ISIT'2013 1569720475

Computing a k-sparse n-length Discrete Fourier Transform using at most $4k$ samples and $O(k \log k)$ complexity

Sameer Pawar and Kannan Ramchandran
Dept. of Electrical Engineering and Computer Sciences, U.C. Berkeley.
{spawar, kannanr}@eecs.berkeley.edu

***Abstract*—Given an n-length input signal $\mathbf{x}$, it is well known that its Discrete Fourier Transform (DFT), $\mathbf{X}$, can be computed in $O(n \log n)$ complexity using a Fast Fourier Transform. If the spectrum $\mathbf{X}$ is exactly k-sparse (where $k << n$), can we do better? We show that asymptotically in k and n, when k is sub-linear in n (i.e., $k \propto n^{\delta}$ where $0 < \delta < 1$), and the support of the non-zero DFT coefficients is uniformly random, we can exploit this sparsity in two fundamental ways (i) sample complexity: we need only $M = rk$ deterministically chosen samples of the input signal $\mathbf{x}$ (where $r < 4$ when $0 < \delta < 0.99$); and (ii) computational complexity: we can reliably compute the DFT $\mathbf{X}$ using $O(k \log k)$ operations, where the constants in the big Oh are small. Our algorithm succeeds with high probability, with the probability of failure vanishing to zero asymptotically in the number of samples acquired, M. Our approach is based on filterless subsampling of the input signal $\mathbf{x}$ using a small set of *carefully chosen uniform subsampling patterns guided by the Chinese Remainder Theorem (CRT)*. Specifically, our subsampling operation on $\mathbf{x}$ is designed to create aliasing patterns on the spectrum $\mathbf{X}$ that "look like" parity-check constraints of good erasure-correcting sparse-graph codes. *We show how computing the sparse DFT $\mathbf{X}$ is equivalent to decoding of these sparse-graph codes* and is low in both sample complexity and decoding complexity. We accordingly dub our algorithm the FFAST (Fast Fourier Aliasing-based Sparse Transform) algorithm. In our analysis, we rigorously connect our CRT based graph constructions to random sparse-graph codes based on a balls-and-bins model and analyze the convergence behavior of the latter using well-studied density evolution techniques from coding theory. We provide simulation results in Section IV that corroborate our theoretical findings, and validate the empirical performance of the FFAST algorithm.**

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[1] of $O(n \log n)$. Many applications of interest involve signals, e.g. audio, image, video data, biomedical signals etc., which have a sparse Fourier spectrum. If the n-length DFT, $\mathbf{X}$, of $\mathbf{x}$, is k-sparse, where $k << n$, can we do better in terms of both **sample** and **computational** complexity of computing the sparse DFT? We answer this question affirmatively. Our main result is that asymptotically in k and n, when k is sub-linear in n (precisely, $k \propto n^{\delta}$ where $0 < \delta < 1$), our proposed FFAST (Fast Fourier Aliasing-based Sparse Transform) algorithm reliably computes the DFT $\mathbf{X}$ using deterministically chosen $M = rk$ samples and $O(k \log k)$ operations. Further, the oversampling ratio $r < 4$ when $0 < \delta < 0.99$,[2] and the constants in the big Oh are small.

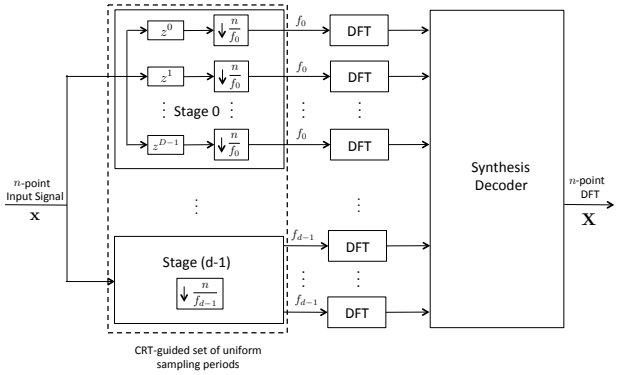

Fig. 1: Schematic block diagram of the FFAST architecture. The n-point input signal $\mathbf{x}$ is subsampled by a set of uniform sampling patterns, guided by the CRT, to obtain dD sub-streams. Each of the d stages has D sub-streams, of length approximately equal to the sparsity k. The aggregate number of samples $M = D\sum_{i=0}^{d-1} f_i \triangleq rk$, for a small constant r. Next, the (short) DFTs, of each of the resulting sub-streams are computed using an efficient FFT algorithm of choice. The big n-point DFT $\mathbf{X}$ is then synthesized from the smaller DFTs using the peeling-like FFAST decoder.

At a high level, our idea is to cleverly exploit rather than avoid the aliasing to induce spectral artifacts that look like the parity constraints of good erasure-correcting codes, e.g., Low-Density-Parity-Check (LDPC) codes [2], fountain codes [3], verification codes [4], etc. Why? These codes: (a) have low computational complexity decoder; and (b) are near-capacity achieving for the erasure channel. The first property bestows the desired low computational complexity, while the second property ensures that the sample complexity of the FFAST algorithm is near-optimal. But how do we achieve this goal? We cannot induce any arbitrary code in the spectral domain at our 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)* [5], 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:

This research was funded in part by NSF grant 1116404.

[1]Recall that a single variable function $f(x)$ is said to be $O(g(x))$, if $\lim_{x\to\infty} |f(x)| < c|g(x)|$ for some constant c.

[2]Our analysis applies for any value of $0 < \delta < 1$. The oversampling ratio r increases as δ approaches 1, e.g., when $\delta = 0.999$, $r < 5$.

- The "very-sparse" regime (sparsity-index $0 < \delta \leq 1/3$), where the subsampling patterns are based on relatively co-prime numbers.
- The "less-sparse" regime ($1/3 < \delta < 1$), where the subsampling patterns comprise of "cyclically-shifted" overlapping co-prime integers.

Our approach is summarized in Fig. 1. Note that the FFAST algorithm requires the signal length n to be a product of a few (typically 3 to 9) distinct primes[3]. We provide a brief summary of simulation results in Section IV to validate the empirical performance of the FFAST algorithm. For more extensive simulation results see [1], where we apply the FFAST algorithm to a wide variety of exactly and approximately sparse 1-D as well as 2-D signals, including applications like MRI.

Related Work: The problem of computing a sparse DFT of a signal is related to the rich literature of frequency estimation [6], [9], [10], compressive-sensing (CS) [7], [8] and sampling signals with finite rate of innovation [12]. While the frequency estimation literature focus on 'super-resolution' techniques based on well-studied statistical methods like MUSIC and ESPRIT [9], [10], our approach combines tools from coding theory, graph theory and signal processing. Unlike the compressive sensing problem, where the resources to be optimized are the number of measurements (where each measurement can further be a linear combination of multiple input samples) and the recovery cost, in our problem, we want to minimize the *number of input samples* processed by an algorithm in addition to the recovery cost.

A number of previous works [13], [14] and references therein, 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 (typically of length $O(k\log(n))$), and then recovering them iteratively one at a time. 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., see [15]. In [16], the author proposes sub-linear time algorithms with a sample and computational complexity of $O(k\ poly(\log n))$ to compute a sparse DFT, either with high probability or zero-error. The algorithm in [16] exploits the CRT to identify the locations of the non-zero DFT coefficients using $O(poly(\log n))$ number of sampling patterns. In contrast, the FFAST algorithm uses a *constant* number of subsampling patterns, guided by the CRT, to induce 'good' sparse-graph codes.

In summary, to the best of our knowledge, the FFAST algorithm is the first that we are aware of to compute an exactly k-sparse n-point DFT that has all of the following features:

- it has $O(k)$ sample complexity and $O(k\log k)$ computational complexity;
- it covers the *entire* sub-linear regime of $0 < \delta < 1$;
- it has a probability of failure that vanishes to zero asymptotically;
- it features the novel use of the Chinese Remainder Theorem to guide the design of a small deterministic set of uniform subsampling patterns that induce good sparse-graph channel codes.

The rest of the paper is organized as follows. Section II states the problem and presents the main result of the paper. Section III exemplifies the mapping of computing the DFT to decoding over an appropriate sparse-graph code. Section IV provides simulation results that corroborate our theoretical findings, and validate the empirical performance of the FFAST algorithm and we conclude the paper in Section V.

II. Problem formulation and main result

A. Problem formulation

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

$$x[p] = \sum_{q=0}^{k-1} a_q e^{2\pi i \omega_q p/n}, \qquad p = 0, 1, \ldots, n-1, \qquad (1)$$

where, we assume that the discrete frequencies ω_q are *uniformly randomly* distributed in $\{0, 1, \ldots, n-1\}$ and the amplitudes $a_q \in \mathbb{C}$, for $q = 0, 1, \ldots, k-1$. We consider the problem of identifying the k unknown frequencies and the corresponding complex amplitudes a_q from the time domain samples $\mathbf{x}$.

Notation	Description
n	Ambient dimension of the signal $\mathbf{x}$.
k	Number of non-zero DFT coefficients.
δ	Sparsity-index: $k \propto n^{\delta}, 0 < \delta < 1$.
M	Sample complexity: Number of samples of $\mathbf{x}$ used by the FFAST algorithm to compute the DFT $\mathbf{X}$.
$r = M/k$	Oversampling ratio.
d	Number of stages in the "sub-sampling front end" of the FFAST architecture.
D	Number of sub-streams of input samples per stage in the FFAST architecture.
f_i	Number of samples of $\mathbf{x}$ per sub-stream in the i^{th} stage of the FFAST architecture.
$(a)_N$	a modulo N.

TABLE I: Glossary of important notations and definitions used in the rest of the paper (also see Figure 1 for details).

B. Main result

Theorem 1. *For any $0 < \delta < 1$, and n large enough, there exists a FFAST algorithm with parameters (n, k, M), where $k = O(n^{\delta})$, such that the FFAST algorithm can compute a k-sparse DFT $\mathbf{X}$ of an n-length input $\mathbf{x}$ with:*

1) **Sample complexity:** *The algorithm needs M samples of $\mathbf{x}$, where M can be as small as rk, for $r > 1$ a small constant that depends on the sparsity index δ;*
2) **Computational complexity:** *The computational complexity is $cM\log(M)$, where the constant c is small and is related to the constants associated with computing a few approximately-M-length FFTs*[4].

[3]This is not a major restriction as 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.

[4]Note that when $M = rk$, the computational complexity is $O(k\log k)$.

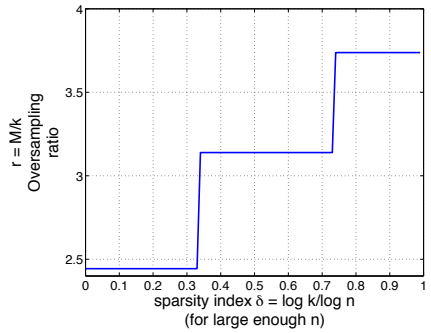

Fig. 2: The plot shows the achievable tradeoff between the oversampling ratio $r = M/k$, and the sparsity index δ for $0 < \delta < 0.99$, where $k \propto n^{\delta}$. The FFAST algorithm computes the k-sparse n-point DFT $\mathbf{X}$ of the n-point signal $\mathbf{x}$, for r as low as the threshold given in the plot. Note that for nearly the entire sub-linear regime of practical interest, i.e., $k < n^{0.99}$, the oversampling ratio $r < 4$.

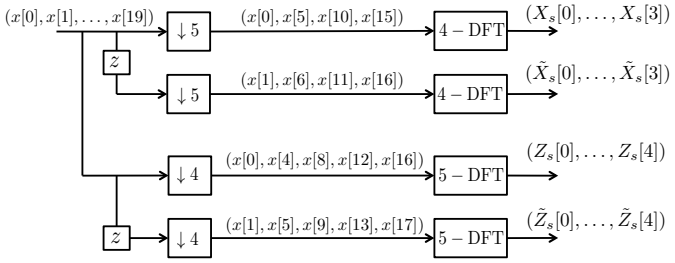

Fig. 3: A toy-example of the front end sub-sampling architecture of the FFAST algorithm. The input to the FFAST architecture is a 20-point discrete-time signal $\mathbf{x}$. The input signal and its unit delayed version are first subsampled by 5 to obtain two sub-streams ($D = 2$), each of length $f_0 = 4$. A 4-point DFT of each sub-stream is then computed to obtain the observations $(X_s[.], \tilde{X}_s[.])$. Similarly, downsampling by 4 followed by a 5-point DFT provides the second set of $f_1 = 5$ observations $(Z_s[.], \tilde{Z}_s[.])$.

3) **Probability of success:** *The algorithm successfully computes the k-sparse DFT $\mathbf{X}$ with probability at least 1-$O(1/M)$.*

Proof: See Appendix A for a sketch of proof and [1] for details. ■

III. Fourier transform using decoding on sparse-graphs

In this section we describe our (deterministic) sub-sampling "front-end" architecture as well as the associated FFAST decoding 'back-end" algorithm for computing the k-sparse n-point DFT, and we will connect this to the framework of decoding over sparse-graph codes. We will use simple examples. Consider a 20-point discrete-time signal $\mathbf{x} = (x[0], \ldots, x[19])$ whose 20-point DFT $\mathbf{X}$ is 5-sparse. Let the 5 non-zero DFT coefficients of $\mathbf{x}$ be $X[1] = 1, X[3] = 4, X[5] = 1, X[10] = 3$ and $X[13] = 7$.

1) **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 uniform subsampling of $\mathbf{x}$ by a factor of 5 (see Fig. 3) to get $\mathbf{x}_s = (x[0], x[5], x[10], x[15])$. Then, the 4-point DFT $\mathbf{X}_s$ has coefficients $(X_s[0], X_s[1], X_s[2], X_s[3]) = (0, 9, 3, 4)$. In general if the sampling period is N (we assume that N divides n) then $X_s[i] = \sum_{j \equiv (i)_{n/N}} X[j]$.

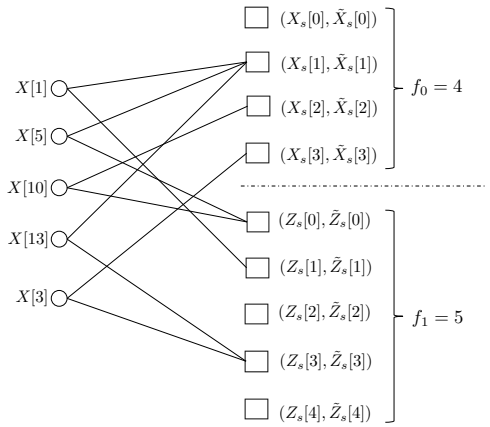

Fig. 4: A 2-left regular degree bi-partite graph, where the variable (left) nodes correspond to the unknown DFT coefficients and the check (right) nodes are the observations obtained through the FFAST architecture shown in Fig. 3, for the 20-point example signal $\mathbf{x}$. Each check node has two complex valued observations.

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

A. Computing a sparse DFT is equivalent to decoding on a sparse-graph

The 20-point example signal $\mathbf{x}$ when processed through the front-end of the FFAST architecture, produces the observations $(X_s[\cdot], \tilde{X}_s[\cdot])$ and $(Z_s[\cdot], \tilde{Z}_s[\cdot])$ as shown in Fig. 3. The relation between the unknown DFT coefficients of the signal $\mathbf{x}$ and the output observations generated through the FFAST architecture can be represented using a bi-partite graph (see Fig. 4). The left (variable) nodes correspond to the non-zero DFT coefficients of $\mathbf{x}$ (unknown and to be computed) and the right (check) nodes correspond to the observations generated through the FFAST architecture of Fig. 3. Note that the observation $(X_s[1], \tilde{X}_s[1])$ is a combination of three non-zero DFT coefficients of $\mathbf{x}$, we refer to such check nodes as "multi-tons". Likewise the check nodes whose observation consists of exactly one non-zero DFT coefficient of $\mathbf{x}$ are referred to as "singletons", e.g. $(X_s[2], \tilde{X}_s[2])$, while the ones that consists of all the zero DFT coefficients are called "zero-tons", e.g., $(Z_s[2], \tilde{Z}_s[2])$. Consider first that a "genie" informs us which check nodes are singletons, as well as, the location and value of the contributing non-zero DFT coefficient. We will later explain (in Section III-B) how to get rid of the genie. Then, the FFAST algorithm can recover the DFT coefficients that are connected to a singleton check node, e.g., $X[1], X[3]$ and $X[10]$. After, subtracting the contribution of these uncovered DFT coefficients from the observations, more singleton check nodes are created. Thus, with the help of the genie, the FFAST algorithm can potentially recover all the 5 non-zero DFT coefficients of $\mathbf{x}$ via an iterative peeling procedure on the graph of Fig. 4. Thus, the FFAST architecture has *transformed the problem of computing the DFT of $\mathbf{x}$ into that of decoding over a sparse bi-partite graph* of Fig. 4.

Algorithm 1 FFAST Decoder

Set the initial estimate of the n-point DFT $\hat{\mathbf{X}} = 0$.
for each iteration **do**
  for each check node **do**
    Let $(y[0], y[1])$ be the observations of a check node.
    if $y[0] = y[1] = 0$ **then**
      the check node is a *zero-ton*.
    else
      singleton-test: If the location estimate $\hat{loc} = \frac{n}{2\pi}(\angle y[1] - \angle y[0])$, is an integer between 0 and $n-1$, then the check node is a *singleton*.
      Peeling: If the check node is a singleton set $\hat{\mathbf{X}}[\hat{loc}] = y[0]$. Subtract the contribution of $\hat{\mathbf{X}}[loc]$ from all the neighboring check nodes.
    else
      the check node is a *multi-ton*.
    end if
  end for
end for

B. FFAST Decoder

In this section, we show how to get rid of the "genie" by using the additional observations $\tilde{X}_s[\cdot]$ and $\tilde{Z}_s[\cdot]$. First, we make the following important observation: if a check node is a singleton, then by computing the phase of the ratio of its observations, e.g., $\frac{n}{2\pi}\angle(\tilde{X}_s[2]/X_s[2]) = 10$, we can identify the index and the first observation $X_s[2] = 3$, provides the value of the non-zero DFT coefficient, connected to this singleton check node. In general, as shown in [17], identifying the support of a non-zero DFT coefficient connected to a singleton check node from multiple observations is equivalent to the problem of estimating the frequency of a single sinusoid from its time domain samples. In particular, by having two[5] observations per check node, one can identify whether the check node is a zero-ton, singleton or multi-ton with overwhelmingly high reliability. Algorithm 1 provides the pseudocode for the FFAST decoder. It is easy to verify that performing the peeling procedure on the example graph of Fig. 4 results in successful decoding, with the coefficients being uncovered in the following possible order: $X_{10}, X_3, X_1, X_5, X_{13}$.

IV. Simulation Results

In this section we validate the empirical performance of our FFAST algorithm for exactly sparse 1-D signal. For more extensive simulation results see [1], where we apply the FFAST algorithm to a wide variety of exactly and approximately sparse 1-D as well as 2-D signals, including applications like MRI. The simulation setup for exactly sparse 1-D signal is as follows:

- **Very-sparse regime** $(0 < \delta \leq 1/3)$**:** The input signal $\mathbf{x}$ is of ambient dimension $n = 511 \times 512 \times 513 \approx 134 \times 10^6$. The sparsity parameter k is varied from 400 to 1300, which corresponds to the very-sparse regime, i.e., $k \propto n^{1/3}$. The FFAST architecture has $d = 3$ stages and $D = 2$ sub-streams per stage (see Fig. 1). The number of samples per sub-stream, for the three different stages are relatively co-prime $f_0 = 511$, $f_1 = 512$ and $f_2 = 513$ respectively and the total number of samples[6] $M < 2(f_0 + f_1 + f_2) = 3072$.
- **Less-sparse regime** $(1/3 < \delta < 1)$**:** The input signal $\mathbf{x}$ is of ambient dimension $n = 16 \times 17 \times 19 \times 21 \approx 0.1 \times 10^6$. The sparsity parameter k is varied from 5000 to 19000, which corresponds to the less-sparse regime of $n^{0.73} < k < n^{0.85}$. The FFAST architecture has $d = 4$ stages and $D = 2$ sub-streams per stage. The number of samples per sub-stream, for the four different stages are $f_0 = 16 \times 17 \times 19 = 5168$, $f_1 = 17 \times 19 \times 21 = 6783$, $f_2 = 19 \times 21 \times 16 = 6384$ and $f_3 = 21 \times 16 \times 17 = 5712$ respectively. Note that the number of samples in the four stages are composed of "cyclically-shifted" co-prime numbers. The total number of samples $M < 48094$.
- For each run, an n-dimensional k-sparse signal $\mathbf{X}$ is generated with non-zero values $X_i \in \{\pm 10\}$ with uniformly random support in $\{0, 1 \ldots, n-1\}$. The time domain signal $\mathbf{x}$ is then generated from $\mathbf{X}$ using an IDFT operation and given to the FFAST algorithm as an input.
- Each sample point in Fig. 5 is averaged over 10000 runs.
- Decoding is successful if all the DFT coefficients are recovered perfectly.

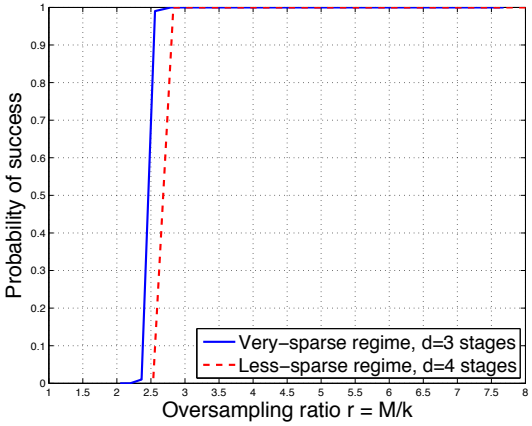

Fig. 5: The probability of success of the FFAST algorithm as a function of the oversampling ratio r. The plot is obtained for the two different sparsity regimes: i) the very-sparse regime and ii) the less-sparse regime. Each point in the plot is obtained by averaging over 10000 runs. Note that the oversampling ratio $r < 4$ is sufficient for successful computation of the k-sparse n-point DFT using the FFAST algorithm.

V. Conclusion

In this paper we have proposed a FFAST algorithm to compute a k-sparse n-point DFT $\mathbf{X}$, using deterministically chosen $O(k)$ samples in $O(k \log k)$ operations. Our approach is based on filterless subsampling of the input signal $\mathbf{x}$ using a small set of carefully chosen uniform subsampling patterns guided by the Chinese Remainder Theorem (CRT). While the theoretical claims in this paper are for 1-D exactly k-sparse signal $\mathbf{x}$, the FFAST algorithm can be extended to the more general multi-dimensional and approximately sparse signals,

[5]For the case when the signal has an approximately sparse DFT, more observations per check node will be needed to robustly identify whether the check node is a zero-ton, singleton or multi-ton. I.e., in Fig. 1, D will no longer be 2 as in the exactly sparse case.

[6]The samples in the different sub-streams overlap, e.g., $x[0]$ is common to all the zero delay sub-streams in each stage. Hence, $M < 2(f_0 + f_1 + f_2)$.

albeit at the cost of increased sample and computational complexity. An empirical evidence of the noise robustness of the FFAST algorithm for 1-D and 2-D signals is provided in [1], which also contains more technical details of this paper. A theoretical analysis of the noise robustness of the FFAST algorithm is part of our future work.

APPENDIX

A. Sketch of a proof of Theorem 1

In this section we provide brief sketch of a proof of Theorem 1 for $k = O(n^{1/3})$, an operating point in the very-sparse regime. In [1], we provide a detailed and complete proof of Theorem 1. Let $\mathcal{F} = \{f_0, f_1, f_2\}$, be a set of pairwise co-prime integers where $f_i = O(k)$, and $m \triangleq \sum_{i=0}^{2} f_i$. Also let $n = \prod_{i=0}^{2} f_i$, i.e., $k = O(n^{1/3})$. The integers f_i's are the number of samples per sub-stream in $d = 3$ different stages of the FFAST architecture (see Fig. 1).

B. Randomized construction based on the "Balls-and-Bins" model: $\mathcal{C}_1^k(\mathcal{F}, m)$

The ensemble $\mathcal{C}_1^k(\mathcal{F}, m)$ of bi-partite graphs with k *variable nodes* on the left and m *check nodes* on the right is constructed as follows. Partition the set of m check nodes into three subsets of f_0, f_1 and f_2 check nodes (see Fig. 4 for an example graph where $f_0 = 4, f_1 = 5, d = 2$ and $k = 5$). Each variable node is connected to one neighboring check node in each of the $d = 3$ subsets, uniformly at random.

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

Partition the set of m check nodes into three subsets of f_0, f_1 and f_2 check nodes. Consider a set $\mathcal{I}$ of k integers, chosen uniformly at random with replacement between 0 and $n-1$. Assign these k integers 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 $i = 0, 1, 2$. A 3-left regular degree bi-partite graph 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 $i = 0, 1, 2$. Note that the modulo rule used to generate a graph in the ensemble $\mathcal{C}_2^k(\mathcal{F}, n)$ is same as the one used in Section III. Thus, the FFAST architecture of Fig. 1, *generates graphs from the CRT ensemble* $\mathcal{C}_2^k(\mathcal{F}, n)$, where the indices $\mathcal{I}$ of the k variable nodes are the locations of the non-zero DFT coefficients[7] of the signal $\mathbf{x}$.

Lemma 2. *The ensemble of bipartite graphs $\mathcal{C}_1^k(\mathcal{F}, m)$ is identical to the ensemble $\mathcal{C}_2^k(\mathcal{F}, n)$.*

Proof: See [1] for details. ∎

The graphs in the ensemble $\mathcal{C}_1^k(\mathcal{F}, m)$ correspond to random sparse-graph codes constructed based on a balls-and-bins model. Hence, one can analyze the performance of the peeling-decoder over the graphs from the ensemble $\mathcal{C}_1^k(\mathcal{F}, m)$, using well-studied density evolution technique as in [18], [19]. Then, using lemma 2 we obtain a bound on the performance of the FFAST decoder over the graphs from the ensemble $\mathcal{C}_2^k(\mathcal{F}, n)$. Next, we highlight the main technical components required to show that the FFAST-decoder successfully computes a k-sparse n-point DFT with high probability.

- *Density evolution:* First, assume that a local neighborhood of a fixed depth of every edge in a typical graph from the ensemble is tree-like, i.e., cycle-free. Under this assumption, all the messages between variable nodes and the check nodes are independent. Using this independence assumption, we derive a recursive equation that represents the expected evolution of the number of singletons uncovered at each round for this typical graph.
- *Convergence to the cycle-free, case:* Using a Doob martingale we show that a randomly chosen graph from the ensemble behaves like a "typical" graph and the peeling-decoder decodes all but an arbitrarily small fraction of the variable nodes with high probability, in a constant number of iterations.
- *Completing the decoding using the graph expansion property:* First, we show that if a graph is an "expander", then once the peeling-decoder has successfully decoded all but an arbitrarily small fraction of the variable nodes, it decodes all the variable nodes. Later, we show that a random graph from the ensemble is a good expander with high probability.

[7] The integers in the set $\mathcal{I}$ are chosen uniformly at random, with replacement, between 0 and $n-1$. A set $\mathcal{I}$ with repeated elements then corresponds to a signal with fewer than k non-zero DFT coefficients.

REFERENCES

[1] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete fourier transform using at most 4k samples and o (k log k) complexity," *arXiv preprint arXiv:1305.0870*, 2013.

[2] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.

[3] M. Luby, "Digital fountain, inc. luby@ digitalfountain. com," 2002.

[4] M. Luby and M. Mitzenmacher, "Verification codes: simple ldpc codes for large alphabets," in *Proc. 40th Annu. Allerton Conf.*, 2002.

[5] R. Blahut, *Fast algorithms for digital signal processing*, 1985.

[6] R. Prony, "Essai experimental–,-," *J. de l'Ecole Polytechnique*, 1795.

[7] D. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[8] 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.

[9] R. Schmidt, "Multiple emitter location and signal parameter estimation," *Antennas and Propagation, IEEE Transactions on*, 1986.

[10] 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.

[11] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *Information Theory, IEEE Transactions on*, vol. 53, no. 12, pp. 4655–4666, 2007.

[12] M. Vetterli, P. Marziliano, and T. Blu, "Sampling signals with finite rate of innovation," *Signal Processing, IEEE Transactions on*, 2002.

[13] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A tutorial on fast fourier sampling," *Signal Processing Magazine, IEEE*, 2008.

[14] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Nearly optimal sparse fourier transform," in *Proc. of the 44th SOTC*. ACM, 2012.

[15] 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.

[16] M. Iwen, "Combinatorial sublinear-time fourier algorithms," *Foundations of Computational Mathematics*, vol. 10, no. 3, pp. 303–338, 2010.

[17] S. Pawar and K. Ramchandran, "A hybrid dft-ldpc framework for fast, efficient and robust compressive sensing," in *Proc. of Allerton*, 2012.

[18] 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.

[19] 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.