

A FFAST framework for computing a k -sparse DFT in $O(k \log k)$ time using sparse-graph alias codes

Sameer Pawar and Kannan Ramchandran

Dept. of Electrical Engineering and Computer Sciences

University of California, Berkeley

{spawar, kannanr}@eecs.berkeley.edu

Abstract

Given an n -length input signal \vec{x} , it is well known that its Discrete Fourier Transform (DFT), \vec{X} , can be computed from n samples in $O(n \log n)$ operations using a Fast Fourier Transform (FFT) algorithm. If the spectrum \vec{X} is k -sparse (where $k \ll n$), can we do better? We show that asymptotically in k and n , when k is sub-linear in n (precisely, $k = O(n^\delta)$ where $0 < \delta < 1$), and the support of the non-zero DFT coefficients is uniformly random, our proposed FFAST (Fast Fourier Aliasing-based Sparse Transform) algorithm computes the DFT \vec{X} , from $O(k)$ samples in $O(k \log k)$ arithmetic operations, with high probability. Further, the constants in the big Oh notation for both sample and computational cost are small, e.g., when $\delta < 0.99$, which essentially covers almost all practical cases of interest, the sample cost is less than $4k$.

Our approach is based on filterless subsampling of the input signal \vec{x} using a set of *carefully chosen uniform subsampling patterns guided by the Chinese Remainder Theorem (CRT)*. The idea is to cleverly exploit, rather than avoid, the resulting aliasing artifacts induced by subsampling. Specifically, the subsampling operation on \vec{x} is designed to create aliasing patterns on the spectrum \vec{X} that “look like” parity-check constraints of a good erasure-correcting sparse-graph code. *Next, we show that computing the sparse DFT \vec{X} is equivalent to decoding of sparse-graph codes.* These codes further allow for fast peeling-style decoding. The resulting DFT computation is low in both sample complexity and decoding complexity. We analytically connect our proposed CRT-based aliasing framework to random sparse-graph codes, and analyze the performance of our algorithm using density evolution techniques from coding theory. We also provide simulation results, that are in tight agreement with our theoretical findings.

I. INTRODUCTION

Spectral analysis using the Discrete Fourier Transform (DFT) has been of universal importance in engineering and scientific applications for a long time. The Fast Fourier Transform (FFT) is the fastest known way to compute the DFT of an arbitrary n -length signal, and has a computational complexity of $O(n \log n)$ ¹. Many applications of interest involve signals, e.g. audio, image, video data, biomedical signals etc., which have a sparse Fourier spectrum. In such cases, a small subset of the spectral components typically contain most or all of the signal energy, with most spectral components being either zero or negligibly small. If the n -length DFT, \vec{X} , is k -sparse, where $k \ll n$, can we do better in terms of both **sample** and **computational** complexity of computing the sparse DFT? We answer this question affirmatively. In particular, we show that asymptotically in k and n , when k is sub-linear in n (precisely, $k = O(n^\delta)$ where $0 < \delta < 1$), and the support of the non-zero DFT coefficients is uniformly random, our proposed FFAST (Fast Fourier Aliasing-based Sparse Transform) algorithm computes the DFT \vec{X} , from judiciously chosen $O(k)$ samples in $O(k \log k)$ arithmetic operations, with high probability. Further, the constants in the big Oh notation for both sample and computational cost are small, e.g., when $\delta < 0.99$, which essentially covers almost all practical cases of interest, the sample cost is less than $4k$.

The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Istanbul, Turkey, 2013.

¹Recall that a single variable function $f(x)$ is said to be $O(g(x))$, if for a sufficiently large x the function $|f(x)|$ is bounded above by $|g(x)|$, i.e., $\lim_{x \rightarrow \infty} |f(x)| < c|g(x)|$ for some constant c . Similarly, $f(x) = \Omega(g(x))$ if $\lim_{x \rightarrow \infty} |f(x)| > c|g(x)|$ and $f(x) = o(g(x))$ if the growth rate of $|f(x)|$ as $x \rightarrow \infty$, is negligible as compared to that of $|g(x)|$, i.e. $\lim_{x \rightarrow \infty} |f(x)|/|g(x)| = 0$.

At a high level, our idea is to cleverly exploit rather than avoid the aliasing resulting from subsampling, and to shape the induced spectral artifacts to look like parity constraints of “good” erasure-correcting codes, e.g., Low-Density-Parity-Check (LDPC) codes [1] and fountain codes [2], that have both low computational complexity and are near-capacity achieving. Towards our goal of shaping the spectral artifacts to look like parity constraints of good erasure-correcting codes, we are challenged by not being able to design any arbitrary spectral code at will, as we can control only the subsampling operation on the time-domain signal. The key idea is to design subsampling patterns, guided by the *Chinese-Remainder-Theorem (CRT)* [3], that create the desired code-like aliasing patterns. Based on the qualitative nature of the subsampling patterns needed, our analysis is decomposed into two regimes (see Section VI and Section VII for more details):

- The “very-sparse” regime, where $k = O(n^\delta)$, $0 < \delta \leq 1/3$. For the very sparse regime the subsampling patterns are based on relatively co-prime numbers.
- The “less-sparse” regime, where $k = O(n^\delta)$, $1/3 < \delta < 1$. The sub-sampling pattern, for the less-sparse regime, comprise of “cyclically-shifted” overlapping co-prime integers.

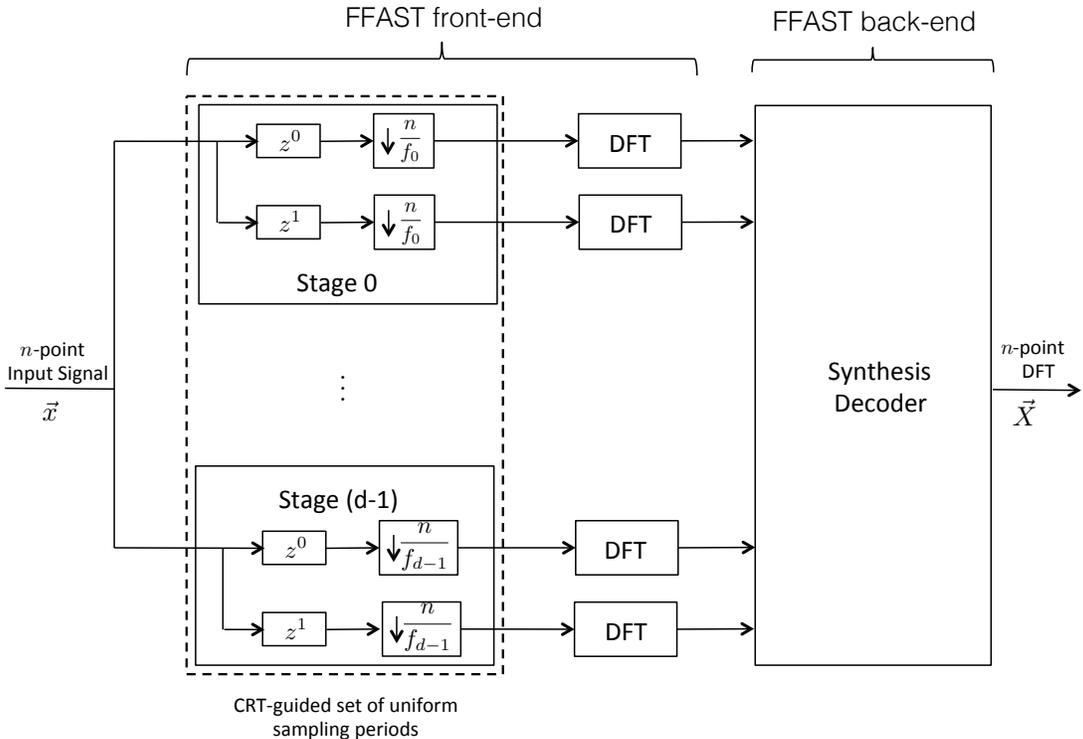


Fig. 1. Block diagram of the FFAST architecture. The n -length input signal \vec{x} is uniformly subsampled, through d stages. Each stage subsamples the input signal and its circularly shifted version by a carefully chosen (guided by the CRT as explained in Section VI and Section VII) set of sampling periods, to generate $O(k)$ samples per sub-sampling path. Next, the big n -length DFT \vec{X} is synthesized, from the short (length $O(k)$) DFTs, using the FFAST back-end decoder.

Our approach is summarized in Fig. 1. The n -length input signal \vec{x} is uniformly subsampled, through a small number² of stages, say d . Each stage subsamples the input signal and its circularly shifted version by a CRT guided set (see Sections VI and VII) of sampling periods, to generate $f_i = O(k)$ samples per sub-sampling path, for $i = 0, \dots, d-1$. Next, the large n -length DFT \vec{X} is synthesized from much smaller (length $O(k)$) DFTs, using the FFAST back-end peeling decoder. For the entire range of practical interest of sub-linear sparsity, i.e., $0 < \delta < 0.99$, the FFAST algorithm computes n -length k -sparse DFT \vec{X} in $O(k \log k)$ computations from less than $3.74k$ samples. It is gratifying to note that both the sample

²We show that the number of stages depend on the sparsity index δ , and is in the range of 3 to 8 for $\delta \leq 0.99$, as shown in Table II.

complexity and the computational complexity depend only on the sparsity parameter k , which is sub-linear in n .

We emphasize the following caveats. First, our analytical results are probabilistic, and hold for asymptotic values of k and n , with a success probability that approaches 1 asymptotically. This contrasts the $O(n \log n)$ FFT algorithm which works deterministically for all values of k and n . Secondly, we assume a uniformly random model for the support of the non-zero DFT coefficients. Lastly, we require the signal length n to be a product of a few (typically 3 to 9) distinct primes³. In effect, our algorithm trades off sample and computational complexity for asymptotically zero probability of failure guarantees, for a uniformly random model of the support. The FFAST algorithm is applicable whenever k is sub-linear in n (i.e. k is $o(n)$), but is obviously most attractive when k is much smaller than n . As a concrete example, when $k = 300$, and $n = 2^7 3^5 5^3 \approx 3.8 \times 10^6$, our algorithm achieves computational savings by a factor of more than 6000, and savings in the number of input samples by a factor of more than 3900 over the standard FFT (see [5] for computational complexity of a prime factor FFT algorithm). This can be a significant advantage in many existing applications, as well as enable new classes of applications not thought practical so far.

The focus of this paper is on signals having an exactly-sparse DFT. Our motivation for this focused study is three-fold: (i) to provide conceptual clarity of our proposed approach in a noiseless setting; (ii) to present our deterministic subsampling front-end measurement subsystem as a viable alternative to the class of randomized measurement matrices popular in the compressive sensing literature [6, 7]; and (iii) to explore the fundamental limits on both sample complexity and computational complexity for the exactly sparse DFT problem, which is of intellectual interest. In addition, the key insights derived from analyzing the exactly sparse signal model, apply more broadly to the noisy setting (see discussion in Section X), i.e., where the observations are further corrupted by noise [4].

The rest of the paper is organized as follows. Section II states the problem and introduce important notations. Section III presents our main results, and the related literature is reviewed in Section IV. Section V exemplifies the mapping of the problem of computing a sparse DFT to decoding over an appropriate sparse-graph code. In Section VI and Section VII we analyze the performance of the FFAST algorithm for the *very-sparse* and the *less-sparse* regimes respectively. Section IX provides simulation results that empirically corroborate our theoretical findings. The paper is concluded in Section X, with few remarks about extending the FFAST framework for noise robustness and other applications.

II. PROBLEM FORMULATION, NOTATIONS AND PRELIMINARIES

A. Problem formulation

Consider an n -length discrete-time signal \vec{x} that is sum of $k \ll n$ complex exponentials, i.e., its n -length discrete Fourier transform has k non-zero coefficients:

$$x[p] = \sum_{q=0}^{k-1} X[\ell_q] e^{2\pi i \ell_q p / n}, \quad p = 0, 1, \dots, n-1, \quad (1)$$

where the discrete frequencies $\ell_q \in \{0, 1, \dots, n-1\}$ and the amplitudes $X[\ell_q] \in \mathbb{C}$, for $q = 0, 1, \dots, k-1$. We further assume that the discrete frequencies ℓ_q are uniformly random in 0 to $n-1$ and the amplitudes $X[\ell_q]$ are drawn from some continuous distribution over complex numbers. Under these assumptions, we consider the problem of identifying the k unknown frequencies ℓ_q and the corresponding complex amplitudes $X[\ell_q]$ from the time domain samples \vec{x} . A straightforward solution is to compute an n -length DFT, \vec{X} , using a standard FFT algorithm [3], and locate the k non-zero coefficients. Such an algorithm

³This is not a major restriction for two reasons. Firstly, in many problems of interest, the choice of n is available to the system designer, and choosing n to be a power of 2 is often invoked only to take advantage of the readily-available radix-2 FFT algorithms. Secondly, by truncating or zero-padding the given signal, by a constant number of samples, one can obtain a modified signal of a desired length n . The DFT of the modified signal has dominant non-zero DFT coefficients at the same frequencies as the original signal but with additional noise-like side-lobes, due to windowing effect, and can be decoded using a noise robust version of the FFAST algorithm [4].

requires n samples and $O(n \log n)$ computations. When the DFT \vec{X} is known to be exactly k -sparse and $k \ll n$, computing all the n DFT coefficients seems redundant.

In this paper, we address the problem of designing an algorithm, to compute the k -sparse n -length DFT of \vec{x} for the asymptotic regime of k and n . We would like the algorithm to have the following features:

- it takes as few input samples m of \vec{x} as possible.
- it has a low computational cost, that is possibly a function of only the number of input samples m .
- it is applicable for the entire sub-linear regime, i.e., for all $0 < \delta < 1$, where $k = O(n^\delta)$.
- it computes the DFT \vec{X} with a probability of failure vanishing to 0 as m becomes large.

In Table I, we provide notations and definitions of the important parameters used in the rest of the paper.

Notation	Description
n	Signal length.
k	Number of non-zero coefficients in the DFT \vec{X} .
δ	Sparsity-index: $k = O(n^\delta), 0 < \delta < 1$.
m	Sample complexity: Number of samples of \vec{x} input to the FFAST algorithm.
$r = m/k$	Oversampling ratio: Input samples per non-zero DFT coefficient of the signal.
d	Number of stages in the sub-sampling ‘‘front-end’’ architecture of the FFAST.
f_i	Number of output samples at each sub-sampling path of stage i of the FFAST front-end.

TABLE I

GLOSSARY OF IMPORTANT NOTATIONS AND DEFINITIONS USED IN THE REST OF THE PAPER. THE LAST TWO PARAMETERS ARE RELATED TO THE FFAST ‘‘FRONT END’’ ARCHITECTURE (SEE FIGURE 1 FOR REFERENCE).

B. The Chinese-Remainder-Theorem (CRT)

The CRT plays an important role in our proposed FFAST architecture as well as in the FFAST decoder. For integers a, N , we use $(a)_N$ to denote the operation, $a \bmod N$, i.e., $(a)_N \triangleq a \bmod N$.

Theorem II.1 (Chinese-Remainder-Theorem [3]). *Suppose n_0, n_1, \dots, n_{d-1} are pairwise co-prime positive integers and $N = \prod_{i=0}^{d-1} n_i$. Then, every integer ‘ a ’ between 0 and $N - 1$ is uniquely represented by the sequence r_0, r_1, \dots, r_{d-1} of its remainders modulo n_0, \dots, n_{d-1} respectively and vice-versa.*

Further, given a sequence of remainders r_0, r_1, \dots, r_{d-1} , where $0 \leq r_i < n_i$, one can find an integer ‘ a ’, such that,

$$(a)_{n_i} \equiv r_i \quad \text{for } i = 0, 1, \dots, d - 1. \quad (2)$$

For example, consider the following pairwise co-prime integers $n_0 = 3, n_1 = 4$ and $n_2 = 5$. Then, given a sequence of remainders $r_0 = 2, r_1 = 2, r_2 = 3$, there exists a unique integer ‘ a ’, such that,

$$\begin{aligned} 2 &\equiv a \pmod{3} \\ 2 &\equiv a \pmod{4} \\ 3 &\equiv a \pmod{5} \end{aligned} \quad (3)$$

It is easy to verify that $a = 38$ is a unique integer, between 0 and 59, that satisfies the congruencies in (3).

III. MAIN RESULTS

We propose a novel FFAST architecture and algorithm to compute a k -sparse n -length DFT, \vec{X} , of an n -length signal \vec{x} . In this paper, an n -length input signal \vec{x} is said to have a k -sparse DFT \vec{X} , if \vec{X} has

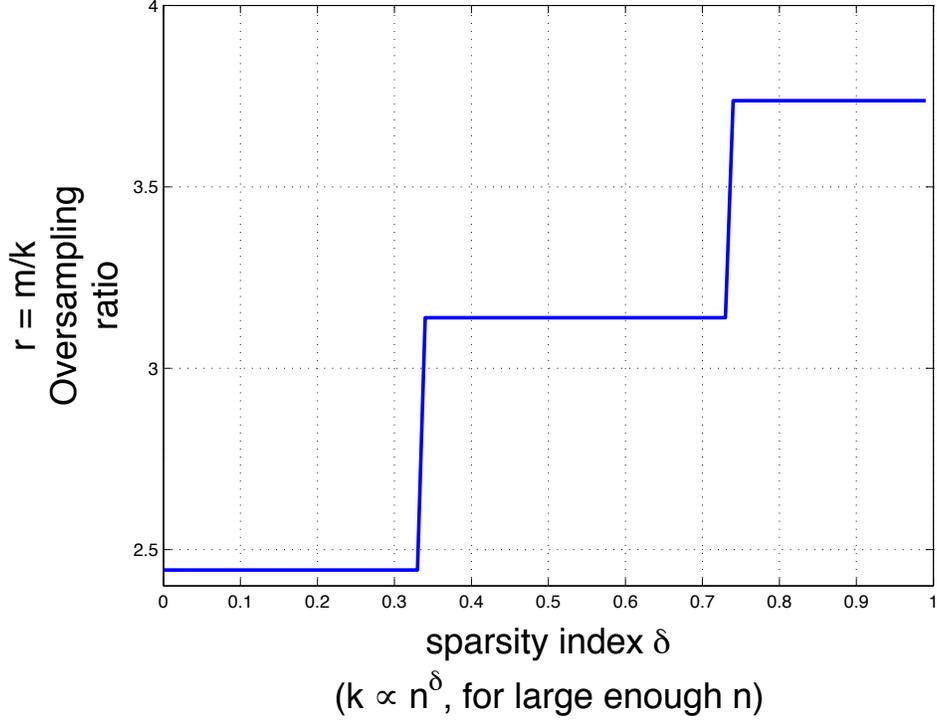


Fig. 2. The plot shows the relation between the oversampling ratio $r = m/k$, and the sparsity index δ , for $0 < \delta < 0.99$, where $k = O(n^\delta)$. The FFAST algorithm successfully computes the k -sparse n -length DFT \vec{X} of the desired n -length signal \vec{x} with high probability, as long as the number of samples m is above the threshold given in the plot. Note that for nearly the entire sub-linear regime of practical interest, e.g. $k < n^{0.99}$, the oversampling ratio $r < 4$. The above plot is an achievable performance of the FFAST algorithm using the constructions described in Section VI-D and Section VII-D. There are many other constructions of FFAST architecture that may achieve similar or better performance for a specific sparsity index δ .

at most k non-zero coefficients, whose locations are uniformly randomly distributed in $\{0, 1, \dots, n-1\}$. The FFAST algorithm computes the k -sparse n -length DFT with high probability, using as few as $O(k)$ samples of \vec{x} and $O(k \log k)$ computations. The following theorem states the main result of the paper.

Theorem III.1. *For any $0 < \delta < 1$, and large enough n , the FFAST algorithm computes a k -sparse DFT \vec{X} of an n -length input \vec{x} , where $k = O(n^\delta)$, with the following properties:*

- 1) **Sample complexity:** *The algorithm needs $m = rk$ samples of \vec{x} , where $r > 1$ is a small constant and depends on the sparsity index δ ;*
- 2) **Computational complexity:** *The FFAST algorithm computes DFT \vec{X} using $O(k \log k)$, arithmetic operations.*
- 3) **Probability of success:** *The probability that the algorithm correctly computes the k -sparse DFT \vec{X} is at least $1 - O(1/m)$.*

Proof: We prove the theorem in three parts. In Section VI, we analyze the performance of the FFAST algorithm for the very-sparse regime of $0 < \delta \leq 1/3$, and in Section VII we analyze the less-sparse regime of $1/3 < \delta < 1$. Lastly, in Section VIII we analyze the sample and computational complexity of the FFAST algorithm. ■

Remark III.2. *[Oversampling ratio r] The achievable oversampling ratio r in the sample complexity $m = rk$, depends on the number of stages d used in the FFAST architecture. The number of stages d , in turn, is a function of the sparsity index δ (recall $k = O(n^\delta)$), and increases as $\delta \rightarrow 1$ (i.e., as the number of the non-zero coefficients, k , approach the linear regime in n). In Sections VI and VII, we provide constructions of the FFAST front-end architecture, that require increasing number of stages d as δ increase from 0 to 1. In our proposed construction, the increase in d occurs over intervals of δ , resulting*

in a staircase plot as shown in Fig. 2. Note, that the plot in Fig. 2 is an achievable performance of the FFAST algorithm using the constructions described in Section VI-D and Section VII-D. There are many other constructions of FFAST architecture that may achieve similar or better performance for a specific sparsity index δ . Table II provides some example values of r and d for different values of the sparsity index δ .

δ	1/3	2/3	0.99	0.999	0.9999
d	3	6	8	11	14
r	2.45	3.14	3.74	4.64	5.51

TABLE II

THE TABLE SHOWS THE NUMBER OF SUBSAMPLING STAGES d USED IN THE FFAST ARCHITECTURE, AND THE CORRESPONDING VALUES OF THE OVERSAMPLING RATIO r , FOR DIFFERENT VALUES OF THE SPARSITY INDEX δ .

IV. RELATED WORK

The problem of computing a sparse discrete Fourier transform of a signal is related to the rich literature of frequency estimation [8, 9, 10, 11] in statistical signal processing as well as compressive-sensing [6, 7]. In frequency estimation, it is assumed that a signal consists of k complex exponentials embedded in additive noise. The frequency estimation techniques are based on well-studied statistical methods like MUSIC and ESPRIT [8, 9, 10, 11], where the focus is on ‘super-resolution’ spectral estimation from initial few consecutive samples, i.e., extrapolation. In contrast, the algorithm proposed in this paper combine tools from coding theory, number theory, graph theory and signal processing, to ‘interpolate’ the signal from interspersed but significantly less number of samples. In compressive sensing, the objective is to reliably reconstruct the sparse signal from as few measurements as possible, using a fast recovery technique. The bulk of this literature concentrates on random linear measurements, followed by either convex programming or greedy pursuit reconstruction algorithms [7, 12, 13]. An alternative approach, in the context of sampling a continuous time signal with a finite rate of innovation is explored in [14, 15, 16, 17]. Unlike the compressive sensing problem, where the resources to be optimized are the number of measurements⁴ and the recovery cost, in our problem, we want to minimize the *number of input samples* needed by the algorithm in addition to the recovery cost.

At a higher level though, despite some key differences in our approach to the problem of computing a sparse DFT, our problem is indeed closely related to the spectral estimation and compressive sensing literature, and our approach is naturally inspired by this, and draws from the rich set of tools offered by this literature.

A number of previous works [18, 19, 20, 21, 22] have addressed the problem of computing a 1-D DFT of a discrete-time signal that has a sparse Fourier spectrum, in sub-linear sample and time complexity. Most of these algorithms achieve a sub-linear time performance by first isolating the non-zero DFT coefficients into different bins, using specific filters or windows that have ‘good’ (concentrated) support in both, time and frequency. The non-zero DFT coefficients are then recovered iteratively, one at a time. The filters or windows used for the binning operation are typically of length $O(k \log(n))$. As a result, the sample and computational complexity is typically $O(k \log(n))$ or more. Moreover the constants involved in the big-Oh notation can be large, e.g., the empirical evaluation of [19] presented in [23] shows that for $n = 2^{22}$ and $k = 7000$, the number of samples required are $m \approx 2^{21} = 300k$ which is 75 times more than the sample complexity $4k$ of the FFAST algorithm⁵. The work of [20] provides an excellent tutorial on some

⁴Consider a compressive sensing problem with a measurement matrix A , i.e., $\mathbf{y} = A\bar{\mathbf{x}}$, where \mathbf{y} is a measurement vector and $\bar{\mathbf{x}}$ is the input signal. Then, the sample complexity is equal to the number of non-zero columns of A and the measurement complexity is equal to the number of non-zero rows of A .

⁵As mentioned earlier, the FFAST algorithm requires the length of the signal n to be a product of a few distinct primes. Hence, the comparison is for an equivalent $n \approx 2^{22}$ and $k = 7000$.

of the key ideas used by most sub-linear time sparse FFT algorithms in the literature. While we were writing this paper, we became aware of a recent work [24], in which the authors consider the problem of computing a noisy as well as an exactly-sparse 2-D DFT of size $\sqrt{n} \times \sqrt{n}$ signal. For an exactly sparse signal, and when $k = O(\sqrt{n})$, the algorithm in [24] uses $O(k)$ samples to compute the 2-D DFT of the signal in $O(k \log k)$ time with a constant probability of failure (that is controllable but that does not appear go to zero asymptotically). In [25], the author proposes a sub-linear time algorithm with a sample complexity of $O(k \log^4 n)$ or $O(k^2 \log^4 n)$ and computational complexity of $O(k \log^5 n)$ or $O(k^2 \log^4 n)$ to compute a sparse DFT, with high probability or zero-error respectively. The algorithm in [25] exploits the Chinese-Remainder-Theorem, along with $O(\text{poly}(\log n))$ number of subsampling patterns to identify the locations of the non-zero DFT coefficients. In contrast, the FFAST algorithm exploits the CRT to induce ‘good’ sparse-graph codes using a small constant number of subsampling patterns and computes the sparse DFT with a vanishing probability of failure.

V. COMPUTING DFT USING DECODING OF ALIAS-CODES

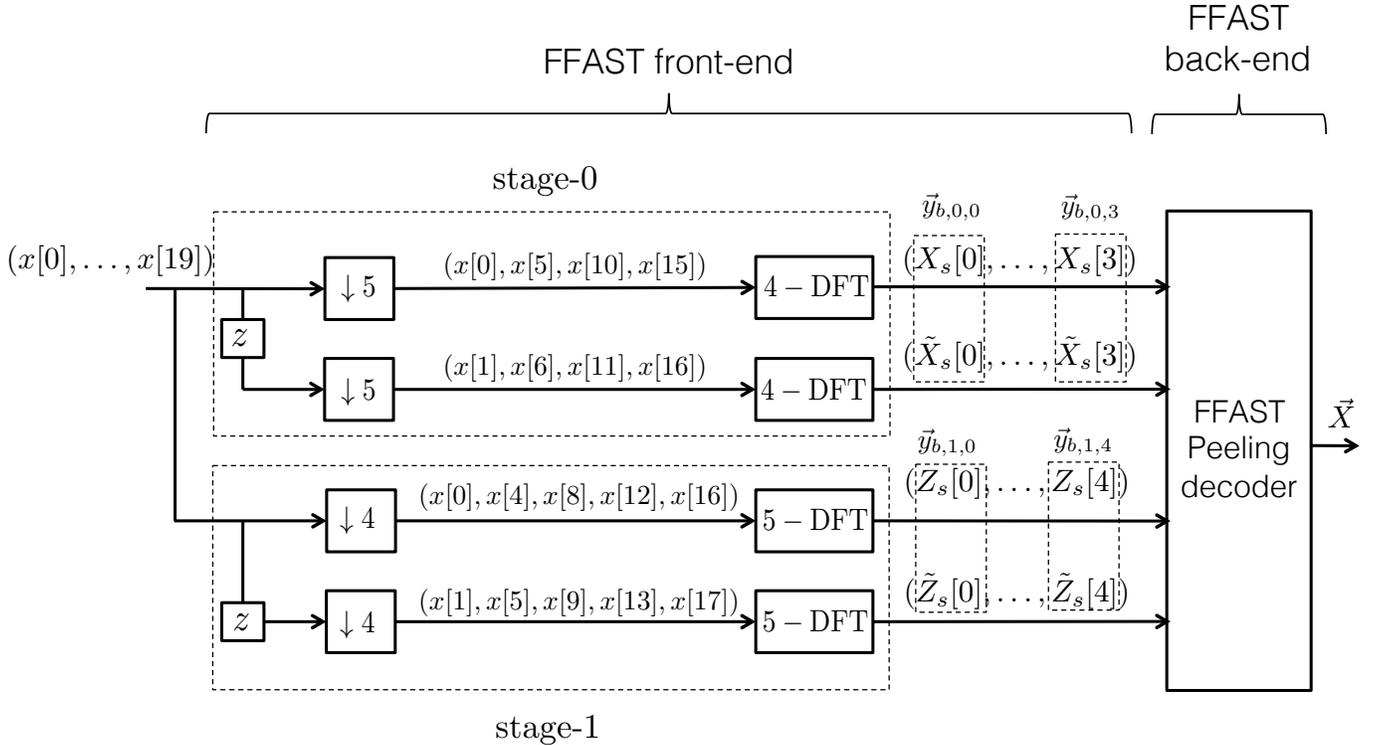


Fig. 3. An example FFAST architecture. The input to the FFAST architecture is a 20-length discrete-time signal $\vec{x} = (x[0], \dots, x[19])$. The input signal and its circularly shifted version are first subsampled by 5 to obtain two streams of sampled signal, also referred as delay-chains, each of length $f_0 = 4$. A 4-length DFT of the output of each delay-chain is then computed to obtain the observations $(X_s[\cdot], \tilde{X}_s[\cdot])$. Similarly, downsampling by 4 followed by a 5-length DFT provides the second set of $f_1 = 5$ observations $(Z_s[\cdot], \tilde{Z}_s[\cdot])$. Note that the number of output samples f_0 and f_1 in the two different stages are pairwise co-prime and are factors of $n = 20$. In general, the number of stages and the choice of the subsampling factors depend on the sparsity index δ , as will be described in Section VI and Section VII.

In this section, we use a simple example to illustrate the working mechanics of the FFAST sub-sampling ‘front-end’ and the associated ‘back-end’ peeling-decoder. Then we demonstrate, how the output of the FFAST front-end sub-sampling can be viewed as a sparse graph code in the frequency domain, which we refer to as ‘Alias-codes’, and computing the DFT is equivalent to decoding of this resulting alias-code. Later, we also point out a connection between the FFAST and coding techniques for a packet erasure channel.

Consider a 20-length discrete-time signal $\vec{x} = (x[0], \dots, x[19])$, such that its 20-length DFT \vec{X} , is 5-sparse. Let the 5 non-zero DFT coefficients of the signal \vec{x} be $X[1] = 1, X[3] = 4, X[5] = 2, X[10] = 3$ and $X[13] = 7$.

A. FFAST sub-sampling front-end

In general, the FFAST sub-sampling front-end architecture, as shown in Fig. 1, consists of multiple sub-sampling stages d and each stage further has 2 sub-sampling paths, referred to as “delay-chains”. The sampling periods used in both the delay-chains of a stage are identical. For example, consider a 2 stage FFAST sub-sampling front-end, shown in Fig. 3, that samples the 20-length input signal \vec{x} and its circularly shifted version⁶ by factors 5 and 4 respectively. In the sequel we use, f_i to denote the number of the output samples per delay-chain of stage i , e.g., $f_0 = 4$ and $f_1 = 5$ in Fig. 3. The FFAST front-end then computes short DFT’s of appropriate lengths of the individual output data streams of the delay-chains. Next, we group the output of short DFT’s into “bin-observation”.

1) **Bin observation:** A bin-observation is a 2-dimensional vector formed by collecting one scalar output value from the DFT output of the signal from each of the 2 delay chains in a stage. For example, $\vec{y}_{b,0,1}$ is an observation vector of bin 1 in stage 0 and is given by,

$$\vec{y}_{b,0,1} = \begin{pmatrix} X_s[1] \\ \tilde{X}_s[1] \end{pmatrix}. \quad (4)$$

The first index of the observation vector corresponds to the stage number, while the second index is the bin number within a stage. Note that in the FFAST architecture of Fig. 3, there are total of 4 bins in stage 0 and 5 bins in stage 1.

Using basic Fourier transform properties, reviewed below, of sampling, aliasing and circular shift, one can compute the relation between the original 20-length DFT \vec{X} of the signal \vec{x} and the output bin-observations $\{\vec{y}_{b,i,j}\}$ of the FFAST front-end.

- **Aliasing:** If a signal is subsampled in the time-domain, its frequency components mix together, i.e., alias, in a pattern that depends on the sampling procedure. For example, consider the output of the first delay-chain of the stage-0 in Fig. 3. The input signal \vec{x} is uniformly sampled by a factor of 5 to get $\vec{x}_s = (x[0], x[5], x[10], x[15])$. Then, the 4-length DFT of \vec{x}_s is related to the original 20-length DFT \vec{X} as:

$$\begin{aligned} X_s[0] &= X[0] + X[4] + X[8] + X[12] + X[16] = 0 \\ X_s[1] &= X[1] + X[5] + X[9] + X[13] + X[17] = 10 \\ X_s[2] &= X[2] + X[6] + X[10] + X[14] + X[18] = 3 \\ X_s[3] &= X[3] + X[7] + X[11] + X[15] + X[19] = 4 \end{aligned}$$

More generally, if the sampling period is N (we assume that N divides n) then,

$$X_s[i] = \sum_{j \equiv (i)_{n/N}} X[j], \quad (5)$$

where the notation $j \equiv (i)_{n/N}$, denotes $j \equiv i \pmod{n/N}$.

- **Circular Shift in time:** A circular shift in the time-domain results in a phase shift in the frequency-domain. Consider a circularly shifted signal $\vec{x}^{(1)}$ obtained from \vec{x} as $x^{(1)}[i] = x[(i+1)_n]$. The DFT coefficients of the shifted signal $\vec{x}^{(1)}$, are given as, $X^{(1)}[j] = \omega_n^j X[j]$, where $\omega_n = \exp(2\pi i/n)$ is an n^{th} root of unity. In general a circular shift of n_0 results in $X^{(n_0)}[j] = \omega_n^{jn_0} X[j]$.

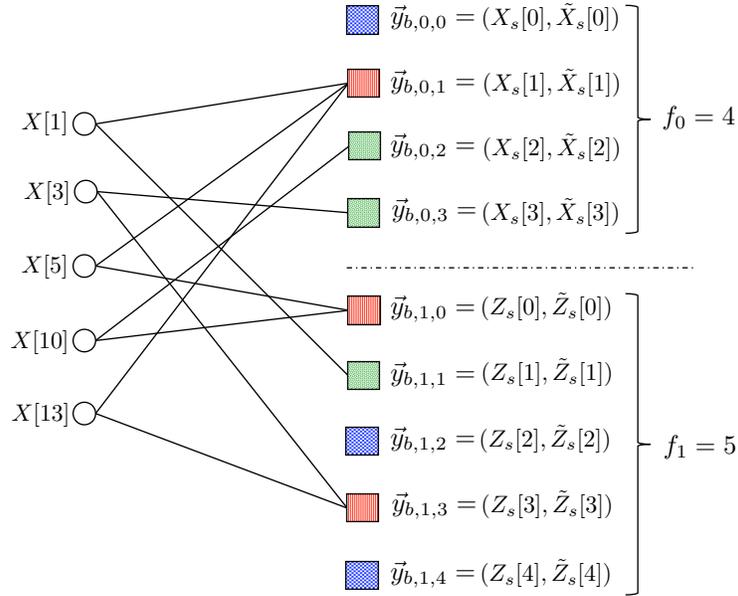


Fig. 4. A 2-left regular degree bi-partite graph representing relation between the unknown non-zero DFT coefficients, of the 20-length example signal \vec{x} , and the bin observations obtained through the FFAST front-end architecture shown in Fig. 3. Left nodes of the bi-partite graph represent the 5 non-zero DFT coefficients of the input signal \vec{x} , while the right nodes represent the “bins” (also sometimes referred in the sequel as “check nodes”) with vector observations. An edge connects a left node to a right node iff the corresponding non-zero DFT coefficient contributes to the observation vector of that particular bin. The observation at each check node is a 2-dimensional complex-valued vector, e.g., $\vec{y}_{b,0,0} = (X_s[0], \tilde{X}_s[0])$.

B. Alias-codes and its connection to computing sparse DFT from sub-sampled signal

In this section, we represent the relation between the original 20-length 5-sparse DFT \vec{X} and the FFAST front-end output, i.e., bin-observations, obtained using the Fourier properties, in a graphical form and interpret it as a “channel-code”. In particular, since this code is a result of a sub-sampling and aliasing operations, we refer to it as an “Alias-code”. We also establish that computing the sparse DFT \vec{X} of the signal \vec{x} from its sub-sampled data is equivalent to decoding the alias-code resulting from processing the input signal \vec{x} through the FFAST front-end.

Suppose that the 20-length example input signal \vec{x} is processed through a 2 stage FFAST front-end architecture shown in Fig. 3, to obtain the bin-observation vectors $(X_s[\cdot], \tilde{X}_s[\cdot])$ and $(Z_s[\cdot], \tilde{Z}_s[\cdot])$. Then, the relation between the 9 bin-observation vectors and the 5 non-zero DFT coefficients of the signal \vec{x} can be computed using the signal processing properties. A graphical representation of this relation is shown in Fig. 4. Left nodes of the bi-partite graph represent the 5 non-zero DFT coefficients of the input signal \vec{x} , while the right nodes represent the “bins” (also sometimes referred in the sequel as “check nodes”) with vector observations. An edge connects a left node to a right node iff the corresponding non-zero DFT coefficient contributes to the observation vector of that particular bin, e.g., after aliasing, due to sub-sampling, the DFT coefficient $X[10]$ contributes to the observation vector of bin 2 of stage 0 and bin 0 of stage 1. Thus, the problem of computing a 5-sparse 20-length DFT has been transformed into that of decoding the values and the support of the left nodes of the bi-partite graph in Fig. 4, i.e., decoding of alias-code. Next, we classify the observation bins based on its edge degree in the bi-partite graph, i.e., number of non-zero DFT coefficients contributing to a bin, which is then used to decode the alias-code.

- **zero-ton:** A bin that has no contribution from any of the non-zero DFT coefficients of the signal, e.g., bin 0 of stage 0 in Fig. 4. A zero-ton bin can be trivially identified from its observations.

⁶Conventionally, in signal processing literature z is used to denote a time-delay. In this paper, for ease of exposition we use z to denote a time-advancement (see Fig. 3 for an example).

- **single-ton**: A bin that has contribution from exactly one non-zero DFT coefficient of the signal, e.g., bin 2 of stage 0. Using the signal processing properties, it is easy to verify that the observation vector of bin 2 of stage 0 is given as,

$$\vec{y}_{b,0,2} = \begin{pmatrix} X[10] \\ e^{2\pi i 10/20} X[10] \end{pmatrix}.$$

The observation vector of a single-ton bin can be used to determine the support and the value, of the only non-zero DFT coefficient contributing to that bin, as follows:

- *support*: The support of the non-zero DFT coefficient contributing to a single-ton bin can be computed as,

$$10 = \frac{20}{2\pi} \angle \vec{y}_{b,0,2}[1] y_{b,0,2}^\dagger[0] \quad (6)$$

- *Value*: The value of the non-zero DFT coefficient is given by the observation $y_{b,0,2}[0]$.

We refer to this procedure as a “ratio-test”, in the sequel. Thus, a simple ratio-test on the observations of a single-ton bin correctly identifies the support and the value of the only non-zero DFT coefficient connected to that bin. It is easy to verify that this property holds for all the single-ton bins.

- **multi-ton**: A bin that has a contribution from more than one non-zero DFT coefficients of the signal, e.g., bin 1 of stage 0. The observation vector of bin 1 of stage 0 is,

$$\begin{aligned} \vec{y}_{b,0,1} &= X[1] \begin{pmatrix} 1 \\ e^{i2\pi/20} \end{pmatrix} + X[5] \begin{pmatrix} 1 \\ e^{i2\pi 5/20} \end{pmatrix} + X[13] \begin{pmatrix} 1 \\ e^{i2\pi 13/20} \end{pmatrix} \\ &= \begin{pmatrix} 10 \\ -3.1634 - i3.3541 \end{pmatrix} \end{aligned}$$

Now, if we perform the “ratio-test” on these observations, we get, the support to be 12.59. Since, we know that the support has to be an integer value between 0 to 19, we conclude that the observations do not correspond to a single-ton bin. In Appendix B, we rigorously show that the ratio-test identifies a multi-ton bin almost surely.

Hence, using the “ratio-test” on the bin-observations, the FFAST decoder can determine if a bin is a zero-ton, a single-ton or a multi-ton, almost surely. Also, when a bin is single-ton the ratio-test provides the support as well as the value of the non-zero DFT coefficient connected to that bin. In the following Section V-C, we provide the FFAST peeling-decoder that computes the support and the values of all the non-zero DFT coefficients of the signal \vec{x} .

C. FFAST back-end peeling-decoder

In the previous section we have explained how the *ratio-test* can be used to determine if a bin is a zero-ton, single-ton or a multi-ton bin almost surely. Also, in the case of a single-ton bin the ratio-test also identifies the support and the value of the non-zero DFT coefficient connected to that bin.

FFAST peeling-decoder: The FFAST decoder repeats the following steps (the pseudocode is provided in Algorithm 1 and Algorithm 2 in Appendix A):

- 1) Select all the edges in the graph with right degree 1 (edges connected to single-ton bins).
- 2) Remove these edges from the graph as well as the associated left and right nodes.
- 3) Remove all the other edges that were connected to the left nodes removed in step-2. When a neighboring edge of any right node is removed, its contribution is subtracted from that check node.

Decoding is successful if, at the end, all the edges have been removed from the graph. It is easy to verify that the FFAST peeling-decoder operated on the example graph of Fig. 4 results in successful decoding, with the coefficients being uncovered in the following possible order: $X[10]$, $X[3]$, $X[1]$, $X[5]$, $X[13]$.

Thus, the FFAST architecture has *transformed the problem of computing the DFT of \vec{x} into that of decoding alias-code* of Fig. 4. Clearly the success the FFAST decoder depends on the properties of the

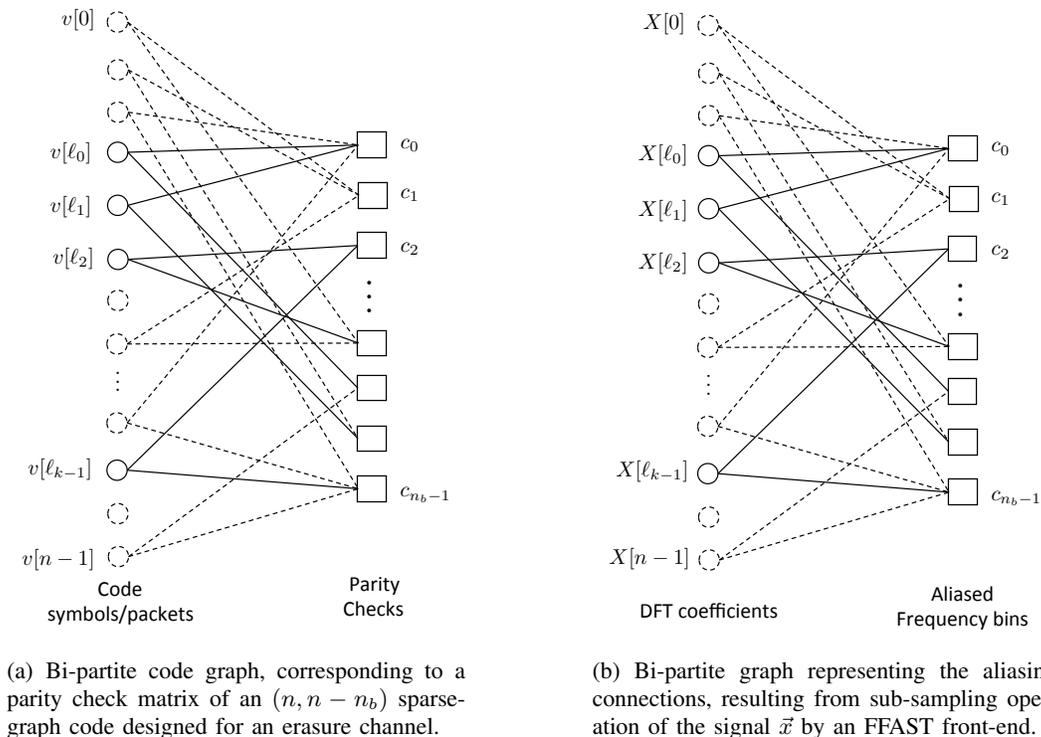


Fig. 5. Comparison between the bi-partite graphs corresponding to the parity check matrix of a sparse-graph code for an erasure channel and a graph induced by the FFAST front-end subsampling architecture.

sparse bi-partite graph resulting from the sub-sampling operation of the FFAST front-end. In particular, if the sub-sampling induced aliasing bi-partite graph is *peeling-friendly*, i.e., has few single-ton bins to initiate the peeling procedure and creates new single-tons at each iteration, until all the DFT coefficients are uncovered, the FFAST peeling-decoder succeeds in computing the DFT \vec{X} .

D. Connection to coding for packet erasure channels

The problem of decoding sparse-graph codes for erasure channel has been well studied in the coding theory literature. In this section we draw an analogy between decoding over sparse-graph codes for a packet erasure channel and decoding over sparse bi-partite graphs induced by the FFAST architecture.

a) Sparse-graph code for a packet-erasure channel: Consider an $(n, n - n_b)$ packet erasure code. Each n -length codeword consists of $(n - n_b)$ information packets and n_b parity packets. The erasure code is defined by a bi-partite graph as shown in Fig. 5(a). An n -length sequence of packets that satisfies the constraints defined by the graph in Fig. 5(a), i.e., sum of the packets connected to a parity check node equals zero, is a valid codeword. Suppose a codeword from the code, defined by the graph of Fig. 5(a), is transmitted over an erasure channel that uniformly at random drops some k number of packets. In Fig. 5(a), we use dotted circles to represent the correctly received packets (a cyclic redundancy check can be used to verify the correctly received packets). Let $\{v[\ell_0], \dots, v[\ell_{k-1}]\}$ be the k packets that are dropped/erased by the channel. The dotted edges in the bi-partite graph of Fig. 5(a) denote the operation of subtracting the contribution of all the correctly received packets from the corresponding check nodes. A peeling-decoder can now iteratively unravel the erased packets that are connected to a check node with exactly one erased packet. If the bipartite graph consisting only of the solid variable nodes and the solid edges is such that the peeling-decoder successfully unravels all the erased packets, decoding is successful.

b) Decoding over a bipartite graph induced by the FFAST: Consider an n -length signal \vec{x} that has a k -sparse DFT \vec{X} . Let the signal \vec{x} be sub-sampled by some appropriately designed FFAST front-end. The induced spectral-domain aliasing due to FFAST front-end sub-sampling operation is graphically

Erasure Channel	Sparse DFT
1. Explicitly designed sparse-graph code.	1. Implicitly designed sparse-graph code induced by sub-sampling.
2. $n - k$ correctly received packets.	2. $n - k$ zero DFT coefficients.
3. k -erased packets.	3. k unknown non-zero DFT coefficients
4. Peeling-decoder recovers the values of the erased packets using ‘single-ton’ check nodes. The identity (location) of the erased packets is <i>known</i> .	4. Peeling-decoder recovers <i>both</i> the values and the locations of the non-zero DFT coefficients using ‘single-ton’ check nodes. The locations of the non-zero DFT coefficients are <i>not known</i> . This results in a $2\times$ cost in the sample complexity.
5. Codes based on regular-degree bipartite graphs are near-capacity-achieving. More efficient, capacity-achieving irregular-degree bipartite graph codes can be designed.	5. The FFAST architecture based on uniform subsampling can induce only left-regular degree bi-partite graphs.

TABLE III

COMPARISON BETWEEN DECODING OVER A SPARSE-GRAPH CODE FOR A PACKET ERASURE CHANNEL AND COMPUTING A SPARSE DFT USING THE FFAST ARCHITECTURE.

represented by a bipartite graph shown in Fig. 5(b). The variable (left) nodes correspond to the n -length DFT \vec{X} and the check (right) nodes are the bins consisting of the aliased DFT coefficients. A variable node is connected to a check node, iff after aliasing that particular DFT coefficient contributes to the observation of the considered check node. Let $\{X[\ell_0], X[\ell_1], \dots, X[\ell_{k-1}]\}$ be the k non-zero DFT coefficients. The zero DFT coefficients are represented by dotted circles. The dashed edges in Fig. 5(b) denotes that the contribution to the bin-observation, due to this particular edge is zero. Using the ratio-test on the vector observation at each check-node one can determine if the check node is a “single-ton”, i.e., has exactly one solid edge. A peeling-decoder can now iteratively unravel the non-zero DFT coefficients connected to single-ton check nodes. If the bipartite graph consisting only of the solid variable nodes and the solid edges is such that peeling-decoder successfully unravels all the variable nodes, the algorithm succeeds in computing the DFT \vec{X} . In Table III we provide a comparison between decoding over bi-partite graphs of Fig. 5(a) and Fig. 5(b).

Thus, the problem of decoding bi-partite graphs corresponding to sparse-graph codes designed for a packet-erasure channel is closely related to decoding the sparse bi-partite graphs induced by the FFAST architecture. We use this analogy: a) to design a sub-sampling front-end that induces a ‘good’ left-regular degree sparse-graph codes; and b) to formally connect our proposed Chinese-Remainder-Theorem based aliasing framework to a random sparse-graph code constructed using a balls-and-bins model (explained in Section VI-A), and analyze the convergence behavior of our algorithm using well-studied density evolution techniques from coding theory.

Next, we address the question of how to carefully design the sub-sampling parameters of the FFAST front-end architecture so as to induce “good-graphs” or “alias-codes”. In Section VI and Section VII we provide constructions of the FFAST front-end architecture and analyze the performance of the FFAST peeling-decoder for the *very-sparse*, i.e., $0 < \delta \leq 1/3$, and the *less-sparse*, i.e., $1/3 < \delta < 1$, regimes of sparsity respectively.

VI. FFAST CONSTRUCTION AND PERFORMANCE ANALYSIS FOR THE *very-sparse* ($k = O(n^\delta)$, $0 < \delta \leq 1/3$) REGIME

In Section V, using an example, we illustrated that the problem of computing a k -sparse n -length DFT of a signal can be transformed into a problem of decoding over sparse bipartite graphs using the FFAST architecture. In this section, we provide a choice of parameters of the FFAST front-end architecture and analyze the probability of success of the FFAST peeling-decoder for the very-sparse regime of $0 < \delta < 1/3$. As shown in Section V-D, the FFAST decoding process is closely related to the decoding procedure on sparse-graph codes designed for erasure channels. From the coding theory literature, we know that there exist several sparse-graph code constructions that are low-complexity and capacity-achieving for the erasure channels. The catch for us is that we are not at liberty to use any arbitrary bi-partite graph, but *can choose only those graphs that correspond to the alias-codes, i.e., are induced via aliasing through our proposed FFAST subsampling front-end*. How do we go about choosing the right parameters and inducing the good graphs?

We describe two ensembles of bi-partite graphs. The first ensemble is based on a “balls-and-bins” model, while the second ensemble is based on the CRT. The balls-and-bins model based ensemble of graphs is closer in spirit to the sparse-graph codes in the coding-theory literature. Hence, is amenable to a rigorous analysis using coding-theoretic tools like density-evolution [26]. The bi-partite graphs induced by the FFAST front-end sub-sampling operation belong to the CRT ensemble. Later, in Lemma VI.1, we show that the two ensembles are equivalent. Hence, the analysis of the balls-and-bins construction carries over to the FFAST. We start by setting up some notations and common parameters.

Consider a set $\mathcal{F} = \{f_0, \dots, f_{d-1}\}$ of pairwise co-prime integers. Let the signal length $n = \mathcal{P} \prod_{i=0}^{d-1} f_i$, for some positive integer $\mathcal{P} \geq 1$, and $n_b \triangleq \sum_{i=0}^{d-1} f_i$. The integers f_i 's are chosen such that they are approximately equal and we use \mathbf{F} to denote this value. More precisely, $f_i = \mathbf{F} + O(1)$, for $i = 0, \dots, d-1$, where \mathbf{F} is an asymptotically large number. The $O(1)$ perturbation term in each f_i is used to obtain a set of co-prime integers⁷ approximately equal to \mathbf{F} . We construct a FFAST sub-sampling front-end architecture with d stages. Each stage further has 2 delay-chains (see Fig. 1 for reference). The sub-sampling period used in both delay-chains of stage i is n/f_i and hence the number of output samples is f_i . The total number of input samples used by the FFAST algorithm is $m = 2d\mathbf{F} + O(1)$ (see Fig. 1). In this section, we use $\mathbf{F} = \eta k$, for some constant $\eta > 0$. This results in a sparsity index $0 < \delta \leq 1/d$, depending on the value of the integer \mathcal{P} .

A. Ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ of bi-partite graphs constructed using a “Balls-and-Bins” model

Bi-partite graphs in the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, have k variable nodes on the left and n_b check nodes on the right. Further, each variable (left) node is connected to d right nodes, i.e., left-regular degree bi-partite graphs. An example graph from an ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, for $\mathcal{F} = \{4, 5\}$, $d = 2$, $k = 5$ and $n_b = 9$ is shown in Fig. 4. More generally, the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ of d -left regular edge degree bipartite graphs constructed using a “balls-and-bins” model is defined as follows. Set $n_b = \sum_{i=0}^{d-1} f_i$, where $\mathcal{F} = \{f_i\}_{i=0}^{d-1}$. Partition the set of n_b check nodes into d subsets with the i^{th} subset having f_i check nodes. For each variable node, choose one neighboring check node in each of the d subsets, uniformly at random. The corresponding d -left regular degree bipartite graph is then defined by connecting the variable nodes with their neighboring check nodes by an undirected edge.

An edge e in the graph is represented as a pair of nodes $e = \{v, c\}$, where v and c are the variable and check nodes incident on the edge e . By a directed edge \vec{e} we mean an ordered pair (v, c) or (c, v) corresponding to the edge $e = \{v, c\}$. A path in the graph is a directed sequence of directed edges $\vec{e}_1, \dots, \vec{e}_t$ such that, if $\vec{e}_i = (u_i, u'_i)$, then the $u'_i = u_{i+1}$ for $i = 1, \dots, t-1$. The length of the path is the number of directed edges in it, and we say that the path connecting u_1 to u_t starts from u_1 and ends at u_t .

⁷An example construction of an approximately equal sized 3 co-prime integers can be obtained as follows. Let $\mathbf{F} = 2^{r_0}3^{r_1}5^{r_2}$ for any integers r_0, r_1, r_2 greater than 1. Then, $f_0 = \mathbf{F} + 2$, $f_1 = \mathbf{F} + 3$ and $f_2 = \mathbf{F} + 5$ are co-prime integers.

1) *Directed Neighborhood*: The *directed neighborhood of depth ℓ* of $\vec{e} = (v, c)$, denoted by $\mathcal{N}_{\vec{e}}^{\ell}$, is defined as the induced subgraph containing all the edges and nodes on paths $\vec{e}_1, \dots, \vec{e}_{\ell}$ starting at node v such that $\vec{e}_1 \neq \vec{e}$. An example of a directed neighborhood of depth $\ell = 2$ is given in Fig. 6. If the induced sub-graph corresponding to the directed neighborhood $\mathcal{N}_{\vec{e}}^{\ell}$ is a tree then we say that the depth- ℓ neighborhood of the edge \vec{e} is *tree-like*.

B. Ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ of bipartite graphs constructed using the Chinese-Remainder-Theorem (CRT)

The ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ of d -left regular degree bipartite graphs, with k variable nodes and n_b check nodes, is defined as follows. Partition the set of n_b check nodes into d subsets with the i^{th} subset having f_i check nodes (see Fig. 4 for an example). Consider a set \mathcal{I} of k integers, where each element of the set \mathcal{I} is between 0 and $n - 1$. Assign the k integers from the set \mathcal{I} to the k variable nodes in an arbitrary order. Label the check nodes in the set i from 0 to $f_i - 1$ for all $i = 0, \dots, d - 1$. A d -left regular degree bi-partite graph with k variable nodes and n_b check nodes, is then obtained by connecting a variable node with an associated integer v to a check node $(v)_{f_i}$ in the set i , for $i = 0, \dots, d - 1$. The ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ is a collection of all the d -left regular degree bipartite graphs induced by all possible sets \mathcal{I} .

Lemma VI.1. *The ensemble of bipartite graphs $\mathcal{C}_1^k(\mathcal{F}, n_b)$ is identical to the ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$.*

Proof: It is trivial to see that $\mathcal{C}_2^k(\mathcal{F}, n, n_b) \subset \mathcal{C}_1^k(\mathcal{F}, n_b)$. Next we show the reverse. Consider a graph $\mathcal{G}_1 \in \mathcal{C}_1^k(\mathcal{F}, n_b)$. Suppose, a variable node $v \in \mathcal{G}_1$ is connected to the check nodes numbered $\{r_i\}_{i=0}^{d-1}$. Then, using the CRT, one can find P number of integer's 'q' between 0 and $n - 1$ such that $(q)_{f_i} = r_i \forall i = 0, \dots, d - 1$. Thus, for every graph $\mathcal{G}_1 \in \mathcal{C}_1^k(\mathcal{F}, n_b)$, there exists a set \mathcal{I} of k integers, that will result in an identical graph using the CRT based construction. Hence, $\mathcal{C}_1^k(\mathcal{F}, n_b) = \mathcal{C}_2^k(\mathcal{F}, n, n_b)$. ■

Note that the modulo rule used to generate a graph in the ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ is same as the one used in equation (5) of Section V. Thus, the FFAST architecture of Fig. 1, *generates graphs from the CRT ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$* , where the indices \mathcal{I} of the k variable nodes correspond to the locations (or support) of the non-zero DFT coefficients⁸ of the signal \vec{x} . Also, under the assumption that the support of the non-zero DFT coefficients of the signal \vec{x} is uniformly random, the resulting aliasing graph is uniformly random choice from the ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$.

Next, we analyze the performance of the FFAST peeling-decoder over a uniformly random choice of a graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, which along with the Lemma VI.1, provides a lower bound on the success performance of the FFAST decoder over graphs from the ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$. Although the construction and the results described in this section are applicable to any value of d , we are particularly interested in the case when $d = 3$. For $d = 3$ and $\mathcal{P} = O(1)$, we achieve the sub-linear sparsity index $\delta = 1/3$, while other values of $0 < \delta < 1/3$, are achieved using larger values of \mathcal{P} .

C. Performance analysis of the FFAST peeling-decoder on graphs from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$

In this section, we analyze the probability of success of the FFAST peeling-decoder, over a randomly chosen graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, after a fixed number of peeling iterations ℓ . Our analysis follows exactly the arguments in [26] and [27]. Thus, one may be tempted to take the results from [26] ‘‘off-the-shelf’’. However, we choose here to provide a detailed analysis for two reasons. First, our graph construction in the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ is different from that used in [26], which results in some fairly important differences in the analysis, such as the expansion properties of the graphs, thus warranting an independent analysis. Secondly, we want to make the paper more self-contained and complete.

We now provide a brief outline of the proof elements highlighting the main technical components needed to show that the FFAST peeling-decoder decodes all the non-zero DFT coefficients with high probability.

⁸A set \mathcal{I} with repeated elements corresponds to a signal with fewer than k non-zero DFT coefficients.

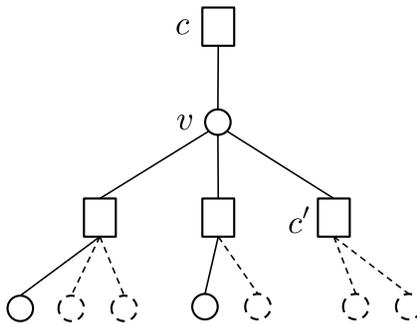


Fig. 6. Directed neighborhood of depth 2 of an edge $\vec{e} = (v, c)$. The dashed lines correspond to nodes/edges removed at the end of iteration j . The edge between v and c can be potentially removed at iteration $j + 1$ as one of the check nodes c' is a single-ton (it has no more variable nodes remaining at the end of iteration j).

- *Density evolution:* We analyze the performance of the message-passing algorithm, over a typical graph from the ensemble, for ℓ iterations. First, we assume that a local neighborhood of depth 2ℓ of every edge in a typical graph in the ensemble is tree-like, i.e., cycle-free. Under this assumption, all the messages between variable nodes and the check nodes, in the first ℓ rounds of the algorithm, are independent. Using this independence assumption, we derive a recursive equation that represents the expected evolution of the number of single-tons uncovered at each round for this typical graph.
- *Convergence to the cycle-free, case:* Using a Doob martingale as in [26], we show that a random graph from the ensemble, chosen as per nature's choice of the non-zero DFT coefficients, behaves like a "typical" graph, i.e., 2ℓ -depth neighborhood of most of the edges in the graph is cycle-free, with high probability. This proves that for a random graph in $\mathcal{C}_1^k(\mathcal{F}, n_b)$, the FFAST peeling-decoder decodes all but an arbitrarily small fraction of the variable nodes with high probability in a constant number of iterations, ℓ .
- *Completing the decoding using the graph expansion property:* We first show that if a graph is an "expander" (as will be defined later in Section VI-C3), and the FFAST peeling-decoder successfully decodes all but a small fraction of the non-zero DFT coefficients, then it decodes all the non-zero DFT coefficients successfully. Next, we show that a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ is an expander with high probability.

1) **Density evolution for local tree-like view:** In this section we assume that a local neighborhood of depth 2ℓ of every edge in a graph in the ensemble is tree-like. Next, we define the edge-degree distribution polynomials of the bipartite graphs in the ensemble as $\lambda(\alpha) \triangleq \sum_{i=1}^{\infty} \lambda_i \alpha^{i-1}$ and $\rho(\alpha) \triangleq \sum_{i=1}^{\infty} \rho_i \alpha^{i-1}$, where λ_i (resp. ρ_i) denotes the probability that an edge of the graph is connected to a left (resp. right) node of degree i . Thus for the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, constructed using the balls-and-bins procedure, $\lambda(\alpha) = \alpha^{d-1}$ by construction. Further, as shown in Appendix C, the edge degree distribution $\rho(\alpha) = \exp(-(1-\alpha)/\eta)$.

Let p_j denote the probability that an edge is present (or undecoded) after round j of the FFAST peeling-decoder, then $p_0 = 1$. Under the tree-like assumption, the probability p_{j+1} , is given as,

$$p_{j+1} = \lambda(1 - \rho(1 - p_j)) \quad j = 0, 1, \dots, \ell - 1. \quad (7)$$

Equation (7) can be understood as follows (also see Fig. 6): the tree-like assumption implies that, up to iteration ℓ , messages on different edges are independent. Thus, the total probability, that at iteration $j + 1$, a variable node v is *decoded* due to a particular check node is given by $\rho(1 - p_j) = \sum_{i=1}^{\infty} \rho_i (1 - p_j)^{i-1}$ and similarly the total probability that none of the neighboring check nodes decode the variable node v is $p_{j+1} = \lambda(1 - \rho(1 - p_j))$. Specializing equation (7) for the edge degree distributions of $\mathcal{C}_1^k(\mathcal{F}, n_b)$ we get,

$$p_{j+1} = \left(1 - e^{-\frac{p_j}{\eta}}\right)^{d-1}, \quad \forall j = 0, 1, \dots, \ell - 1 \quad (8)$$

where $p_0 = 1$. The evolution process of (8) asymptotically (in the number of iterations ℓ) converges to 0, for an appropriate choice of the parameter η , e.g., see Table IV.

d	2	3	4	5	6	7	8	9
η	1.0000	0.4073	0.3237	0.2850	0.2616	0.2456	0.2336	0.2244
$d\eta$	2.0000	1.2219	1.2948	1.4250	1.5696	1.7192	1.8688	2.0196

TABLE IV

MINIMUM VALUE OF η , REQUIRED FOR THE DENSITY EVOLUTION OF (8) TO CONVERGE ASYMPTOTICALLY. THE THRESHOLD VALUE OF η DEPENDS ON THE NUMBER OF STAGES d .

2) **Convergence to cycle-free case:** In the following Lemma VI.2 we show; a) the expected behavior over all the graphs in the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ converges to that of a cycle-free case, and b) with exponentially high probability, the proportion of the edges that are not decoded after ℓ iterations of the FFAST peeling-decoder is tightly concentrated around p_ℓ , as defined in (8).

Lemma VI.2 (Convergence to Cycle-free case). *Over the probability space of all graphs $\mathcal{C}_1^k(\mathcal{F}, n_b)$, let Z be the total number of edges that are not decoded after ℓ (an arbitrarily large but fixed) iterations of the FFAST peeling-decoder over a randomly chosen graph. Further, let p_ℓ be as given in the recursion (8). Then there exist constants β and γ such that for any $\epsilon_1 > 0$ and sufficiently large k we have*

$$(a) \quad \mathbb{E}[Z] < 2kdp_\ell. \quad (9)$$

$$(b) \quad \Pr(|Z - \mathbb{E}[Z]| > kd\epsilon_1) < e^{-\beta\epsilon_1^2 k^{1/(4\ell+1)}}. \quad (10)$$

Proof: Please see Appendix F. ■

3) **Successful Decoding using Expansion:** In the previous section we have shown that with high probability, the FFAST peeling-decoder decodes all but an arbitrarily small fraction of variable nodes. In this section, we show how to complete the decoding if the graph is a “good-expander”. Our problem requires the following definition of an “expander-graph”, which is somewhat different from conventional notions of an expander-graph in literature, e.g., *edge expander*, *vertex expander* or *spectral expander* graphs.

Definition VI.3 (Expander graph). *A d -left regular degree bipartite graph from ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, is called an (α, β, d) expander, if for all subsets S , of variable nodes, of size at most αk , there exists a right neighborhood of S , i.e., $N_i(S)$, that satisfies $|N_i(S)| > \beta|S|$, for some $i = 0, \dots, d-1$.*

In the following lemma, we show that if a graph is an expander, and if the FFAST peeling-decoder successfully decodes all but a small fraction of the non-zero DFT coefficients, then it decodes all the non-zero DFT coefficients successfully.

Lemma VI.4. *Consider a graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, with $|\mathcal{F}| = d$, that is an $(\alpha, 1/2, d)$ expander for some $\alpha > 0$. If the FFAST peeling-decoder over this graph succeeds in decoding all but at most αk variable nodes, then it decodes all the variable nodes.*

Proof: See Appendix D ■

In Lemma VI.5, we show that most of the graphs in the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ are expanders.

Lemma VI.5. *Consider a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, where $d \geq 3$. Then, all the subsets S of the variable nodes, of the graph, satisfy $\max\{|N_i(S)|\}_{i=0}^{d-1} > |S|/2$,*

a) *with probability at least $1 - e^{-\epsilon k \log(n_b/k)}$, for sets of size $|S| = \alpha k$, for small enough $\alpha > 0$ and some $\epsilon > 0$.*

b) *with probability at least $1 - O(1/n_b)$, for sets of size $|S| = o(k)$.*

Proof: See Appendix E ■

The condition $d \geq 3$ is a necessary condition for part (b) of Lemma VI.5. This can be seen as follows. Consider a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, where $|\mathcal{F}| = d$. If any two variable nodes in the graph have the same set of d neighboring check nodes, then the expander condition, for the set S consisting of these two variable nodes, will not be satisfied. In a bi-partite graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, there are a total of $O(k^d)$ distinct sets of d check nodes. Each of the k variable nodes chooses a set of d check nodes, uniformly at random and with replacement, from the total of $O(k^d)$ sets. If we draw k integers uniformly at random between 0 to $n - 1$, the probability $Pr(k; n)$ that at least two numbers are the same is given by,

$$Pr(k; n) \approx 1 - e^{-k^2/2n}. \quad (11)$$

This is also known as the *birthday paradox* or the *birthday problem* in literature [28]. For a graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, we have $n = O(k^d)$. Hence, if the number of stages $d \leq 2$, there is a constant probability that there exists a pair of variable nodes that share the same neighboring check nodes, in both the stages, thus violating the expander condition.

Theorem VI.6. *The FFAST peeling-decoder over a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, where $d \geq 3$ and $\mathbf{F} = \eta k$:*

- a) *successfully uncovers all the variable nodes with probability at least $1 - O(1/n_b)$;*
- b) *successfully uncovers all but a vanishingly small fraction, i.e., $o(k)$, of the variable nodes with probability at least $1 - e^{-\beta \epsilon_1^2 k^{1/(4\ell+1)}}$ for some constants $\beta, \epsilon_1 > 0$, and $\ell > 0$.*

for an appropriate choice of the constant η as per Table IV.

Proof: Consider a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$. Let Z be the number of the edges not decoded by the FFAST peeling-decoder in ℓ (large but fixed constant) iterations after processing this graph. Then, from recursion (8) and Lemma VI.2, for an appropriate choice of the constant η (as per Table IV), $Z \leq \alpha k$, for an arbitrarily small constant $\alpha > 0$, with probability at least $1 - e^{-\beta \epsilon_1^2 k^{1/(4\ell+1)}}$. The result then follows from Lemmas VI.5 and VI.4. ■

Corollary VI.7. *The FFAST peeling-decoder over a random graph from the ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$, where $d \geq 3$ and $\mathbf{F} = \eta k$:*

- a) *successfully uncovers all the variable nodes with probability at least $1 - O(1/n_b)$;*
- b) *successfully uncovers all but a vanishingly small fraction, i.e., $o(k)$, of the variable nodes with probability at least $1 - e^{-\beta \epsilon_1^2 k^{1/(4\ell+1)}}$ for some constants $\beta, \epsilon_1 > 0$, and $\ell > 0$.*

for an appropriate choice of the constant η as per Table IV.

Proof: Follows from equivalence of ensembles Lemma VI.1 and Theorem VI.6. ■

D. The FFAST front-end architecture parameters for achieving the sparsity index $0 < \delta \leq 1/3$

Consider a set $\mathcal{F} = \{f_0, f_1, f_2\}$ of a pairwise co-prime integers. The integers f_i 's are such that they are approximately equal, i.e., $f_i = \mathbf{F} + O(1)$, for $i = 0, 1, 2$, where $\mathbf{F} = 0.4073k$ (see Table IV) is an asymptotically large number. Set the signal length $n = \mathcal{P} \prod_{i=0}^2 f_i$, where, $\mathcal{P} = \mathbf{F}^a$, thus achieving the sparsity index $\delta = 1/(3 + a)$, for any positive constant $a > 0$. We construct a FFAST sub-sampling front-end with $d = 3$ stages, where each stage further has 2 delay-chains (see Fig. 1). The sub-sampling period used in the both the delay-chains of the i^{th} stage is n/f_i . As a result, the number of samples at the output of each delay-chain of the i^{th} stage is f_i for $i = 0, 1, 2$, i.e., the total number of samples m used by the FFAST algorithm is $m = 2(f_0 + f_1 + f_2) < 2.45k$.

VII. FFAST CONSTRUCTION AND PERFORMANCE ANALYSIS FOR THE *less-sparse*

($k = O(n^\delta)$, $1/3 < \delta < 1$) REGIME

In the FFAST front-end architecture for the less-sparse regime, the integers in the set $\mathcal{F} = \{f_0, \dots, f_{d-1}\}$, unlike for the very-sparse regime of Section VI, are not pairwise co-prime. Instead, for the less-sparse

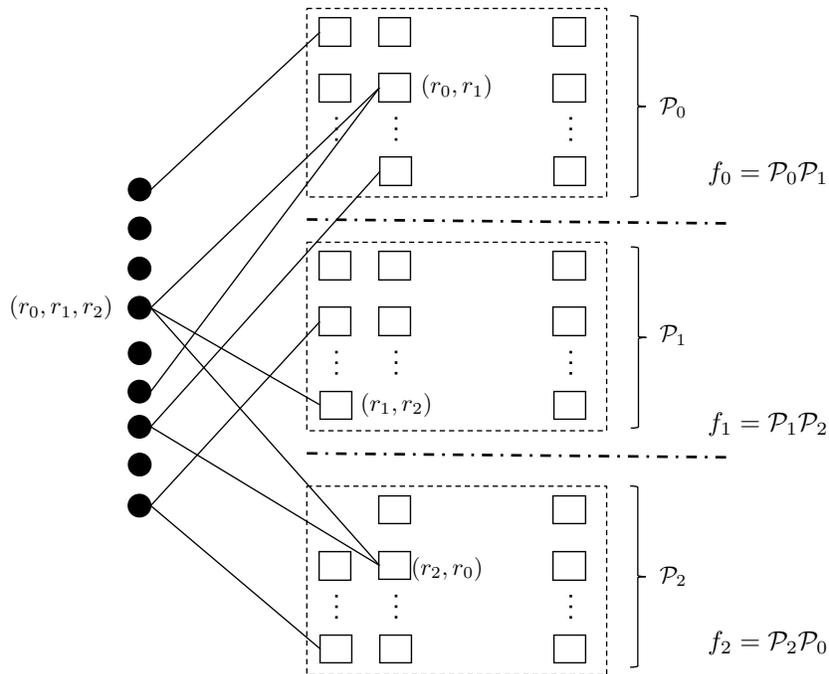


Fig. 7. A bi-partite graph with k variable nodes and $n_b = \sum_{i=0}^2 f_i$ check nodes, constructed using a balls-and-bins model. The check nodes in each of the 3 sets are arranged in a matrix format, e.g., the f_0 check nodes in the set 0 are arranged in \mathcal{P}_0 rows and \mathcal{P}_1 columns. A check node in row r_0 and column r_1 in the set 0, is indexed by a pair (r_0, r_1) and so on and so forth for all the other check nodes. Each variable node chooses a triplet (r_0, r_1, r_2) , where r_i is between 0 and $\mathcal{P}_i - 1$ uniformly at random. A 3-regular degree bi-partite graph is then constructed by connecting a variable node with a triplet (r_0, r_1, r_2) to check nodes (r_0, r_1) , (r_1, r_2) and (r_2, r_0) in the three sets of check nodes respectively.

regime ($k = O(n^\delta)$, $1/3 < \delta < 1$) the relation between the integers f_i 's is bit more involved. Hence, for ease of exposition, we adhere to the following approach:

- First, we describe the FFAST front-end construction and analyze performance of the FFAST decoding algorithm for a simple case of less-sparse regime of $\delta = 2/3$.
- Then, in Section VII-B, we provide a brief sketch of how to generalize the FFAST architecture of $\delta = 2/3$, to $\delta = 1 - 1/d$, for integer values of $d \geq 3$. This covers the range of values of $\delta = 2/3, 3/4, \dots$ etc.
- In Section VII-C, we show how to achieve the intermediate values of $1/3 < \delta < 1$.
- Finally, in Section VII-D, we use all the techniques learned in the previous sections to provide an explicit choice of parameters for the FFAST front-end architecture that achieves all the sparsity indices in the range $1/3 < \delta < 1$.

A. Less-sparse regime of $\delta = 2/3$

1) **FFAST front-end construction:** Consider $n = \prod_{i=0}^2 \mathcal{P}_i$, where the set $\{\mathcal{P}_i\}_{i=0}^2$ consists of approximately equal sized co-prime integers with each $\mathcal{P}_i = \mathbf{F} + O(1)$ and $\mathbf{F} = \sqrt{\eta k}$. This results in $\delta = 2/3$. Choose the integers $\{f_i\}_{i=0}^2$, such that $\{f_0, f_1, f_2\} = \{\mathcal{P}_0\mathcal{P}_1, \mathcal{P}_1\mathcal{P}_2, \mathcal{P}_2\mathcal{P}_0\}$. Then, we construct a $d = 3$ stage FFAST front-end architecture, where stage i has two delay-chains each with a sub-sampling period of n/f_i and f_i number of output samples.

2) **Performance analysis of the FFAST decoding algorithm:** In order to analyze the performance of the FFAST decoding algorithm, we follow a similar approach as in Section VI for the very-sparse regime of $0 < \delta \leq 1/3$. We first provide an ensemble of bi-partite graphs constructed using a balls-and-bins model. Then, we provide CRT based ensemble of bi-partite graphs, that are generated by the FFAST front-end of Section VII-A1. We show by construction these two ensembles are equivalent and analyze the performance of the FFAST peeling-decoder on a uniformly random graph from the balls-and-bins ensemble.

a) *Balls-and-Bins construction*: We construct a bi-partite graph with k variable nodes on the left and $n_b = \sum_{i=0}^2 f_i$, check nodes on the right (see Fig. 7) using balls-and-bins model as follows. Partition the n_b check nodes into 3 sets/stages containing f_0, f_1 and f_2 check nodes respectively. The check nodes in each of the 3 sets are arranged in a matrix format as shown in Fig. 7, e.g., f_0 check nodes in the set 0 are arranged as \mathcal{P}_0 rows and \mathcal{P}_1 columns. A check node in row r_0 and column r_1 in the set 0, is indexed by a pair (r_0, r_1) and so on and so forth. Each variable node uniformly randomly chooses a triplet (r_0, r_1, r_2) , where r_i is between 0 and $\mathcal{P}_i - 1$. The triplets are chosen with replacement and independently across all the k variable nodes. A 3-regular degree bi-partite graph with k variable nodes and n_b check nodes is then constructed by connecting a variable node with a triplet (r_0, r_1, r_2) to the check nodes $(r_0, r_1), (r_1, r_2)$ and (r_2, r_0) in the three sets on right respectively, as shown in Fig. 7.

b) *CRT based bi-partite graphs induced by the FFAST architecture*: Each variable node is associated with an integer v between 0 and $n - 1$ (location of the DFT coefficient). As a result of the subsampling and computing a smaller DFTs in the FFAST architecture (see Fig 1), a variable node with an index v is connected to the check nodes $(v)_{f_0}, (v)_{f_1}$ and $(v)_{f_2}$ in the 3 stages, in the resulting aliased bi-partite graph. The CRT implies that v is uniquely represented by a triplet (r_0, r_1, r_2) , where $r_i = (v)_{\mathcal{P}_i}$. Also, $((v)_{f_i})_{\mathcal{P}_i} = (v)_{\mathcal{P}_i} = r_i$, for all $i = 0, 1, 2$. Thus, the FFAST architecture induces a 3-regular degree bi-partite graph with k variable nodes and n_b check nodes, where a variable node with an associated triplet (r_0, r_1, r_2) is connected to the check nodes $(r_0, r_1), (r_1, r_2)$ and (r_2, r_0) in the three sets respectively. Further, a uniformly random model for the support v of the non-zero DFT coefficients, corresponds to choosing the triplet (r_0, r_1, r_2) uniformly at random. Thus, by construction the ensemble of bi-partite graphs generated by the FFAST front-end is equivalent to the balls-and-bins construction.

Following the outline of the proof of Theorem III.1 (provided in Section VI), we can show the following:

- 1) *Density evolution for the cycle-free case*: Assuming a local tree-like neighborhood derive a recursive equation (similar to equation 8) representing the expected evolution of the number of single-tons uncovered at each round for a ‘‘typical’’ graph from the ensemble.
- 2) *Convergence to the cycle-free case*: Using a Doob martingale show an equivalent of Lemma VI.2 for the less-sparse regime, where the number of check nodes in the 3 different stages f_0, f_1 and f_2 are not pairwise co-prime.
- 3) *Completing the decoding using the graph expansion property*: A random graph from the ensemble is a good expander with high probability. Hence, if the FFAST decoder successfully decodes all but a constant fraction of variable nodes, it decodes all the variable nodes.

The analysis of the first two items for the less-sparse regime is similar in spirit to the one in Section VI, and will be skipped here. However, the analysis of the third item will be described here as there are some key differences, mainly arising due to fact that integers in the set $\{f_i\}_{i=0}^2$ are not co-prime as in Section VI. For the very-sparse regime we have shown (in Lemma VI.5) that the bottleneck failure event is; not being able to decode *all* the DFT coefficients. Hence, in this section, we analyze this bottleneck failure event for the case of the less-sparse regime. In particular, we show that if the FFAST decoder has successfully decoded all but a small constant number of DFT coefficients, then it decodes all the DFT coefficients successfully with high probability.

c) *Decoding all the variable nodes using the expansion properties of the CRT construction*: Consider an alternative visualization of the bi-partite graph in Fig. 7, as shown in Fig. 8. A variable node associated with a triplet (r_0, r_1, r_2) is represented by a ball at the position (r_0, r_1, r_2) . The plane R_0 - R_1 corresponds to the check nodes in the stage f_0 , in a sense that all the variable nodes that have identical (r_0, r_1) but distinct r_2 are connected to the check node (r_0, r_1) and so on. Similarly the planes R_1 - R_2 and R_2 - R_0 correspond to the check nodes in stages f_1 and f_2 respectively. Thus, a variable node with co-ordinates (r_0, r_1, r_2) is connected to multi-ton check nodes, if and only if there exist variable nodes with co-ordinates $(r_0, r_1, r'_2), (r'_0, r_1, r_2)$ and (r_0, r'_1, r_2) (see Fig. 8), i.e., one neighbor in each axis. The FFAST decoder stops decoding if all the check nodes are multi-tons. Next, we find an upper bound on the probability of this ‘bad’ event.

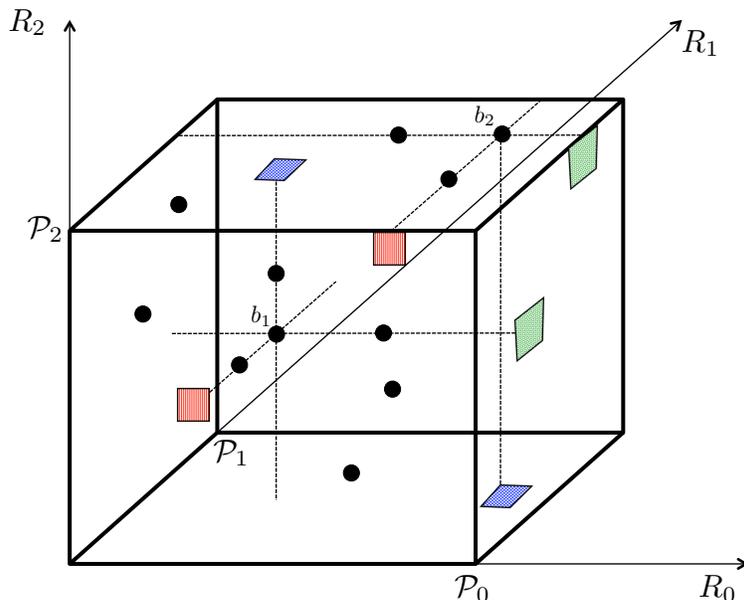


Fig. 8. A 3D visualization of the bi-partite graph in Fig. 7, that belongs to the CRT ensemble corresponding to $\delta = 2/3$. Recall that for $\delta = 2/3$, $n = \prod_{i=0}^2 \mathcal{P}_i$, where the set $\{\mathcal{P}_i\}_{i=0}^2$ consists of approximately equal sized co-prime integers with each $\mathcal{P}_i = \mathbf{F} + O(1)$ and $\mathbf{F} = \sqrt{\eta k}$. The integers $\{f_i\}_{i=0}^2$, are such that $\{f_0, f_1, f_2\} = \{\mathcal{P}_0 \mathcal{P}_1, \mathcal{P}_1 \mathcal{P}_2, \mathcal{P}_2 \mathcal{P}_0\}$. A variable node with an associated triplet (r_0, r_1, r_2) is represented by a ‘ball’ at the position (r_0, r_1, r_2) . The f_0 check nodes in the stage-1 of the bi-partite graph are represented by ‘blue’ (checked pattern) squares and likewise the ones in f_1 are ‘green’ (dotted pattern) and the check nodes in stage f_2 are ‘red’ (lines pattern). All the neighboring check nodes of a variable node, e.g., b_1 , are multi-ton iff there is at least one more variable node along each of the three directions R_0, R_1 and R_2 . The green (dotted pattern) and red (lines pattern) neighboring check nodes connected to the ball b_2 are multi-tons, while the blue (checked pattern) neighboring check node is a single-ton since there are no other variable nodes along the R_2 direction of b_2 .

Consider a set S of variable nodes such that $|S| = s$, where s is a small constant. Let E_S be an event that all the neighboring check nodes of all the variable nodes in the set S are multi-tons, i.e., the FFAST decoder fails to decode the set S . Also, let E be an event that there exists such a set. We first compute an upper bound on the probability of the event E_S , and then apply a union bound over all $\binom{k}{s}$ sets to get an upper bound on the probability of the event E .

Each variable node in the set S chooses an integer triplet (r_0, r_1, r_2) uniformly at random in a cube of size $\mathcal{P}_0 \times \mathcal{P}_1 \times \mathcal{P}_2$. Let p_{\max} denote the maximum number of distinct values taken by these s variable nodes on any co-ordinate. The FFAST decoder fails to decode the set S if and only if all the variable nodes have at least one neighbor along each of the 3 directions R_0, R_1, R_2 (see Fig. 8). This implies that $s \geq 4p_{\max}$. Also, $p_{\max} > 1$, i.e., $s \geq 8$, since by the CRT all the variable nodes s (with distinct associated integers) cannot have an identical triplet (r_0, r_1, r_2) . An upper bound on the probability of the event E_S is then obtained as follows:

$$\begin{aligned}
 Pr(E_S) &< \prod_{i=0}^2 \left(\frac{s}{4\mathcal{P}_i} \right)^s \binom{\mathcal{P}_i}{s/4} \\
 &\approx \left(\frac{s}{4\mathbf{F}} \right)^{3s} \binom{\mathbf{F}}{s/4}^3 \\
 &\stackrel{(a)}{<} \left(\frac{s}{4\mathbf{F}} \right)^{3s} \left(\frac{4\mathbf{F}e}{s} \right)^{3s/4} \\
 &= \left(\frac{se^{1/3}}{4\mathbf{F}} \right)^{9s/4}, \tag{12}
 \end{aligned}$$

where in (a) we used $\binom{p}{q} \leq (p^q/q!) \leq (pe/q)^q$. Then, using a union bound over all possible $\binom{k}{s}$ sets, we get:

$$\begin{aligned} \Pr(E) &< \Pr(E_S) \binom{k}{s} \\ &< \left(\frac{se^{1/3}}{4\mathbf{F}} \right)^{9s/4} \left(\frac{ke}{s} \right)^s \\ &= O(1/n_b), \end{aligned} \tag{13}$$

where in the last inequality, we used $s \geq 8$, $k = O(\mathbf{F}^2)$ and $n_b = O(\mathbf{F}^2)$.

Thus, the FFAST decoder decodes all the variable nodes with probability at least $1 - O(1/n_b)$.

B. FFAST front-end architecture for $\delta = 1 - 1/d$, where $d \geq 3$ is an integer

Consider $n = \prod_{i=0}^{d-1} \mathcal{P}_i$, where the set $\{\mathcal{P}_i\}_{i=0}^{d-1}$ consists of approximately equal sized co-prime integers with each $\mathcal{P}_i = \mathbf{F} + O(1)$, and $\mathbf{F}^{d-1} = \eta k$, for a constant η chosen as per Table IV. This results in $\delta = 1 - 1/d$. Choose the integers $\{f_i\}_{i=0}^{d-1}$, such that $f_i = \mathcal{P}_{(i)d} \mathcal{P}_{(i+1)d} \cdots \mathcal{P}_{(i+d-2)d}$, for $i = 0, \dots, d-1$. Then, we construct a d stage FFAST front-end architecture, where stage i has two delay-chains each with a sub-sampling period of n/f_i and f_i number of output samples. The performance of the back-end FFAST peeling-decoder, for these constructions, can be analyzed following the outline of Section VI and Section VII. For $\delta = 2/3$, the FFAST front-end architecture had $d = 3$ stages and as shown in Section VII-A, the bottleneck failure event was to decode a set S of size $|S| = 8 = 2^3$. For a general d stage FFAST front-end construction, using induction, one can show that the worst case failure event is when the FFAST decoder fails to decode a set of size $|S| = 2^d$. The probability of this event is upper bounded by $1/\mathbf{F}^{2^d - 2^d}$.

C. Achieving the intermediate values of δ

In this section, we show how to extend the scheme in Section VI, that was designed for $k = O(n^{1/3})$, to achieve a sparsity regime of $k = O(n^{(1+a)/(3+a)})$ for some $a > 0$. This extension technique can essentially be used in conjunction with any of the operating points described earlier. Thus covering the full range of sparsity indices in $0 < \delta < 1$.

Consider $n = \mathcal{P} \prod_{i=0}^2 \mathcal{P}_i$, where the set $\{\mathcal{P}_i\}_{i=0}^2$ consists of approximately equal sized co-prime integers with each $\mathcal{P}_i = \mathbf{F} + O(1)$ and $\mathcal{P} = \mathbf{F}^a$ for a constant $a > 0$. The parameter $\mathbf{F}^{1+a} = \eta k$, for a constant η chosen as per Table IV. This results in $\delta = (1+a)/(3+a)$. Further, choose the integers $\{f_0, f_1, f_2\} = \{\mathcal{P}\mathcal{P}_0, \mathcal{P}\mathcal{P}_1, \mathcal{P}\mathcal{P}_2\}$. Then, we construct a $d = 3$ stage FFAST front-end architecture, where stage i has two delay-chains each with a sub-sampling period of n/f_i and f_i number of output samples.

1) *Union of Disjoint problems:* The check nodes in each of the 3 sets are arranged so that a j^{th} check node in the set i , belongs to the row $(j)_{\mathcal{P}_i}$ and the column $\lfloor j/\mathcal{P} \rfloor$ (see Fig. 9).

A variable node with an associated integer v is uniquely represented by a quadruplet (r_0, r_1, r_2, q) , where $r_i = (v)_{\mathcal{P}_i}$, $i = 0, 1, 2$ and $q = \lfloor v/\mathcal{P} \rfloor$, and is connected to check node (r_i, q) , i.e., check node in row r_i and column q , of set i . The resulting bipartite graph is a union of \mathcal{P} disjoint bi-partite graphs, where each bi-partite subgraph behaves as an instance of the 3-stage perfect co-prime case discussed in Section VI. Then, using a union bound over these disjoint graphs one can compute the probability of the FFAST decoder successfully decoding all the variable nodes for asymptotic values of k, n .

D. FFAST architecture for achieving the sparsity index $1/3 < \delta \leq 1$

In this section, we provide a sketch of the parameters used for the FFAST construction to achieve all values of sparsity index δ in the less-sparse range, i.e., $1/3 < \delta < 1$. The proposed construction is built based on the design principles illustrated in Sections VII-A, VII-B and Section VII-C and is an achievable construction. There are many other construction parameters of FFAST that may achieve similar sparsity index with equal or better performance.

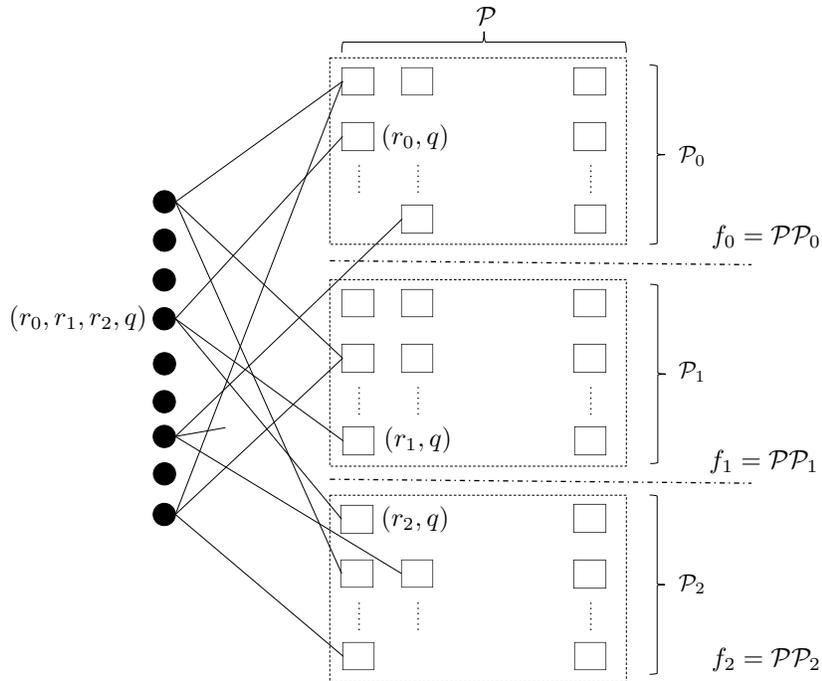


Fig. 9. Bi-partite aliasing graph resulting from processing an $n = \mathcal{P} \prod_{i=0}^2 \mathcal{P}_i$ length signal, using a $d = 3$ stage FFAST front-end. The set $\{\mathcal{P}_i\}_{i=0}^2$ consists of approximately equal sized co-prime integers with each $\mathcal{P}_i = \mathbf{F} + O(1)$ and $\mathcal{P} = \mathbf{F}^a$ for some constant $a > 0$. Further, the integers $\{f_0, f_1, f_2\} = \{\mathcal{P}\mathcal{P}_0, \mathcal{P}\mathcal{P}_1, \mathcal{P}\mathcal{P}_2\}$. The check nodes in each of the 3 sets are arranged so that a j^{th} check node in the set i , belongs to the row $(j)_{\mathcal{P}_i}$ and the column $\lfloor j/\mathcal{P} \rfloor$. A variable node with an associated integer v is uniquely represented by a quadruplet (r_0, r_1, r_2, q) , where $r_i = (v)_{\mathcal{P}_i}$, $i = 0, 1, 2$ and $q = \lfloor v/\mathcal{P} \rfloor$, and is connected to check node (r_i, q) , i.e., check node in row r_i and column q , of set i . The bipartite graph is a union of \mathcal{P} disjoint bi-partite graphs.

1) $1/3 < \delta < 0.73$: Consider $n = \mathcal{P} \prod_{i=0}^5 \mathcal{P}_i$, where the set $\{\mathcal{P}_i\}_{i=0}^5$ consists of approximately equal sized co-prime integers with each $\mathcal{P}_i = \mathbf{F} + O(1)$, and $\mathcal{P} = \mathbf{F}^a$, for $0 < a \leq 9$. Further, choose $\mathbf{F}^{2+a} = \eta k$, where constant η is given as per Table IV. This results in $1/3 < \delta < 0.73$. We construct a $d = 6$ stage FFAST construction such that stage i has two delay-chains each with a sub-sampling period of n/f_i and f_i number of output samples. The integers $\{f_i\}_{i=0}^5$ are such that $\{f_0, f_1, f_2\} = \{\mathcal{P}\mathcal{P}_0\mathcal{P}_1, \mathcal{P}\mathcal{P}_1\mathcal{P}_2, \mathcal{P}\mathcal{P}_2\mathcal{P}_0\}$ and $\{f_3, f_4, f_5\} = \{\mathcal{P}\mathcal{P}_3\mathcal{P}_4, \mathcal{P}\mathcal{P}_4\mathcal{P}_5, \mathcal{P}\mathcal{P}_5\mathcal{P}_3\}$. Note that for $a = 0$, this choice of parameters corresponds to two disjoint designs of $\delta = 2/3$ the FFAST construction described in Section VII-A. Using the analysis from Section VII-A, along with a union bound over \mathcal{P} sets of disjoint problems, as explained in Section VII-C, it is easy to verify that all δ between $1/3 < \delta < 0.73$, can be achieved with probability of error bounded above by $O(1/n_b)$, if $\eta \geq 0.2616$. Thus, the total number of samples m used by the FFAST algorithm, for this range of sparsity index δ , is $m = 2 \times 6 \times \eta k \leq 3.14k$.

2) $0.73 < \delta < 0.875$: For the sparsity index δ in the range $0.73 < \delta < 0.875$, one possible construction is to use a $d = 8$ stage FFAST construction consisting of two disjoint designs for $\delta = 4/3$, with 3 cyclically shifted primes at a time. For example, consider a set $\{\mathcal{P}_i\}_{i=0}^7$ of 8 co-prime integers of approximately equal size then, $\{f_0, f_1, f_2, f_3\} = \{\mathcal{P}\mathcal{P}_0\mathcal{P}_1\mathcal{P}_2, \mathcal{P}\mathcal{P}_1\mathcal{P}_2\mathcal{P}_3, \mathcal{P}\mathcal{P}_2\mathcal{P}_3\mathcal{P}_0, \mathcal{P}\mathcal{P}_3\mathcal{P}_0\mathcal{P}_1\}$ and $\{f_4, f_5, f_6, f_7\} = \{\mathcal{P}\mathcal{P}_4\mathcal{P}_5\mathcal{P}_6, \mathcal{P}\mathcal{P}_5\mathcal{P}_6\mathcal{P}_7, \mathcal{P}\mathcal{P}_6\mathcal{P}_7\mathcal{P}_4, \mathcal{P}\mathcal{P}_7\mathcal{P}_4\mathcal{P}_5\}$. For this construction, the parameter a in $\mathcal{P} = \mathbf{F}^a$, is set between $10.5 < a < 32$, to achieve an error performance that asymptotically approaches zero as $O(1/n_b^{0.9})$. From Table IV, for a $d = 8$ stage FFAST architecture $\eta = 0.2336$. Thus, the total sample complexity for this design is $m = 2d\eta k < 3.74k$.

3) $0.875 < \delta < 1$: The sparsity index δ in the range $0.875 < \delta < 1$, can be achieved by the combination of designs proposed in Section VII-B and Section VII-C for increasing (but constant) values of d as dictated by δ . For example, for $0.875 < \delta < 0.99$, a $d = 8$ stage FFAST front-end design as in Section VII-B

along with approach of Section VII-C achieves an error performance that asymptotically approaches zero with $m < 3.74k$ as shown in Fig. 2.

VIII. SAMPLE AND COMPUTATIONAL COMPLEXITY OF THE FFAST ALGORITHM

The FFAST algorithm performs the following operations in order to compute the n -length DFT of an n -length discrete-time signal \vec{x} (see Algorithm 1 in Appendix A)

- 1) **Sub-sampling:** A FFAST architecture (see Fig. 1 in Section I) with d stages, has d distinct subsampling patterns chosen as per the discussions in Sections VI and VII. These patterns are deterministic and are pre-computed. We assume the presence of random-access-memory, with unit cost per I/O operation, for reading the subsamples. For the very-sparse regime of $0 < \delta \leq 1/3$, as shown in Section VI-D, the FFAST architecture with $d = 3$ stages is sufficient to compute a k -sparse n -length DFT using $m = 2.44k$ samples. For the less-sparse regime of $1/3 < \delta \leq 0.99$, as shown in Section VII-D, we have three different FFAST architectures, with $d = 6$ and $d = 8$, for different range of δ . The total samples used for these regions are $m = 3.14k$ and $m = 3.74k$ respectively. In general for any fixed $0 < \delta < 1$, the sample complexity m can be as small as rk , where r is a constant that depends on the sparsity index δ .
- 2) **DFT:** The FFAST algorithm then computes $2d$ DFT's, each of length approximately equal to ηk , for some constant η depending on the number of stages d of the FFAST front-end. Using a standard FFT algorithm, e.g., prime-factor FFT or Winograd FFT algorithm [3], one can compute each of these DFT's in $O(k \log k)$ computations. Thus, the total computational cost of this step is $O(k \log k)$.
- 3) **Peeling-decoder over sparse graph codes:** It is well known [29], that the computational complexity of the FFAST peeling-decoder over sparse graph codes is linear in the dimension of the graph, i.e., $O(k)$.

Thus, the FFAST algorithm computes a k -sparse n -length DFT with $O(k)$ samples using no more than $O(k \log k)$ computations for all $0 < \delta < 1$.

IX. SIMULATION RESULTS

In this section we validate the empirical performance of the FFAST algorithm for computing the DFT of signals having a sparse Fourier spectrum. We contrast the observed empirical performance, in terms of sample complexity, computational complexity and probability of success, with the theoretical claims of Theorem III.1.

A. FFAST architecture and signal generation

- **Very sparse regime:** The input signal \vec{x} is of length $n = 511 \times 512 \times 513 \approx 134 \times 10^6$. The number of non-zero DFT coefficients k is varied from 400 to 1300, this corresponds to the very-sparse regime of $k = O(n^{1/3})$. We use an FFAST architecture with $d = 3$ stages and 2 delay-chains per stage. The uniform sampling periods for the 3 stages are 512×513 , 511×513 and 511×512 respectively. This results in the number of output samples, per delay-chain, for the three stages to be $f_0 = 511$, $f_1 = 512$ and $f_2 = 513$ respectively. Note that the number of samples in the three different stages, i.e., f_i 's, are pairwise co-prime. The total number of samples used by the FFAST algorithm for this simulation is⁹ $m < 2(f_0 + f_1 + f_2) = 3072$.
- **Less sparse regime:** The input signal \vec{x} is of length $n = 16 \times 17 \times 19 \times 21 \approx 0.1 \times 10^6$. The number of non-zero DFT coefficients k is varied from 5000 to 19000, this corresponds to the less-sparse regime of $n^{0.73} < k < n^{0.85}$. We use an FFAST architecture with $d = 4$ stages. The uniform sampling periods for the 4 stages are 21, 16, 17 and 19 respectively. This results in the number of output samples, per delay-chain, for the four stages to be $f_0 = 16 \times 17 \times 19 = 5168$, $f_1 = 17 \times 19 \times 21 = 6783$,

⁹The samples used by the different sub-streams in the three different stages overlap, e.g., $x[0]$ is common to all the zero delay sub-streams in each stage. Hence, $m < 2(f_0 + f_1 + f_2)$ and not equal.

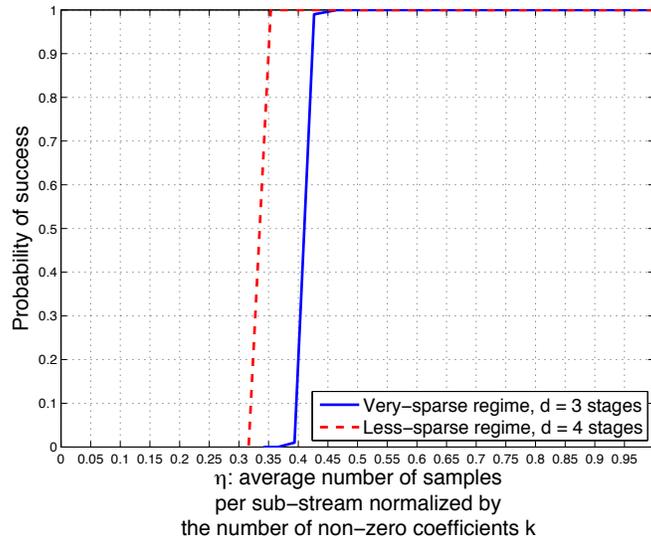


Fig. 10. The probability of success of the FFAST algorithm as a function of η , the average number of samples per delay-chain, normalized by the number of non-zero DFT coefficients. The plot is obtained for two different sparsity regimes: 1) Very-sparse regime: $k = O(n^{1/3})$. For this regime, a $d = 3$ stage FFAST architecture is chosen. The number of samples per sub-stream in each of the 3 stages are perfectly co-prime: $f_0 = 511, f_1 = 512$ and $f_2 = 513$ respectively, and 2) Less-sparse regime: $n^{0.73} < k < n^{0.85}$. For this regime, a $d = 4$ stage FFAST architecture is chosen. The number of samples per sub-stream in each of the 4 stages are *not* co-prime but have “cyclically-shifted” overlapping co-prime factors: $f_0 = 16 \times 17 \times 19 = 5168, f_1 = 17 \times 19 \times 21 = 6783, f_2 = 19 \times 21 \times 16 = 6384$ and $f_3 = 21 \times 16 \times 17 = 5712$ respectively. Each point on the plot is obtained by averaging over 10000 runs. The ambient signal dimension n and the number of samples m are fixed, while the number of non-zero coefficients k is varied to get different values of η . We note that the FFAST algorithm exhibits a threshold behavior with $\eta_1 = 0.427$ being the threshold for the very-sparse regime, and $\eta_2 = 0.354$ for the less-sparse regime respectively. From Table IV in Section VI-C1, we see that the optimal threshold values are $\eta_1^* = 0.4073$ and $\eta_2^* = 0.3237$, which are in close agreement with our simulation results.

$f_2 = 19 \times 21 \times 16 = 6384$ and $f_3 = 21 \times 16 \times 17 = 5712$ respectively. Note that the number of samples in the four stages are composed of “cyclically-shifted” co-prime numbers and are not pairwise co-prime. The total number of samples used by the FFAST algorithm for this simulation is $m < 2(f_0 + f_1 + f_2 + f_3) = 48094$.

- For each run, an n -dimensional k -sparse signal \vec{X} is generated with non-zero values $X_i \in \{\pm 10\}$ at uniformly random positions in $\{0, 1, \dots, n-1\}$. The time domain signal \vec{x} is then generated from \vec{X} using an IDFT operation. This discrete-time signal \vec{x} is then given as an input to FFAST.
- Each sample point in the Fig. 10, is generated by averaging over 10000 runs.
- Decoding is successful if all the DFT coefficients are recovered perfectly.

B. Observations and conclusions

We observe the following aspects of the simulations in detail and contrast it with the claims of Theorem III.1

Density Evolution: The density evolution recursion (8) in Section VI-C1, implies a threshold behavior: if the average number of samples per delay-chain normalized by k , i.e., η , is above a certain threshold (as specified in Section VI-C1 Table IV), then the recursion (8) converges to 0 as $\ell \rightarrow \infty$ otherwise p_ℓ is strictly bounded away from 0. In Fig. 10, we plot the probability of success of the FFAST algorithm averaged over 10000 independent runs, as a function of η , i.e., varying k for a fixed number of samples m . We note that the FFAST algorithm also exhibits a threshold behavior with $\eta_1 = 0.427$, being the threshold for the very-sparse regime with $d = 3$, and $\eta_2 = 0.354$ for the less-sparse regime with $d = 4$ respectively. From Table IV in Section VI-C1, we see that the optimal threshold values are $\eta_1^* = 0.4073$ and $\eta_2^* = 0.3237$, which are in close agreement with our simulation results of Fig. 10.

Regime	Signal length n	Sparsity k	δ	stages d	$m \approx 2d\eta k$		ℓ	failures
					m	η		
Very-sparse	511*512*513 $\approx 134 \times 10^6$	900	0.363	3	3072	0.569	6	1/10 ⁴
		1000	0.369		3072	0.512	9	0/10 ⁴
		1100	0.374		3072	0.465	13	1/10 ⁴
		1200	0.378		3072	0.427	18	99/10 ⁴
Less-sparse	16*17*19*21 $\approx 0.1 \times 10^6$	13000	0.81	4	48094	0.462	6	0/10 ⁴
		15000	0.83		48094	0.401	8	0/10 ⁴
		17000	0.84		48094	0.354	13	2/10 ⁴
		19000	0.85		48094	0.316	29	10 ⁴ /10 ⁴

TABLE V

PERFORMANCE OF THE FFAST ALGORITHM FOR TWO DIFFERENT SPARSITY REGIMES: 1) VERY-SPARSE REGIME: $k = O(n^{1/3})$. FOR THIS REGIME, A $d = 3$ STAGE FFAST ARCHITECTURE IS CHOSEN. THE NUMBER OF SAMPLES PER SUB-STREAM IN EACH OF THE 3 STAGES ARE PERFECTLY CO-PRIME: $f_0 = 511$, $f_1 = 512$ AND $f_2 = 513$ RESPECTIVELY, AND 2) LESS-SPARSE REGIME: $n^{0.73} < k < n^{0.85}$. FOR THIS REGIME, A $d = 4$ STAGE FFAST ARCHITECTURE IS CHOSEN. THE NUMBER OF SAMPLES PER SUB-STREAM IN EACH OF THE 4 STAGES ARE *NOT* CO-PRIME BUT HAVE ‘‘CYCLICALLY-SHIFTED’’ OVERLAPPING CO-PRIME FACTORS: $f_0 = 16 \times 17 \times 19 = 5168$, $f_1 = 17 \times 19 \times 21 = 6783$, $f_2 = 19 \times 21 \times 16 = 6384$ AND $f_3 = 21 \times 16 \times 17 = 5712$ RESPECTIVELY. WE NOTE THAT WHEN $\eta_1 \geq 0.427$ AND $\eta_2 \geq 0.354$, FOR THE VERY-SPARSE AND THE LESS-SPARSE REGIMES RESPECTIVELY, THE FFAST ALGORITHM SUCCESSFULLY COMPUTES ALL THE NON-ZERO DFT COEFFICIENTS FOR ALMOST ALL THE RUNS. FURTHER, IN ONE OR TWO INSTANCES WHEN IT FAILED TO RECOVER ALL THE NON-ZERO DFT COEFFICIENTS, IT HAS RECOVERED ALL BUT 8 (FOR $d = 3$) OR 16 (FOR $d = 4$) NON-ZERO DFT COEFFICIENTS. THIS VALIDATES OUR THEORETICAL FINDINGS. FROM TABLE IV IN SECTION VI-C1, WE SEE THAT THE OPTIMAL THRESHOLD VALUES FOR THE VERY-SPARSE AND LESS-SPARSE REGIMES ARE $\eta_1^* = 0.4073$ AND $\eta_2^* = 0.3237$ RESP., WHICH ARE IN CLOSE AGREEMENT WITH THE SIMULATION RESULTS. THE TABLE ALSO SHOWS THAT THE AVERAGE NUMBER OF ITERATIONS ℓ , REQUIRED FOR THE FFAST ALGORITHM TO SUCCESSFULLY COMPUTE THE DFT FOR BOTH THE REGIMES, ARE QUITE MODEST.

Iterations: In the theoretical analysis of Section VI-C2, we have shown that the FFAST algorithm, if successful, decodes all the DFT coefficients in ℓ iterations, where, ℓ is a large but a fixed constant. In Table V, we observe that when successful, the FFAST peeling-decoder completes decoding in few (≤ 13 in this case) number of iterations.

Probability of success: The empirical probability of success of the FFAST algorithm for the very-sparse as well as the less-sparse regimes is shown in Table V. We observe, as long as the parameter η is above the minimum threshold η^* dictated by Table IV, in Section VI-C1, the FFAST algorithm indeed computes all the non-zero DFT coefficients successfully with high probability. Further, in one or two instances when it failed to recover all the non-zero DFT coefficients, it has recovered all but 8 (for $d = 3$) or 16 (for $d = 4$) non-zero DFT coefficients. Thus, confirming our theoretical analysis of the worst case error event, of being trapped in a cycle of size 2^d .

X. CONCLUSION

In this paper, we addressed the problem of computing an n -length DFT of signals that have k -sparse Fourier transform, where $k \ll n$. We proposed a novel FFAST algorithm that cleverly exploits filterless subsampling operation to induce aliasing artifacts, similar to parity-check constraints of good erasure-correcting sparse-graph codes, on the spectral coefficients. Then, we formally connected the problem of computing sparse DFT to that of decoding of appropriate sparse-graph codes. This connection was further exploited to design a sub-linear complexity FFAST peeling-style back-end decoder. Further, we analyzed the performance of the FFAST algorithm, using well known density evolution techniques from coding theory, to show that our proposed algorithm computes the k -sparse n -length DFT using only $O(k)$ samples in $O(k \log k)$ arithmetic operations, with high probability. The constants in the big Oh notation for both sample and computational cost are small, e.g., when $\delta < 0.99$, which essentially covers almost all practical cases of interest, the sample cost is less than $4k$. We also provide simulation results, that are in tight agreement with our theoretical findings.

Although, the focus of this paper is on signals having an exactly-sparse DFT, the key insights derived from analyzing the exactly sparse signal model apply more broadly to the noisy setting, i.e., where the observations are further corrupted by noise [4]. In particular, in [4], we show that the robustness against sample noise is achieved by an elegant modification of the noiseless FFAST framework. Specifically, the noiseless FFAST framework shown in Fig. 1 has multiple sub-sampling stages, where each stage has 2 delay-chains. In contrast, for the case of noise robust FFAST framework, we use $O(\log n)$ number of delay-chains per sub-sampling stage, further the input signal to each delay-chain is circularly shifted by a random amount prior to sub-sampling, i.e., random delays. In [4], we show that a random choice of the circular shifts endows the effective “measurement matrix” with a good mutual incoherence and Restricted-Isometry-Property [30], thus resulting in a stable recovery.

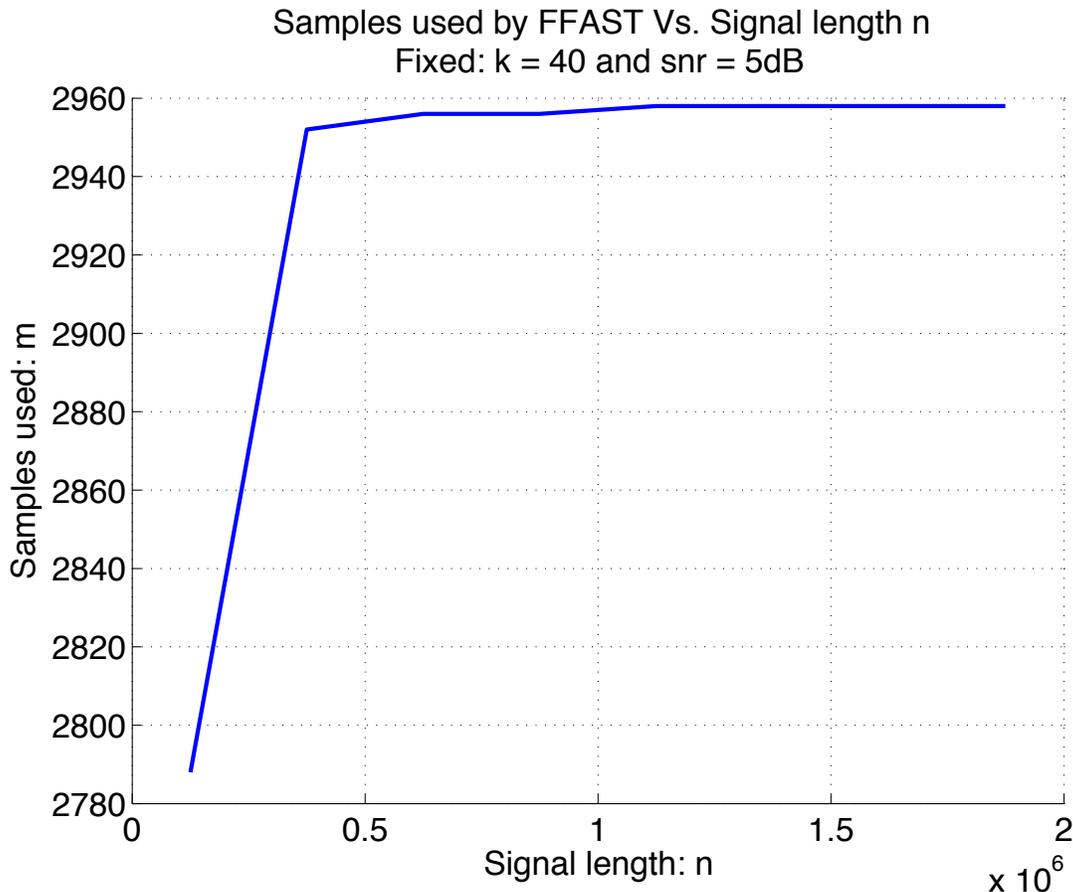


Fig. 11. The plot shows the scaling of the number of samples m required by the noise robust FFAST algorithm to successfully reconstruct a $k = 40$ sparse DFT \vec{X} , for increasing signal length n . For a fixed support recovery probability of 0.99, sparsity k and SNR of 5dB, we note that m scales logarithmically with increasing n .

As an example, we provide here an empirical evaluation of the scaling of the number of samples m , required by a noise robust FFAST algorithm [4], to successfully compute the DFT \vec{X} from noise-corrupted samples, as a function of the signal length n . A noise-corrupted signal $\vec{y} = \vec{x} + \vec{z}$, where \vec{z} is Gaussian, of length n is generated such that the effective signal-to-noise ratio is 5dB and the DFT of \vec{x} has $k = 40$ non-zero coefficients. The signal n is varied from $n = 49 \cdot 50 \cdot 51 \approx 0.1$ million to $n = 49 \cdot 50 \cdot 51 \cdot 15 \approx 1.87$ million. The noise-corrupted samples \vec{y} are processed using a $d = 3$ stage FFAST architecture with D delay-chains per stage. As the signal length n increases the number of delays D per stage are increases to achieve at least 0.99 probability of successful support recovery. From the plot in Fig. 11, we note that m scales logarithmically with increasing n . For further details about this simulation see [4].

The FFAST framework in this paper, has inspired the works of [31] and [32], where the authors have proposed novel algorithms for computing high dimensional sparse “Walsh Hadamard Transform” (WHT), for the noiseless and noisy observed samples respectively. The framework has also been successfully applied for blind acquisition of signals with sparse spectra using sub-Nyquist sampling [33] and to a variant of a phase-retrieval-problem as in [34]. We are hopeful that many more applications will follow.

APPENDIX

A. FFAST Algorithm Pseudocode

Algorithm 1 FFAST Algorithm

1: *Inputs:*

- A discrete time signal \vec{x} of length n , whose n -length DFT \vec{X} has at most k non-zero coefficients.
- The subsampling parameters of the FFAST architecture (see Fig. 1): 1) number of stages d . and 2) number of samples per sub-stream in each of the d stages $\mathcal{F} = \{f_0, f_1, \dots, f_{d-1}\}$, chosen as per discussion in Sections VI and VII.

2: *Output:* An estimate of the k -sparse n -length DFT \vec{X} .

3: *FFAST Decoding:* Set the initial estimate of the n -length DFT $\vec{X} = 0$. Let ℓ denote the number of iterations performed by the FFAST decoder.

4: **for** each iteration **do**

5: **for** each stage $i = 0$ to $d - 1$ **do**

6: **for** each bin $j = 0$ to $f_i - 1$ **do**

7: **if** $\|\vec{y}_{b,i,j}\|^2 == 0$ **then**

8: bin j of stage i is a zero-ton.

9: **else**

10: (single-ton, v_p, p) = *Singleton-Estimator* ($\vec{y}_{b,i,j}$).

11: **if** single-ton = ‘true’ **then**

12: *Peeling:* $\vec{y}_{b,s,q} = \vec{y}_{b,s,q} - v_p \begin{pmatrix} 1 \\ e^{i2\pi p/n} \end{pmatrix}$, for all stages s and bins $q \equiv p \pmod{f_s}$.

13: Set $X[p] = v_p$.

14: **else**

15: bin j of stage i is a *multi-ton*.

16: **end if**

17: **end if**

18: **end for**

19: **end for**

20: **end for**

Algorithm 2 Singleton-Estimator

1: *Input:* The bin observation $\vec{y}_{b,i,j}$.

2: *Outputs:* 1) A boolean flag ‘single-ton’, 2) Estimated value v_p of the non-zero DFT coefficient at position p .

3: *Singleton-Estimator:* Set the single-ton = ‘false’.

4: **if** $(n/2\pi)\angle y_{b,i,j}[1]y_{b,i,j}[0]^\dagger \in \{0, 1, \dots, (n-1)\}$ **then**

5: single-ton = ‘true’.

6: $v_p = y_{b,i,j}[0]$.

7: $p = (n/2\pi)\angle y_{b,i,j}[1]y_{b,i,j}[0]^\dagger$.

8: **end if**

B. Ratio-test for a multi-ton bin

Consider a multi-ton-bin ℓ with $L - 1$ non-zero components where $L > 2$. Let i_0, i_1, \dots, i_{L-2} be the indices of the non-zero components of \vec{X} contributing to the observation of the multi-ton bin ℓ . Then, we have

$$\vec{y}_{b,\ell} = \begin{bmatrix} \vec{v}_{i_0} & \vec{v}_{i_1} & \cdots & \vec{v}_{i_{L-3}} & \vec{v}_{i_{L-2}} \end{bmatrix} \begin{bmatrix} X[i_0] \\ X[i_1] \\ \vdots \\ X[i_{L-3}] \\ X[i_{L-2}] \end{bmatrix}, \quad (14)$$

where vector $\vec{v}_{i_0} = \begin{pmatrix} 1 \\ e^{i2\pi i_0/n} \end{pmatrix}$ consists of the first two entries of the i_0^{th} column of the $n \times n$ IDFT matrix. The multi-ton bin ℓ will be identified as a single-ton bin with location j and value $X[j]$ for some $0 \leq j < n$ iff

$$\begin{aligned} \vec{y}_{b,\ell} &= \vec{v}_j X[j] \\ \text{i.e., } \begin{bmatrix} \vec{v}_{i_0} & \vec{v}_{i_1} & \cdots & \vec{v}_{i_{L-3}} \end{bmatrix} \begin{bmatrix} X[i_0] \\ X[i_1] \\ \vdots \\ X[i_{L-3}] \end{bmatrix} &= \begin{bmatrix} \vec{v}_{i_{L-2}} & \vec{v}_j \end{bmatrix} \begin{bmatrix} -X[i_{L-2}] \\ X[j] \end{bmatrix} \\ \vec{u} &= \begin{bmatrix} \vec{v}_{i_{L-2}} & \vec{v}_j \end{bmatrix} \begin{bmatrix} -X[i_{L-2}] \\ X[j] \end{bmatrix}. \end{aligned} \quad (15)$$

where $\vec{u} \in \mathbb{C}^2$ is some resultant vector. The matrix consisting of the first two rows of an IDFT matrix is a Vandermonde matrix and hence equation (15) has a unique solution. The complex coefficient $X[j]$ is free to choose but $X[i_{L-2}]$ is drawn from a continuous distribution. As a result probability of satisfying equation (15) is essentially zero even after applying union bound over all j . Hence a multi-ton bin is identified correctly almost surely.

C. Check node edge degree-distribution polynomial of the graphs in the balls-and-bins ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$

The check node edge-degree distribution polynomial of a bi-partite graph is defined as $\rho(\alpha) = \sum_{i=1}^{\infty} \rho_i \alpha^{i-1}$, where ρ_i denotes the probability of an edge (or fraction of edges) of the graph is connected to a check node of degree i . Recall that in the randomized construction based on a balls-and-bins model described in Section VI-A, every variable node is connected to d check nodes, one uniformly random check node in each of the d sets. Thus, the number of edges connected to a check node, in the subset with f_0 check nodes, is a binomial $B(1/f_0, k)$ random variable. For large k and $f_0 = \eta k + O(1)$, the binomial distribution $B(1/f_0, k)$ is well approximated by a Poisson random variable with mean $1/\eta$. Thus,

$$Pr(\text{check node in set 0 has edge degree} = i) \approx \frac{(1/\eta)^i e^{-1/\eta}}{i!}. \quad (16)$$

Let $\rho_{0,i}$ be the fraction of the edges, that are connected to a check node of degree i in set 0. Then, we have,

$$\begin{aligned} \rho_{0,i} &= \frac{if_0}{k} Pr(\text{check node has edge degree} = i) \\ &\stackrel{(a)}{\approx} \frac{if_0 (1/\eta)^i e^{-1/\eta}}{k i!} \\ &\stackrel{(b)}{\approx} \frac{(1/\eta)^{i-1} e^{-1/\eta}}{(i-1)!}, \end{aligned} \quad (17)$$

where (a) follows from Poisson approximation of the Binomial random variable and (b) from $f_0 = \eta k + O(1)$. Since, $f_i = \eta k + O(1)$ for all $i = 0, 1, \dots, d-1$. We have,

$$\rho_i \approx \frac{(1/\eta)^{i-1} e^{-1/\eta}}{(i-1)!} \quad (18)$$

and

$$\rho(\alpha) = e^{-(1-\alpha)/\eta} \quad (19)$$

D. Proof of Lemma VI.4

Proof: We provide a proof using a contradiction argument. If possible let S be the set of the variable nodes that the FFAST peeling-decoder fails to decode. We have $|S| \leq \alpha k$. Without loss of generality let $|N_1(S)| \geq |N_i(S)|$, $\forall i \in \{0, \dots, d-1\}$. Then, by the hypothesis of the Theorem $|N_1(S)| > |S|/2$.

Note that the FFAST peeling-decoder fails to decode the set S if and only if there are no more singleton check nodes in the neighborhood of S and in particular in $N_1(S)$. For all the check nodes in $N_1(S)$ to be a multi-ton, the number of edges connecting to the check nodes in the set $N_1(S)$ have to be at least $2|N_1(S)| > |S|$. This is a contradiction since there are only $|S|$ edges going from set S to $N_1(S)$ by construction. ■

E. Proof of Lemma VI.5

Proof: Consider a set S of variable (left) nodes in a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, where $|\mathcal{F}| = d \geq 3$. Let $N_i(S)$ be the right neighborhood of the set S in the i^{th} subset of check nodes, for $i = 0, 1, \dots, d-1$. Also, let E_S denote the event that the all the d right neighborhoods of S are of size $|S|/2$ or less, i.e., $\max\{|N_i(S)|\}_{i=0}^{d-1} \leq |S|/2$. First, we compute an upper bound on the probability of the event E_S as follows:

$$\begin{aligned} Pr(E_S) &< \prod_{i=0}^{d-1} \binom{|S|}{2f_i}^{|S|} \binom{f_i}{|S|/2} \\ &\stackrel{(a)}{\approx} \left(\frac{|S|}{2\mathbf{F}}\right)^{d|S|} \binom{\mathbf{F}}{|S|/2}^d \\ &< \left(\frac{|S|}{2\mathbf{F}}\right)^{d|S|} \left(\frac{2\mathbf{F}e}{|S|}\right)^{d|S|/2} \\ &= \left(\frac{|S|e}{2\mathbf{F}}\right)^{d|S|/2} \end{aligned} \quad (20)$$

where the approximation (a) uses $f_i = \mathbf{F} + O(1)$ for all $i = 0, \dots, d-1$. Consider an event E , that there exists some set of variable nodes of size $|S|$, whose all the d right neighborhoods are of size $|S|/2$ or less. Applying a union bound we get,

$$\begin{aligned} Pr(E) &< Pr(E_S) \binom{k}{|S|} \\ &< \left(\frac{|S|e}{2\mathbf{F}}\right)^{d|S|/2} \left(\frac{ke}{|S|}\right)^{|S|} \\ &\stackrel{(b)}{<} \left[\left(\frac{|S|}{\mathbf{F}}\right)^{d-2} \left(\frac{e}{2}\right)^d \left(\frac{e}{\eta}\right)^2 \right]^{|S|/2} \\ &< O\left(\left(\frac{|S|}{n_b}\right)^{|S|/2}\right) \end{aligned} \quad (21)$$

where in (b) we used $\mathbf{F} = \eta k$ and in the last inequality we have used $d \geq 3$ and $n_b = O(\mathbf{F})$. Then, specializing the bound in (21) for $|S| = \alpha k$, for small enough $\alpha > 0$, and $|S| = o(k)$ we get,

- For $|S| = \alpha k$, for small enough $\alpha > 0$:

$$\Pr(E) < e^{-\epsilon k \log(n_b/k)}, \quad \text{for some } \epsilon > 0 \quad (22)$$

- For $|S| = o(k)$:

$$\Pr(E) < O(1/n_b) \quad (23)$$

■

F. Proof of Lemma VI.2

Proof: a) [Expected behavior] Consider decoding on a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$. Let Z_i , $i = 0, \dots, kd-1$, be an indicator random variable that takes value 1 if the edge \vec{e}_i is not decoded after ℓ iterations of the FFAST peeling-decoder and 0 otherwise. Then, by symmetry $\mathbb{E}[Z] = kd\mathbb{E}[Z_1]$. Next, we compute $\mathbb{E}[Z_1]$ as,

$$\begin{aligned} \mathbb{E}[Z_1] &= \mathbb{E}[Z_1 \mid \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}] \Pr(\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}) \\ &\quad + \mathbb{E}[Z_1 \mid \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}] \Pr(\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}) \\ &\leq \mathbb{E}[Z_1 \mid \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}] + \Pr(\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}). \end{aligned}$$

In Appendix G, we show that $\Pr(\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}) \leq O((\log k)^{2\ell}/k)$. Also we have $\mathbb{E}[Z_1 \mid \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}] = p_\ell$ by definition. Thus,

$$\mathbb{E}[Z] < 2kdp_\ell. \quad (24)$$

b) [Concentration] In this part, we want to show that the number of edges Z , that are not decoded at the end of ℓ iterations of the FFAST peeling-decoder, is highly concentrated around $\mathbb{E}[Z]$. We use the standard martingale argument, of exposing an edge of the graph at a time, along with the Azuma's inequality as in [27], with slight modification to account for the irregular edge degree of the right nodes. The bi-partite graphs in the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, unlike the ones in [27], have left nodes with d -regular edge degree, while the right nodes edge degree is a Poisson random variable with a constant mean. In particular, let $Y_i = \mathbb{E}[Z \mid \vec{e}_i]$ be the expected value of Z after exposing i edges. Then, $Y_0 = \mathbb{E}[Z]$ and $Y_{kd} = Z$, and the sequence Y_i forms a Doob martingale. If $|Y_i - Y_{i-1}| < c_i$ then, using Azuma's inequality we have for all reals t ,

$$\Pr(|Z - \mathbb{E}[Z]| > t) \leq 2 \exp\left(\frac{-t^2}{2 \sum_{i=1}^{kd} c_i^2}\right). \quad (25)$$

In [27], the authors have shown that for bi-partite graphs with edge degrees d_v for left nodes and d_c for right nodes, $c_i < 8(d_c d_v)^\ell$, when the peeling-decoder performs message passing over ℓ iterations. The graphs considered in this paper have constant left degree, but the edge degree of the right nodes is a *Poisson random variable* with a constant mean, hence we probabilistically bound the value of d_c . Let B be an event that all the right nodes of a random graph from the considered ensemble have edge degrees less than or equal to $O(k^{1/(2\ell+0.5)})$. Then, using a bound on the Poisson distribution and a union bound over all the $O(k)$ check nodes, we get,

$$\Pr(\bar{B}) < O(ke^{-\beta_1 k^{\frac{1}{2\ell+0.5}}}). \quad (26)$$

Conditioned on the event B , one can upper bound c_i by $O(k^{\ell/(2\ell+0.5)})$. Now putting (25) and (26) together,

we have,

$$\begin{aligned}
Pr(|Z - \mathbb{E}[Z]| > kd\epsilon_1) &\leq Pr(|Z - \mathbb{E}[Z]| > kd\epsilon_1|B) + Pr(\bar{B}) \\
&\leq 2 \exp\left(\frac{-k^2 d^2 \epsilon_1^2}{2 \sum_{i=1}^{kd} C_i^2}\right) + O(ke^{-\beta_1 k^{\frac{1}{2\ell+0.5}}}) \\
&\leq e^{-\beta_1 \epsilon_1^2 k^{1/(4\ell+1)}}.
\end{aligned} \tag{27}$$

■

G. Probability of Tree-like Neighborhood

Consider an edge \vec{e} in a randomly chosen graph $\mathcal{G} \in \mathcal{C}_1^k(\mathcal{F}, n_b)$. Next, we show that a $2\ell^*$ depth neighborhood $\mathcal{N}_{\vec{e}}^{2\ell^*}$ of the edge \vec{e} is tree-like with high probability, for any fixed ℓ^* . Towards that end, we first assume that the neighborhood $\mathcal{N}_{\vec{e}}^{2\ell}$ of depth 2ℓ , where $\ell < \ell^*$, is tree-like and show that it remains to be tree-like when extended to depth $2\ell + 1$ with probability approaching 1. Let $C_{\ell,i}$ be the number of check nodes, from set i , and M_ℓ be the number of variable nodes, present in $\mathcal{N}_{\vec{e}}^{2\ell}$. Also assume that t more edges from the leaf variable nodes in $\mathcal{N}_{\vec{e}}^{2\ell}$ to the check nodes in set i at depth $2\ell + 1$ are revealed without creating a loop. Then, the probability that the next revealed edge from a leaf variable node to a check node (say in set i) does not create a loop is $\frac{f_i - C_{\ell,i} - t}{f_i - C_{\ell,i}} \geq 1 - \frac{C_{\ell^*,i}}{f_i - C_{\ell^*,i}}$. Thus, the probability that $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like, given $\mathcal{N}_{\vec{e}}^{2\ell}$ is tree-like, is lower bounded by $\min_i (1 - \frac{C_{\ell^*,i}}{f_i - C_{\ell^*,i}})^{C_{\ell+1,i} - C_{\ell,i}}$. Similarly assume that $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like and s more edges from check nodes to the variable nodes at depth $2\ell + 2$ are revealed without creating a loop. Conditioned on the event that a check node has an outgoing edge it has equal chance of connecting to any of the edges of the variable nodes that are not yet connected to any check node. Thus, the probability of revealing a loop creating edge from a check node to a variable node at depth $2\ell + 2$ is upper bounded by, $(1 - \frac{(k - M_{\ell+1} - s)d}{kd - M_{\ell+1}d - s}) \leq \frac{M_{\ell^*}}{(k - M_{\ell^*})}$. Thus, the probability that $\mathcal{N}_{\vec{e}}^{2\ell+2}$ is tree-like given $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like is lower bounded by $(1 - \frac{M_{\ell^*}}{(k - M_{\ell^*})})^{M_{\ell+1} - M_\ell}$.

It now follows that the probability that $\mathcal{N}_{\vec{e}}^{2\ell^*}$ is tree-like is lower bounded by

$$\min_i \left(1 - \frac{M_{\ell^*}}{(k - M_{\ell^*})}\right)^{M_{\ell^*}} \left(1 - \frac{C_{\ell^*,i}}{f_i - C_{\ell^*,i}}\right)^{C_{\ell^*,i}}$$

Hence, for k sufficiently large and fixed ℓ^* ,

$$Pr(\mathcal{N}_{\vec{e}}^{2\ell^*} \text{ not tree-like}) \leq \max_i \frac{M_{\ell^*}^2}{k} + \frac{C_{\ell^*,i}^2}{f_i}$$

Recall, the number of edges connected to a check node in set i is a binomial $B(1/f_i, k)$ random variable, which asymptotically can be approximated by a Poisson random variable with mean $1/\eta$, where $f_i = \eta k + O(1)$, for a constant η . Hence, for any fixed value of ℓ^* , $\max_i \{C_{\ell^*,i}\}$ and M_{ℓ^*} are upper bounded by $O((\log k)^{\ell^*})$ with probability $1/k^c$ for any constant c . Thus,

$$Pr(\mathcal{N}_{\vec{e}}^{2\ell^*} \text{ not tree-like}) \leq O\left(\frac{(\log k)^{2\ell^*}}{k}\right)$$

REFERENCES

- [1] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.
- [2] M. Luby, "Digital fountain, inc. luby@digitalfountain.com," 2002.
- [3] R. Blahut, *Fast algorithms for digital signal processing*, 1985.

- [4] S. Pawar and K. Ramchandran, “A robust r-ffast framework for computing a k -sparse n -length dft in $\mathcal{O}(k \log n)$ sample complexity using sparse-graph codes,” in *Information Theory Proceedings (ISIT), 2014 IEEE International Symposium on*. IEEE, 2014.
- [5] C. Temperton, “Self-sorting mixed-radix fast fourier transforms,” *Journal of computational physics*, vol. 52, no. 1, pp. 1–23, 1983.
- [6] D. Donoho, “Compressed sensing,” *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [7] E. Candes and T. Tao, “Near-optimal signal recovery from random projections: Universal encoding strategies?” *Information Theory, IEEE Transactions on*, vol. 52, no. 12, pp. 5406–5425, 2006.
- [8] R. Prony, “Essai experimental–,-,” *J. de l’Ecole Polytechnique*, 1795.
- [9] V. F. Pisarenko, “The retrieval of harmonics from a covariance function,” *Geophysical Journal of the Royal Astronomical Society*, vol. 33, no. 3, pp. 347–366, 1973.
- [10] R. Schmidt, “Multiple emitter location and signal parameter estimation,” *Antennas and Propagation, IEEE Transactions on*, 1986.
- [11] R. Roy and T. Kailath, “Esprit-estimation of signal parameters via rotational invariance techniques,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 7, pp. 984–995, 1989.
- [12] E. Candès, J. Romberg, and T. Tao, “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,” *Information Theory, IEEE Transactions on*, vol. 52, no. 2, pp. 489–509, 2006.
- [13] J. A. Tropp and A. C. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *Information Theory, IEEE Transactions on*, vol. 53, no. 12, pp. 4655–4666, 2007.
- [14] M. Vetterli, P. Marziliano, and T. Blu, “Sampling signals with finite rate of innovation,” *Signal Processing, IEEE Transactions on*, 2002.
- [15] P. Dragotti, M. Vetterli, and T. Blu, “Sampling moments and reconstructing signals of finite rate of innovation: Shannon meets strang–fix,” *Signal Processing, IEEE Transactions on*, vol. 55, no. 5, pp. 1741–1757, 2007.
- [16] T. Blu, P. Dragotti, M. Vetterli, P. Marziliano, and L. Coulot, “Sparse sampling of signal innovations,” *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 31–40, 2008.
- [17] M. Mishali and Y. Eldar, “From theory to practice: Sub-nyquist sampling of sparse wideband analog signals,” *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 2, pp. 375–391, 2010.
- [18] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss, “Near-optimal sparse fourier representations via sampling,” in *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, ser. STOC ’02. New York, NY, USA: ACM, 2002, pp. 152–161. [Online]. Available: <http://doi.acm.org/10.1145/509907.509933>
- [19] A. C. Gilbert, S. Muthukrishnan, and M. Strauss, “Improved time bounds for near-optimal sparse fourier representations,” in *Proc. SPIE Wavelets XI*, 2003.
- [20] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, “A tutorial on fast fourier sampling,” *Signal Processing Magazine, IEEE*, 2008.
- [21] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, “Nearly optimal sparse fourier transform,” in *Proc. of the 44th SOTC*. ACM, 2012.
- [22] ———, “Simple and practical algorithm for sparse fourier transform,” in *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2012, pp. 1183–1194.
- [23] M. Iwen, A. Gilbert, and M. Strauss, “Empirical evaluation of a sub-linear time sparse dft algorithm,” *Communications in Mathematical Sciences*, vol. 5, no. 4, pp. 981–998, 2007.
- [24] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi, “Sample-optimal average-case sparse fourier transform in two dimensions,” *arXiv preprint arXiv:1303.1209*, 2013.
- [25] M. Iwen, “Combinatorial sublinear-time fourier algorithms,” *Foundations of Computational Mathematics*, vol. 10, no. 3, pp. 303–338, 2010.
- [26] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, “Improved low-density parity-check codes using irregular graphs,” *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 585–598,

2001.

- [27] T. Richardson and R. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding,” *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 599–618, 2001.
- [28] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.
- [29] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, “Efficient erasure correcting codes,” *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 569–584, 2001.
- [30] E. J. Candes and T. Tao, “Decoding by linear programming,” *Information Theory, IEEE Transactions on*, vol. 51, no. 12, pp. 4203–4215, 2005.
- [31] R. Scheibler, S. Haghhighatshoar, and M. Vetterli, “A fast hadamard transform for signals with sub-linear sparsity,” *arXiv preprint arXiv:1310.1803*, 2013.
- [32] X. Li, J. Bradley, S. Pawar, and K. Ramchandran, “Robustifying the sparse walsh-hadamard transform without increasing the sample complexity of $o(k \log n)$,” in *Information Theory Proceedings (ISIT), 2014 IEEE International Symposium on*. IEEE, 2014.
- [33] S. Pawar, V. Ekambaram, and K. Ramchandran, “Computationally-efficient blind sub-nyquist sampling for sparse spectra,” in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 1065–1068.
- [34] R. Pedarsani, K. Lee, and K. Ramchandran, “Phasecode: Fast and efficient compressive phase retrieval based on sparse-graph-codes,” *arXiv preprint arXiv:1408.0034*, 2014.