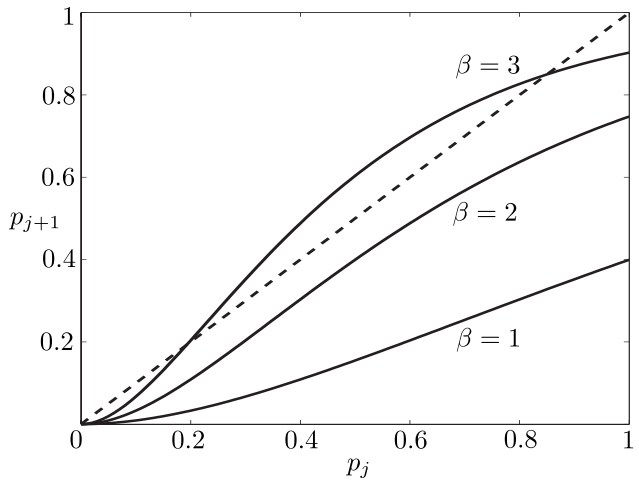

Fig. 5. Density Evolution equation for $C = 3$ and different values of $\beta = \frac{K}{B}$.

a 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 is applied in [15] to show that not only the average number of unpeeled edges is approximately KCp_ℓ, but also with a very high probability the number of those edges is well concentrated around KCp_ℓ.

Eq. (11) is generally known as the density evolution equation. Starting from $p_0 = 1$, this equation fully predicts the behavior of the peeling decoding over the ensemble $\mathcal{G}$. Fig. 5 shows a typical behavior of this iterative equation for different values of the parameter $\beta = \frac{K}{B}$.

For very small values of β, this equation has only a fixed point at 0, which implies that for large dimension N, the peeling decoder can recover a fraction 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 cannot 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 $x = \lambda^{-1}(p)$, we obtain that $x > 1 - \rho(1 - \lambda(x))$ or equivalently

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

This is exactly the same result that we obtained by applying Wormald's method, as in [20]. In particular, this analysis clarifies the role of x in Wormald's method.

Similar to Wormald's method, this analysis only guarantees that for any $\epsilon \in (0, 1)$, asymptotically as N tends to infinity, $1 - \epsilon$ fraction 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 ($\alpha \in (\frac{1}{3}, 1]$), 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 nonzero 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 the variables are peeled off, thus all the remaining checks are zerotons otherwise some of the nonzero spectral values are not recovered and the perfect recovery fails.

As we explain in Section VII-B, 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 nonzero spectral values by an equivalent bipartite graph.
3) Analyzing the performance of the peeling decoder over the resulting bipartite graph.

For simplicity, we consider the case where $\alpha = 1 - \frac{1}{C}$ for some integer $C \geq 3$. 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 $r_i = X_{it}^{(i+1)t-1}$. 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

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

and the hash output is obtained by appending $C - 1$ first r_i, $i \in [C]$. We 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 Section 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 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 nonzero elements per bin in every hash is kept at $\beta = \frac{K}{B} = 1$ and the complexity of computing all the hashes

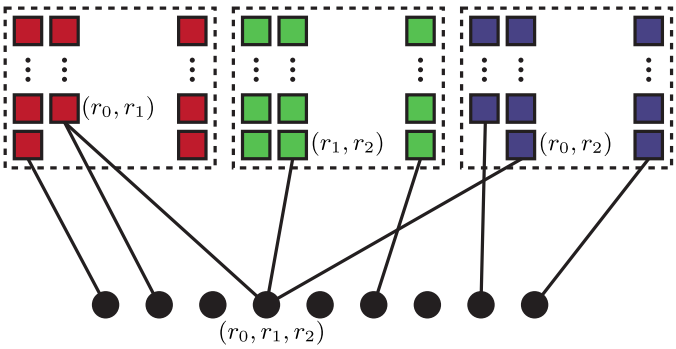

Fig. 6. Bipartite graph representation for the less sparse case $\alpha = \frac{2}{3}$, $C = 3$.

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

Similarly to the very sparse regime, we can assign a bipartite graph with the K left nodes (variable nodes) corresponding to nonzero 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 nonzero 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 independently from one another from $[N]$, then the resulting graph is a random left regular bipartite graph, where each variable node selects completely independently its C neighbors in $S_1, S_2, \ldots, S_C$. However, in the less sparse regime, the selection of the neighbor checks in different hashes is not completely random. To explain more, suppose $\alpha = \frac{2}{3}$, thus $C = 3$. Also assume that for a nonzero spectral variable labelled with X_0^{n-1}, r_i denotes $X_{it}^{(i+1)t-1}$, where $t = \frac{n}{C}$. In this case, this variable is connected to bins labelled with (r_0, r_1), (r_1, r_2) and (r_0, r_2) in 3 different hashes. This has been depicted in Fig. 6.

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 nonzero spectral variables, the bin association for different variables V_i is still made 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 [20]. However, an analysis based on the DE for the BP algorithm can still be applied. In other words, setting $p_0 = 1$ and

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

as in (11) 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 cannot be decoded because all its neighbor check nodes are multitons. We first analyze the structure of the trapping set and find the probability that a specific set of variables builds 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 $C = 3$. 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

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

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

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

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

Proposition 10: Assume that T is a trapping set for the C dimensional lattice representation L of the nonzero spectral-domain variables as explained before. Then for any r_i on the 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 by 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 3: The bound $|T| \geq 2^C$ on the size of the trapping set is actually tight. For example, for $i \in [C]$ consider r_i, r_i' where $r_i \neq r_i'$ and let

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

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 that implies that if the peeling decoder can decode all the variable nodes except a fixed number of them, with a 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. If the peeling decoder decodes all except a set of variables of size s, it can decode all the variables with very high probability.

Proof: The proof is 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 $p_{\max}$ values of r_0 along the R_0 axis. Proposition 10 implies that each $T(R_0 = r_0)$ is a trapping set that 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 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 nonzero variables X_0^{n-1} with the equivalent representation $(r_0, r_1, \ldots, r_{C-1})$, the coordinates of any variable are selected uniformly and completely independently from one another and from the coordinates of the other variables. This implies that

$$\mathbb{P}\{F_s\} \leq \mathbb{P}\{\text{For any variables in } T,\ r_i \in T_i, i \in [C]\} \leq \prod_{i=0}^{C-1} \binom{\mathcal{P}i}{p_i} \left(\frac{p_i}{\mathcal{P}_i}\right)^s \leq \prod_{i=0}^{C-1} \binom{\mathcal{P}i}{s/2^{C-1}} \left(\frac{s}{2^{C-1}\mathcal{P}_i}\right)^s,$$

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

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

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

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

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

D. Generalized Hash Construction

The hash construction for the less sparse regime that we explained in Section VII-A only covers values of $\alpha = 1 - \frac{1}{C}$ for $C \geq 3$, which belongs to the region $\alpha \in [\frac{2}{3}, 1)$. In this section, we explain a hash construction that fills the gap for $\alpha \in (\frac{1}{3}, \frac{2}{3})$, and extends to any value of $\alpha \in (0, 1)$ that is not necessarily of the form $\frac{1}{C}$ (very sparse) or $1 - \frac{1}{C}$ (less sparse).

In the very sparse regime $\alpha = \frac{1}{3}$, we have $C = 3$ different hashes and for a nonzero spectral variable X with the binary index $X_0^{n-1} = (r_0, r_1, r_2)$, the i-th hash output is $h_i(X_0^{n-1}) = r_i$, $i \in \{0, 1, 2\}$, 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 $\alpha = \frac{2}{3}$. Generally, let C be the desired number of hashes. We give the following construction for the hash functions

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

where $b = n\alpha$ and $t = \frac{n}{C}$, and where the values of the indices are computed modulo n, for example $X_n = X_0$. Furthermore, the required number of hashes is given by $C = (\frac{1}{\alpha} \vee \frac{1}{1-\alpha})$.

It is clear that each hash is a surjective map from $\mathbb{F}_2^n$ into $\mathbb{F}_2^b$. Moreover, for this choice of b ($b = n\alpha$), the number of output bins in each hash is $B = 2^{n\alpha} = N^\alpha = K$, thus the average number of nonzero variables per bin in every hash is equal to $\beta = \frac{K}{B} = 1$. Also, for the intermediate values of $\alpha \in (\frac{1}{3}, \frac{2}{3})$, we expect the performance of the peeling decoder for this regime to be between the very sparse regime $\alpha = \frac{1}{3}$ and the less sparse one $\alpha = \frac{2}{3}$.

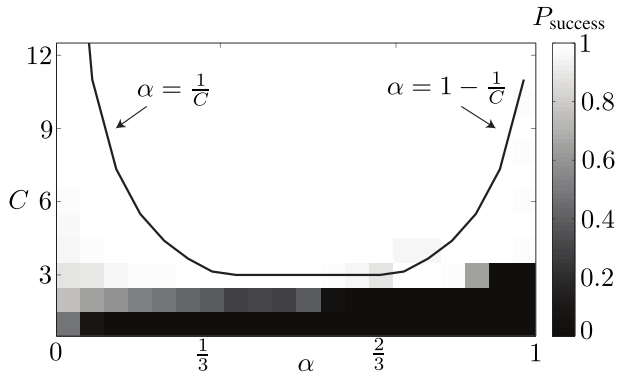

Fig. 7. Probability of success of the algorithm as a function of α and C for deterministic hash construction. The dimension of the signal is $N = 2^{22}$. The black line corresponds to $\alpha = \frac{1}{C}$ and $\alpha = 1 - \frac{1}{C}$ in the very sparse and less sparse regimes respectively. We fix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

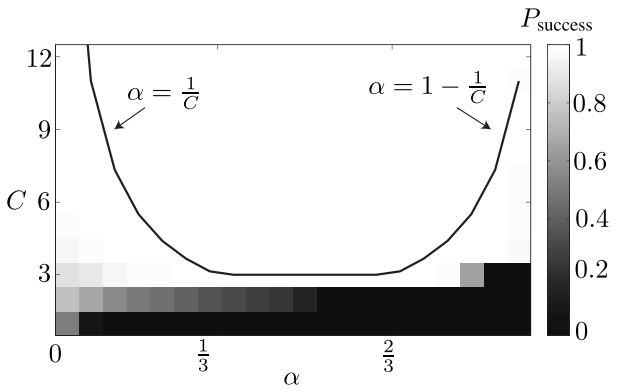

Fig. 8. Probability of success of the algorithm as a function of α and C for random hash construction. The dimension of the signal is $N = 2^{22}$. The black line corresponds to $\alpha = \frac{1}{C}$ and $\alpha = 1 - \frac{1}{C}$ in the very sparse and less sparse regimes respectively. We fix $\beta = 1$. The hashing matrices are picked uniformly 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 the C programming language and the success probability of the algorithm is estimated via a sufficient number of trials. We also provide a comparison of the run time of our algorithm and the standard Hadamard transform. In all experiments, the input signal has support uniformly drawn at random without replacement. The nonzero components are drawn from a zero-mean normal distribution with variance $\sigma^2 = 100$. In the spirit of reproducible research, all the material (C and Matlab code) needed to reproduce the results of this paper is available online at http://lcav.github.io/SparseFHT/.

- *Experiment 1:* We fix the signal size to $N = 2^{22}$ and run the algorithm 1000 times to estimate the success probability for $\alpha \in (0, 1)$ 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, $i \in [C]$, uniformly at random from $\mathrm{GL}(n, \mathbb{F}_2)$. The result is shown in Fig. 8. We observe 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 nonzero coefficients per bin. In our design for hash function, we always use $\beta \approx 1$ throughout the paper. 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 shows the simulation results. For $\beta \leq 2$, the algorithm succeeds with probability very close to one. Moreover, for values of β larger than 3, the success probability sharply goes to 0.

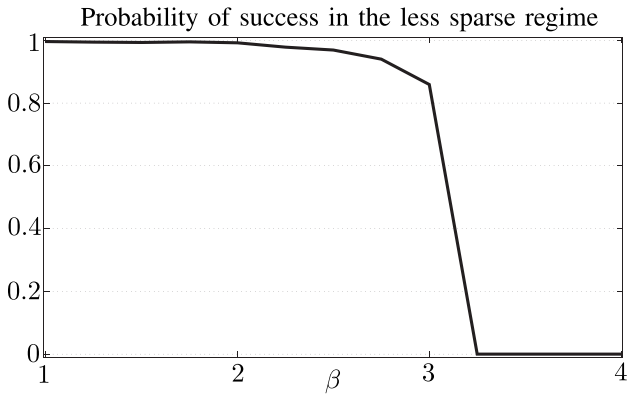

Fig. 9. Probability of success of the algorithm in the less sparse regime as a function of $\beta = K/B$. We fix $N = 2^{22}$, $B = 2^{17}$, $C = 4$, and vary α in the range 0.7 to 0.9.

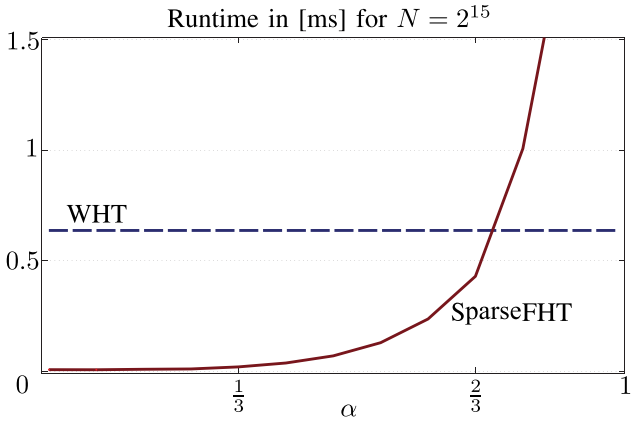

Fig. 10. Comparison of the median runtime in ms of the SparseFHT and conventional WHT for $N = 2^{15}$ and for different values of α.

- *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 interesting 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

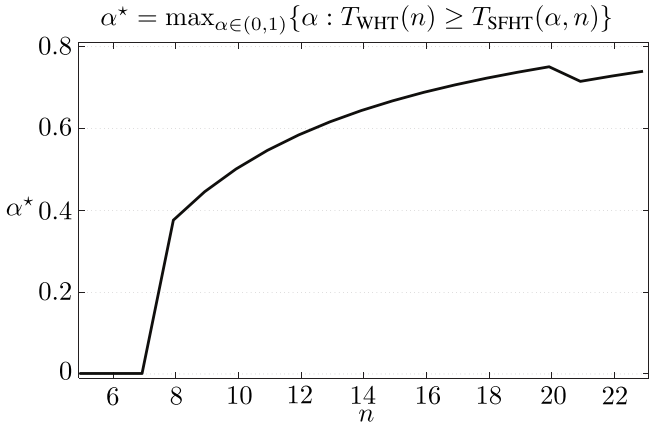

Fig. 11. In this figure, we change the value of $n = \log_2 N$ and plot α^*, the largest value of α such that SparseFHT runs faster than the conventional WHT. When WHT is always faster, we simply set $\alpha^* = 0$.

SparseFHT is faster than FHT for any lower value of α. That is

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

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.

IX. Conclusion

We presented a new algorithm for computing the Hadamard transform of a signal of length N that is K-sparse in the Hadamard domain with $K = O(N^\alpha)$ scaling sub-linearly with N for some $\alpha \in (0, 1)$. Our algorithm computes the K-sparse Hadamard transform of the signal with a computational complexity $O(K \log_2 K \log_2 \frac{N}{K})$, and only requires $O(K \log_2 \frac{N}{K})$ time-domain samples of the signal. We have shown that the algorithm correctly reconstructs the Hadamard transform of the signal with a very high probability approaching 1 for a sufficiently large dimension N.

We evaluated empirically the performance of our algorithm through numerical simulations, and compare its speed with that of the conventional fast Hadamard transform. We observe that our algorithm is much faster, even for moderate signal lengths (e.g. $N = 2^{10}$) and reasonable sparsity.

In our algorithm, we considered a noiseless case, where there is no measurement noise in the time-domain samples. This assumption was necessary in Proposition 2 in order to make a collision detection/support estimation. Hence, a more robust variant of Proposition 2 is necessary for the noisy case. We were informed of a recent publication addressing this issue during the review process [21].

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. As $(H_N)_{i,j} = (-1)^{\langle i\,,\,j\rangle}$, the identity (2) implies that

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

Therefore, for any $i, j \in \mathbb{F}_2^n$, π_1, π_2 must satisfy $\langle \pi_2(i)\,,\ j\rangle = \left\langle i,\ p i_1^{-1}(j)\right\rangle$. By linearity of the inner product, we obtain

$$\begin{aligned} \langle \pi_2(i+k)\,,\ j\rangle &= \left\langle i+k,\ \pi_1^{-1}(j)\right\rangle \\ &= \left\langle i,\ \pi_1^{-1}(j)\right\rangle + \left\langle k\,\pi_1^{-1}(j)\right\rangle \\ &= \langle \pi_2(i)\,,\ j\rangle + \langle \pi_2(k)\,,\ j\rangle\,. \end{aligned}$$

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 (2) are in one-to-one correspondence with the elements of $\mathrm{GL}(n, \mathbb{F}_2)$. ■

C. Proof of Property 3

As Σ is non-singular, Σ^{-1} exists, and from the definition of the WHT, it follows that

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

This completes the proof. ■

D. Proof of Property 4

Note that $m \in \mathbb{F}_2^b$, and $x_{\Psi_b m}$ is a signal of dimension $B = 2^b$. Let $\tilde{X}_k$ denote its WHT, where $k \in \mathbb{F}_2^b$. From the definition of WHT, we have

$$\begin{aligned} \tilde{X}_k &= \frac{1}{\sqrt{B}} \sum_{m\in\mathbb{F}_2^b} (-1)^{\langle k\,,\,m\rangle} x_{\Psi_b m} \\ &\overset{(a)}{=} \frac{1}{\sqrt{BN}} \sum_{m\in\mathbb{F}_2^b} (-1)^{\langle k\,,\,m\rangle} \sum_{u\in\mathbb{F}_2^n} (-1)^{\langle \Psi_b m\,,\,u\rangle} X_u \\ &\overset{(b)}{=} \frac{1}{\sqrt{BN}} \sum_{u\in\mathbb{F}_2^n} X_u \sum_{m\in\mathbb{F}_2^b} (-1)^{\langle m\,,\,k+\Psi_b^T u\rangle} \\ &\overset{(c)}{=} \frac{B}{\sqrt{BN}} \sum_{u\in\mathbb{F}_2^n} X_u \mathbb{1}_{\{k+\Psi_b^T u=0\}}, \end{aligned}$$

where in (a), we used the inverse of the WHT for the N dimensional signal x $(N = 2^n)$ and its transform-domain

signal X, in (b), we used $\langle \Psi_b m\,,\, u\rangle = \langle m\,,\, \Psi_b^T u\rangle$, and in (c), we used the following identity for $s \in \mathbb{F}_2^b$,

$$\sum_{m\in\mathbb{F}_2^b} (-1)^{\langle m\,,\, s\rangle} = B\mathbb{1}_{\{s=0\}}.$$

We can check that $k + \Psi_b^T u = 0$ holds if and only if $u = \Psi_b k + j$ with $j \in \mathcal{N}(\Psi_b^T)$. Hence, we obtain the desired result $\tilde{X}_k = \sqrt{\frac{B}{N}} \sum_{j\in\mathcal{N}(\Psi_b^T)} X_{\Psi_b k+j}$. ∎

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 $\mathcal{I}_k = \{j\,|\mathcal{H}j = k\}$ be the set of variable indices that are hashed to bin k. This set is finite and its elements can be enumerated as $\mathcal{I}_k = \{j_1, \ldots, j_{\frac{N}{B}}\}$. In particular, $\mathcal{I}_k$ is an $n-b$ dimensional affine subspace of $\mathbb{F}_2^n$. We show that a set $\{X_j\}_{j\in\mathcal{I}_k}$ is very unlikely, unless it contains only one nonzero 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

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

where $\sigma_i, i \in \{1, \ldots, n\}$ 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 $\sigma_1, \ldots, \sigma_{n-b}$ are linearly independent, 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-hand side vector is always one of the columns of the matrix and there is a unique solution with only one nonzero 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 solution yields another solution. However, we show that this matrix is full rank (its null space has dimension $2^{n-b} - n + b - 1$), and assuming a continuous distribution for the nonzero components X_j, the probability that $\{X_j\}_{j\in\mathcal{I}_k}$ falls in this null space is zero.

To prove that the matrix is indeed full rank, let us first focus on the sub-matrix obtained by removing the first row. Let us call this matrix A. Also, let $M = -2I + \mathbb{1}\mathbb{1}^T$, where I is the identity matrix of order $n-b$ and $\mathbb{1}$ is the all-one vector of dimension $(n-b)$. We can simply check that all the components of M are ± 1. Hence, the columns of M are contained among the columns of the submatrix A. It is not difficult to 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 $v_0 = \frac{\mathbb{1}}{\sqrt{n-b}}$ of dimension $n-b$ is an eigen-vector of M with eigen-value $\lambda_0 = n-b-2$. Moreover, as the eigen-vectors are orthogonal to each other, we obtain 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 the eigen-values are nonzero and M is invertible, which implies that the sub-matrix A 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 the 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 (9), let Σ_L and Σ_R be the matrices containing the first $n-b$ and the last b columns of Σ respectively, such that $\Sigma = [\Sigma_L\ \Sigma_R]$. If there is only one coefficient in the bin, then (7) implies that $\hat{v} = [\,(j^T\Sigma_L)\ 0\,]^T$. Using definitions (3) and (6), 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 not difficult to see that $\{H_t\}_{t\in[K]}$ is a Markov process. Moreover,

$$\mathbb{E}\left[H_{t+1} - H_t | H_t\right] = (1 - H_t/N), \tag{12}$$

because the size of the random set increases by one if and only if we choose an element from $[N]$ that has not been selected until time t, and conditioned on H_t, this happens with probability $1 - \frac{H_t}{N}$. This implies that $a_{t+1} = 1 + \gamma\, a_t$, where $\gamma = 1 - \frac{1}{N}$. Solving this equation, with initialization $a_0 = 1$, we obtain that

$$a_t = \sum_{r=0}^{t} \gamma^{\,r} = \frac{1-\gamma^{\,t+1}}{1-\gamma} = N(1-\gamma^{\,t+1}). \tag{13}$$

In particular, $a_K = N(1-(1-\frac{1}{N})^K)$, which implies that

$$\begin{aligned} \mathbb{E}\left[\frac{H_K}{K}\right] &= \frac{N}{K}(1-(1-\frac{1}{N})^K) \\ &\geq \frac{N}{K}\Big(1-(1-\frac{K}{N}+\frac{K(K-1)}{2N^2})\Big) \\ &\geq 1 - O(\frac{K}{N}). \end{aligned}$$

We can see that for $K = N^\alpha$, $\alpha \in (0,1)$, 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

$$\mathrm{Var}(H_{t+1}) = \mathbb{E}\left[\mathrm{Var}(H_{t+1}|H_t)\right] + \mathrm{Var}(\mathbb{E}\left[H_{t+1}|H_t\right]). \tag{14}$$

Using Eq. (12), we obtain that

$$\mathrm{Var}(\mathbb{E}\left[H_{t+1}|H_t\right]) = \mathrm{Var}(1+\gamma\, H_t) = \gamma^{\,2} v_t, \tag{15}$$

where v_t denotes the variance of H_t. Moreover, for the first term in Eq. (14), we have

$$\begin{aligned}\mathbb{E}\left[\mathrm{Var}(H_{t+1}|H_t)\right] &= \mathbb{E}_{H_t}\{\mathrm{Var}(H_{t+1}|H_t = h_t)\} \\ &= \mathbb{E}_{H_t}\{\mathrm{Var}(H_{t+1} - H_t|H_t = h_t)\} \\ &\overset{(a)}{=} \mathbb{E}\left[\frac{H_t}{N}\left(1 - \frac{H_t}{N}\right)\right] \\ &= \frac{a_t}{N} + \frac{a_t^2 + v_t}{N^2}, \end{aligned} \tag{16}$$

where in (a), we used the fact that given H_t, $H_{t+1} - H_t$ is a Bernoulli random variable that is zero with probability $\frac{H_t}{N}$, thus its variance in equal to $\frac{H_t}{N}(1 - \frac{H_t}{N})$. Combining (15) and (16), we obtain

$$v_{t+1} = \left(\gamma^2 + \frac{1}{N^2}\right) v_t + \frac{a_t}{N}\left(1 + \frac{a_t}{N}\right). \tag{17}$$

From (13), it is easy to see that a_t is an increasing function of t. Moreover, from (17) it is seen that v_{t+1} is an increasing function of a_t, thus if we consider the following recursion

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

then for any $t \in [K]$, $v_t \leq w_t$. We can also check that w_t, starting with the initialization $w_0 = 0$, is also an increasing sequence of t, thus we have

$$\begin{aligned} v_K \leq w_K \leq w_\infty &= \frac{a_K}{N}\left(1 + \frac{a_K}{N}\right)\Big/\left(1 - \gamma^2 - \frac{1}{N^2}\right) \\ &= \frac{a_K}{2}\left(1 + \frac{a_K}{N}\right)\Big/\left(1 - \frac{1}{N}\right). \end{aligned}$$

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

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

Obviously, $\frac{H_K}{K} \leq 1$, 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 subset of unrecovered variable nodes of size at most ηK, where we will choose η later. Let $\mathcal{N}_i(S), i \in [C]$, be the check neighbors of S in hash i. If for at least one of the hashes $i \in [C]$, $|\mathcal{N}_i(S)| > \frac{|S|}{2}$, there must be at least one check node of degree 1 (a singleton) among the check neighbors $\mathcal{N}_i(S)$, thus 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 $\mathcal{E}_s = \cap_{i=1}^C \mathcal{E}_s^i$. By the construction of the ensemble $\mathcal{G}$, 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 $(\frac{s}{2B})^s$, where B is the total number of output hash bins. Taking the union bound over $\binom{B}{s/2}$ of all such sets, it is seen that $\mathbb{P}\left\{\mathcal{E}_s^i\right\} \leq \binom{B}{s/2}(\frac{s}{2B})^s$, which implies that $\mathbb{P}\{\mathcal{E}_s\} \leq \left(\binom{B}{s/2}(\frac{s}{2B})^s\right)^C$. Taking the union bound over all possible subsets of size s of variables, we obtain that

$$\begin{aligned}\mathbb{P}\{F_s\} &\leq \binom{K}{s}\mathbb{P}\{\mathcal{E}_s\} \leq \binom{K}{s}\left(\binom{B}{s/2}\left(\frac{s}{2B}\right)^s\right)^C \\ &\leq \left(\frac{eK}{s}\right)^s \left(\frac{2eB}{s}\right)^{sC/2}\left(\frac{s}{2B}\right)^{sC} \leq \frac{u^s s^{s(C/2-1)}}{K^{s(C/2-1)}}, \end{aligned}$$

where $u = e^{C/2+1}(\frac{\beta}{2})^{C/2}$ and where F_s denotes the event that the peeling decoder fails to decode a set of variables of size s. We also used the fact that for $n \geq m$, $\binom{n}{m} \leq (\frac{n\,e}{m})^m$. Moreover, $\mathbb{P}\{F_1\} = \mathbb{P}\{F_2\} = 0$ because the number of hashes C is always more than or equal to three. Selecting $\eta = \frac{1}{2u^{2/(C-2)}}$ and applying the union bound, we obtain that

$$\begin{aligned}\mathbb{P}\{F\} &\leq \sum_{s=1}^{\eta K}\mathbb{P}\{F_s\} = \sum_{s=3}^{\eta K}\mathbb{P}\{F_s\} = \sum_{s=3}^{\eta K}\frac{u^s s^{s(C/2-1)}}{K^{s(C/2-1)}} \\ &= O\left(\frac{1}{K^{3(C/2-1)}}\right) = O\left(\frac{1}{N^{3\alpha(C/2-1)}}\right), \end{aligned}$$

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

REFERENCES

[1] W. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proc. IEEE*, vol. 57, no. 1, pp. 58–68, Jan. 1969.

[2] 3GPP, *Spreading and Modulation (FDD)*, 3rd Generation Partnership Project (3GPP), document Rec. TS 25.213, Sep. 2014.

[3] E. D. Nelson and M. L. Fredman, "Hadamard spectroscopy," *J. Opt. Soc. Amer.*, vol. 60, no. 12, pp. 1664–1669, Dec. 1970.

[4] S. Haghighatshoar and E. Abbe. (2013). "Polarization of the Rényi information dimension for single and multi terminal analog compression." [Online]. Available: http://arxiv.org/abs/1301.6388

[5] A. Hedayat and W. D. Wallis, "Hadamard matrices and their applications," *Ann. Statist.*, vol. 6, no. 6, pp. 1184–1238, 1978.

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

[7] J. R. Johnson and M. Püschel, "In search of the optimal Walsh–Hadamard transform," in *Proc. IEEE ICASSP*, vol. 6. Istanbul, Turkey, Jun. 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 *Proc. 34th Annu. ACM STOC*, 2002, pp. 152–161.

[9] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A tutorial on fast Fourier sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 57–66, Mar. 2008.

[10] D. Lawlor, Y. Wang, and A. Christlieb, "Adaptive sub-linear Fourier algorithms," *Adv. Adapt. Data Anal.*, vol. 5, no. 1, p. 1350003, 2013.

[11] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse Fourier transform," in *Proc. 23rd Annu. ACM-SIAM SODA*, 2012, pp. 1183–1194.

[12] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Nearly optimal sparse Fourier transform," in *Proc. 44th Annu. ACM STOC*, 2012, pp. 563–578.

[13] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi, "Sample-optimal average-case sparse Fourier Transform in two dimensions," in *Proc. 51st Allerton*, Oct. 2013, pp. 1258–1265.

[14] S. Pawar and K. Ramchandran, "A hybrid DFT-LDPC framework for fast, efficient and robust compressive sensing," in *Proc. 50th Allerton*, 2012, pp. 1943–1950.

[15] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete Fourier transform using at most $4k$ samples and $O(k \log k)$ complexity," in *Proc. IEEE ISIT*, Istanbul, Turkey, Jul. 2013, pp. 464–468.

[16] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[17] O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions," in *Proc. 21st Annu. ACM STOC*, Feb. 1989, pp. 25–32.

[18] O. Goldreich, *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. New York, NY, USA: Springer-Verlag, 1999.

[19] N. C. Wormald, "Differential equations for random processes and random graphs," *Ann. Appl. Probab.*, vol. 5, no. 4, pp. 1217–1235, Nov. 1995.
[20] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 569–584, Feb. 2001.
[21] X. Li, J. K. Bradley, S. Pawar, and K. Ramchandran, "The SPRIGHT algorithm for robust sparse Hadamard transforms," in *Proc. IEEE ISIT*, Honolulu, HI, Jun./Jul. 2014, pp. 1857–1861.

Robin Scheibler (S'07) received the B.Sc. and M.Sc. in Communications Systems from École Polytechnique Fédérale de Lausanne (EPFL), Switzerland in 2006 and 2009, respectively. From 2009 to 2010, he was with IBM Research - Zürich. After this, he was with the NEC Media Information Processing group in Kawasaki, Japan from 2011 to 2012. He then returned to EPFL where he is working towards the Ph.D. degree at the Laboratory for Audiovisual Communications. His research interests lie in algorithmic signal processing, speech processing, and acoustic beamforming.

Saeid Haghighatshoar (S'12) received the B.Sc. degree in electrical engineering in 2007 in Electronics and the M.Sc. degree in electrical engineering in 2009 in Communication Systems both from Sharif University of Technology, Tehran, Iran, and the Ph.D. degree in Computer and Communication Sciences from École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. His research interests lie in Information theory, Communication systems, Graphical models and Compressed sensing.

Martin Vetterli (S'86–M'86–SM'90–F'95) received the Dipl. El.-Ing. degree from Eidgenössische Technische Hochschule (ETHZ) in 1981, the Master of Science degree from Stanford University in 1982, and the Doctoratès Sciences degree from École Polytechnique Fédérale de Lausanne (EPFL) in 1986.

After his dissertation, he was an Assistant and Associate Professor in Electrical Engineering at Columbia University in New York, and in 1993, he became an Associate and then Full Professor at the Department of Electrical Engineering and Computer Sciences at the University of California at Berkeley. In 1995, he joined the EPFL as a Full Professor. He held several positions at EPFL, including Chair of Communication Systems and founding director of the National Competence Center in Research on Mobile Information and Communication systems (NCCR-MICS). From 2004 to 2011 he was Vice President of EPFL for international affairs, and from 2011 to 2012, he was the Dean of the School of Computer and Communications Sciences. Since January 2013 he is President of the National Research Council of the Swiss National Science Foundation.

He works in the areas of electrical engineering, computer sciences and applied mathematics. His work covers wavelet theory and applications, image and video compression, self-organized communications systems and sensor networks, as well as fast algorithms, and has led to about 150 journals papers, as well as about 30 patents that led to technology transfer to high-tech companies and the creation of several start-ups.

He is the co-author of three textbooks, *Wavelets and Subband Coding* (with J. Kovacevic, Prentice-Hall, 1995), *Signal Processing for Communications* (P. Prandoni, EPFL Press, 2008) and *Foundations of Signal Processing* (with J. Kovacevic and V. Goyal, Cambridge University Press, 2014). These books are available in open access, and his research group follows the reproducible research philosophy.

His work won him numerous prizes, like best paper awards from EURASIP in 1984 and of the IEEE Signal Processing Society in 1991, 1996 and 2006, the Swiss National Latsis Prize in 1996, the SPIE Presidential award in 1999, the IEEE Signal Processing Technical Achievement Award in 2001 and the IEEE Signal Processing Society Award in 2010. He is a Fellow of IEEE, of ACM and EURASIP, was a member of the Swiss Council on Science and Technology (2000-2004), and is a ISI highly cited researcher in engineering.