Time-Data Trade-off in the Sparse Fourier Transform

Abdulmalik Aldharrab, Mike E. Davies

Institute for Digital Communications, University of Edinburgh, EH9 3JL, UK
Email: {a.aldharrab, mike.davies}@ed.ac.uk

***Abstract*—It has been shown that the Discrete Fourier Transform (DFT) can be computed in sublinear time from a sublinear number of samples when the target spectrum is sparse. However, this is usually only expressed qualitatively in terms of the order of number of computations/samples. Here we investigate the explicit time-data tradeoff for the Sparse Fourier Transform (SFT) algorithm proposed by Pawar and Ramchandran using coding theoretic tools. This leads to an optimal oversampling rate and algorithm configuration that minimises computation while keeping the required number of time domain samples close to the minimum value.**

I. Introduction

The time domain samples used to calculate the Discrete Fourier Transform (DFT) should be obtained by sampling the analogue signal at more than Nyquist rate which is twice the maximum frequency that needs to be preserved. However, if the frequency domain of a signal contains only few non-zero components (sparse) the DFT can be calculated from fewer samples using the Sparse Fourier Transform (SFT) techniques.

Sparse Fourier Transform algorithms can be generally divided into two main categories, the first one is the Windowed Sparse Fourier Transform [1] [2] [3], and according to the empirical evaluation conducted by Gilbert et al. (2014) [4] the best results achieved using this technique is by Hassanieh et al. (2012) [3]. The second category is the Aliasing Based Sparse Fourier Transform. This category was invented at least three times independently by [5], [6] and [7] which gives an indication about the importance of such approach.

A. Relation to Prior Work

In most SFT algorithms the minimum number of time domain samples required by the algorithm to achieve high probability of success is not accurately defined. However, Pawar and Ramchandran (2013) [5] use coding theoretic tools to identify the minimum required time domain samples with high accuracy. Nevertheless, using the minimum samples leads to high computational complexity. This paper utilises the coding-theoretic tools to investigate the time-data tradeoff for the Aliasing Based SFT and hence derive an optimal operating point which leads to minimum computational complexity while maintaining the required number of time domain samples to a near minimum value. The ability to place tight bounds on an algorithm is of a great benefit since it allows the algorithm to be operated with high efficiency.

One of the limitations of the considered algorithm is the assumption that the locations of the non-zero frequency components follow an *average case* signal model which assumes that the support of the signal $\mathbf{X} \in \mathbf{C}^N$ is drawn uniformly at random from the set $\{0, \ldots, N-1\}$. Furthermore, the conducted analysis assumes that signals are already in the digital domain. This can be extended to the analogue domain by a proper manipulation of Fourier transforms in a way similar to the work of Feng and Bresler (1996) [8]. Moreover, the considered signals are assumed noiseless and *exactly sparse*. These limitations will be addressed in future work.

Section II introduces the algorithm and Section III maps the problem to a sparse bipartite graph to allow utilising the well-established coding theory. The tradeoff between the number of time domain samples used and the computational complexity is introduced in Section IV. Section V introduces the optimal operating point that leads to both low computational complexity and low sample complexity.

II. Shifting and Sub-sampling in Time

At a high level the algorithm uses multiple stages, each sub-sampling the original signal in time using a unique sub-sampling factor. For each stage the DFTs of two sub-sampled in time signals are calculated where one of the signals is shifted in time prior to sub-sampling [5].

Sub-sampling the time domain signal $\boldsymbol{x} \in \mathbf{C}^N$ will introduce aliasing to the frequency domain signal $\mathbf{X} \in \mathbf{C}^N$. A relation between the original signal and the sub-sampled in time version can be derived starting from the definition of the Inverse-DFT and sub-sampling in time by a factor of p:

$$x[p \cdot n] = \frac{1}{N} \sum_{l=0}^{N-1} X(l) \cdot e^{j2\pi ln/(N/p)} \tag{1}$$

for any m and n that are integers:

$$x[p \cdot n] = \frac{1}{N/p} \sum_{k=0}^{N/p-1} \underbrace{\frac{1}{p} \sum_{m=0}^{p-1} X(k + m \cdot N/p)}_{X_p(k)} \cdot e^{j2\pi kn/(N/p)} \tag{2}$$

ISBN 978-0-9928626-7-1 © EURASIP 2017

The result of sub-sampling in time is a signal that is of a lower dimension $\mathbf{X_p} \in \mathbf{C}^{N/p}$ and can be related to the original high dimensional signal $\mathbf{X} \in \mathbf{C}^N$ as follows:

$$X_p[((k))_{N/p}] = \frac{1}{p}\sum_{m=0}^{p-1} X[((k+m\cdot N/p))_N] \tag{3}$$

where $((\bullet))_N$ indicates periodicity over a period of N. The frequency components that are N/p bins apart will collide into one frequency bin. However, using sub-sampling factors that result in signals with lengths that are either co-prime integers or cyclically shifted sets of co-prime integers will ensure that the components that collide in one of the sub-sampled stages do not collide in the others [5].

Shifting in time will multiply each frequency component by an exponential term:

$$\begin{aligned} x[((n))_N] &\xrightarrow{\mathcal{F}} X[((k))_N] \\ x[((n+n_o))_N] &\xrightarrow{\mathcal{F}} X[((k))_N]\cdot e^{j2\pi k(n_o)/N} \end{aligned} \tag{4}$$

where $x[((n+n_o))_N]$ is the time domain signal circularly shifted by n_o and $X[((k))_N]$ is the Discrete Fourier Transform of $x[((n))_N]$. Observing the exponential term introduced by the shift in time it can be clearly seen that it carries information about the location in the frequency domain at which it exists (k).

Consider the signal $\mathbf{X} \in \mathbf{C}^{20}$ which has a few non-zero frequency components $\{X(3) = 3,\ X(8) = 1.7,\ X(10) = 1,\ X(13) = 2.4,\ X(18) = 4.3\}$. Let $\mathbf{y}_{m,n}$ be the m^{th} *observation vector* which contains the content of the m^{th} frequency bin in both the sub-sampled signal and the shifted (by $n_o = 1$) sub-sampled signal in stage $(n = 0)$ which sub-samples in time by $p = 5$.

$$\mathbf{y}_{0,0}{}^{1} = X(8)\times\begin{bmatrix}1\\ W^8\end{bmatrix} = 1.7\times\begin{bmatrix}1\\ W^8\end{bmatrix} \tag{5}$$

where $W = e^{j2\pi/N}$ is the N^{th} root of unity, and $j = \sqrt{-1}$. Since $\mathbf{y}_{0,0}$ satisfies the following conditions:

$$|y_{m,n}[0]| = |y_{m,n}[1]| \tag{6}$$

$$\left(\frac{N}{2\pi}\right)\times\angle(\ y_{m,n}[1]/y_{m,n}[0])\ integer \in \{0,\ \dots,N-1\} \tag{7}$$

This means that $\mathbf{y}_{0,0}$ can be detected as containing only a single Fourier component ***(Single-ton)*** and subsequently the value and location of that component in $\mathbf{X} \in \mathbf{C}^{20}$ can be recovered as follows:

- Location: $k_{est} = (N/2\pi)\times\angle(\ y_{m,n}[1]/y_{m,n}[0])$.
- Value: $X(k_{est}) = y_{m,n}[0]$.

However, $\mathbf{y}_{2,0}$ does not satisfy any of the conditions and it is called a ***(Multi-ton)***:

$$\mathbf{y}_{2,0} = 1\times\begin{bmatrix}1\\ W^{10}\end{bmatrix} + 4.3\times\begin{bmatrix}1\\ W^{18}\end{bmatrix} = \begin{bmatrix}5.3\\ 2.48-j\ 2.53\end{bmatrix} \tag{8}$$

The non-zero components in $\mathbf{X} \in \mathbf{C}^{20}$ can be recovered by alternating between stage 0 (sub-sampled by $p = 5$) and stage 1 (sub-sampled by $p = 4$) as shown in Fig. 1. Removing the recovered components from all sub-sampled signals will iteratively convert more ***(Multi-tons)*** to ***(Single-tons)*** allowing further recovery [5].

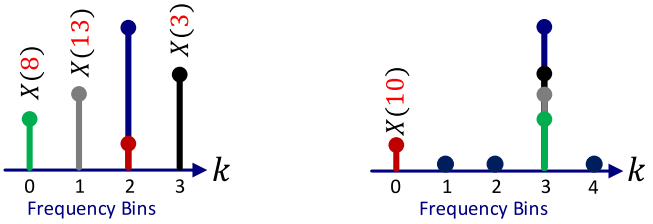

Fig. 1. $\mathbf{X} \in \mathbf{C}^{20}$ sub-sampled in time by $p = 5$ (left), by $p = 4$ (right). Non-colliding components $\{X(8),\ X(13),\ X(3),\ X(10)\}$ can be recovered and removed to allow recovering colliding components $\{X(18)\}$.

III. Relation to Coding Theory

Pawar and Ramchandran (2013) [5] map the problem of recovering sparse signals from sub-sampled versions to fit a randomized graph that is constructed based on the "Balls-and-Bins" model as shown in the example of Fig. 2. Here there are $d = 2$ edges originated from each left node which means that the left degree is $d = 2$ and this corresponds to the number of stages used (the number of sub-sampled signals).

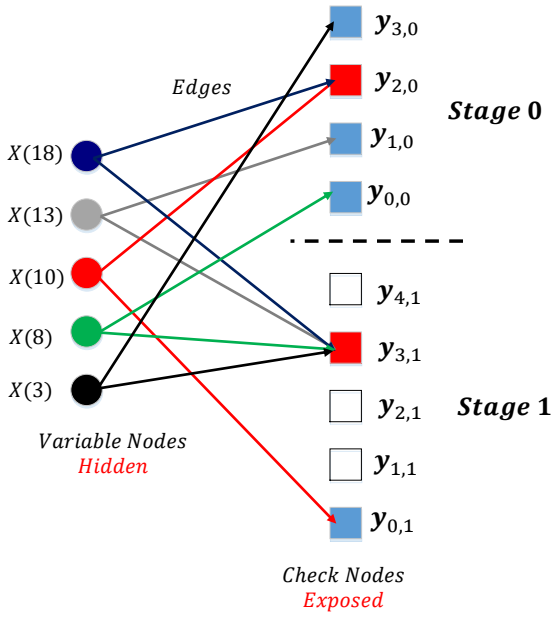

Fig. 2. Mapping the problem of recovering sparse signals from sub-sampled versions into a sparse bipartite graph to utilise the well-established coding theory.

[1]Scaling is omitted from observation vectors, however, it follows the relations provided by equations (1) until (3).

ISBN 978-0-9928626-7-1 © EURASIP 2017

For large N (signal length) and k (sparsity) the number of edges remaining in the graph after each iteration can be estimated using the asymptotic theory Density Evolution which gives an estimation of the density of the remaining edges in the graph after each iteration as follows[2]:

$$P_r = \left(1 - e^{-\frac{1}{\nu} P_{r-1}}\right)^{d-1} \qquad r = 1,\ 2,\ \ldots \tag{9}$$

Where:
P_r: Probability that an edge exists after $\{r\}$ iterations.
$\nu \approx \frac{n_b}{k \cdot d}$: Per-stage oversampling ratio. d: Number of stages.
$n_b = \sum_{i=0}^{d-1} l_i$: Total number of observation vectors.
l_i: Length of the i^{th} sub-sampled signal.
$m = 2 \times n_b$: Total time domain samples used.

Under the *average case* assumption, using sub-sampled signals with comparable lengths $l_0 \approx l_1 \approx \ \ldots \ \approx l_{d-1}$ will reduce the probability of collisions. This will allow defining the oversampling ratio as $\eta = \frac{m}{k} \approx 2d\nu$. The density of the edges remaining in the graph is directly related to the fraction of the non-recovered frequency components and as demonstrated by Fig. 3 as long as $P_r < P_{r-1}$ after each iteration the density will reduce. This can be guaranteed by choosing oversampling ratio which ensures that the Density Evolution relation given by (9) will achieve convergence. Moreover, larger oversampling ratios will allow P_r to reduce in larger steps after each iteration which will allow faster convergence as illustrated by Fig. 5. Pawar and Ramchandran (2013) [5] show that the algorithm closely follows the theoretical results obtained based on Density Evolution.

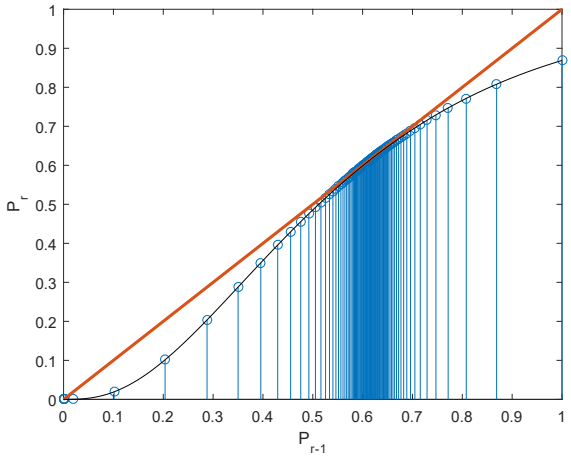

Fig. 3. The behaviour of Density Evolution when the oversampling ratio is chosen as $\eta = 2.5944$ which is just above the minimum required oversampling ratio for ($d = 4$ *stages*).

IV. The SFT Time-Data Tradeoff

Despite the fact that it has been shown that the complexity of such algorithms is $O(k \log k)$ [5] the complexity has an interesting behaviour that depends on the different parameters. To better understand this behaviour Density Evolution is used to identify the number of iterations $\{r\}$ required to achieve probability $\{P_r < \epsilon\}$ that an edge exists, where we set $\epsilon = 1 \times 10^{-8}$ and this is plotted in Fig. 4. This corresponds to high probability that the algorithm converges. As shown in Fig. 5 a minimum oversampling ratio is required to allow convergence and higher oversampling ratios require fewer iterations to achieve convergence, however, the relations do not clarify the impact of increasing oversampling on the complexity. To quantify such an effect the complexity of each part of the algorithm is analysed to find out the overall complexity. Since the complexity of different operations is usually expressed using $O(\bullet)$ notation which gives the order of the complexity rather than the actual complexity this notation is avoided by introducing some simplifying assumptions. These assumptions can be modified with the exact complexity associated with the specific hardware.

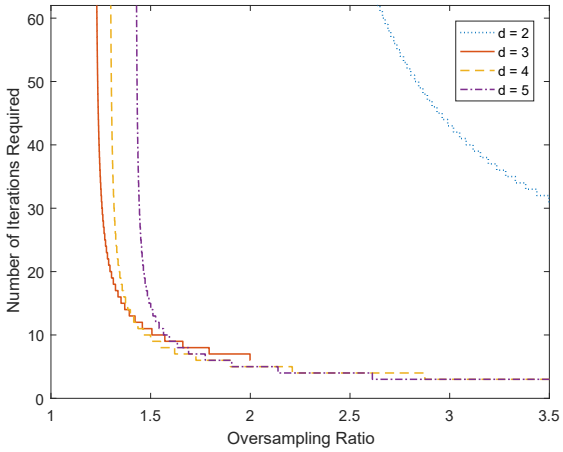

Fig. 4. The relation between the oversampling ratio η and the minimum number of iterations required to achieve low probability that an edge exists (convergence) for different number of stages d.

The algorithm starts by calculating $2 \times d$ DFTs of comparable lengths. Each DFT has a length $\approx \frac{m}{2.d}$. This makes the complexity of calculating the short DFTs:

$$C_1 \approx m \log\left(\frac{m}{2 \cdot d}\right) \tag{10}$$

During every iteration the algorithm searches for non-colliding components by performing the following steps over all the $(n_b = \frac{m}{2})$ *observation vectors*. The first step is checking whether the *observation vector* is empty or not $(y_{m,n}[0] = y_{m,n}[1] = 0)$ and this is assumed to cost two operations. The next check will be whether or not it contains colliding components and this is done through three steps. The first one is comparing the magnitudes of the entries in the observation vector $(|y_{m,n}[0]| = |y_{m,n}[1]|)$ and this is assumed to cost three operations. The second step calculates an estimate of the location from the phase $(k_{est} = (N/2\pi) \times \angle(\ y_{m,n}[1]/y_{m,n}[0]))$ and this requires dividing the two entries

[2]It is worth mentioning that the relation given by Equation (9) is explicitly mentioned in [9] and the recent work by Li, Pawar, and Ramchandran [10].

ISBN 978-0-9928626-7-1 © EURASIP 2017

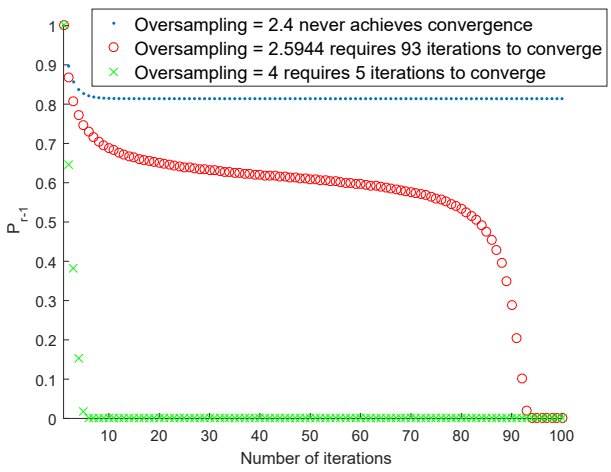

Fig. 5. Number of iterations required to achieve low probability that an edge exists varies based on the oversampling ratio used $(d = 4\ stages)$.

of the observation vector and calculating the angle of the result and then multiplying by $(N/2\pi)$ which is constant and can be calculated once. The second step is assumed to cost three operations. The third step checks that $k_{est} \in \{0, \ \dots \ , N-1\}$ and this can be done by checking if k_{est} is smaller than the largest entry in the set and it is assumed to cost two operations. This makes the cost of searching for a non-colliding component ***(Single-ton)*** 10 operations per *observation vector*:

$$C_2 = 5 \cdot r(\nu, d) \cdot m \tag{11}$$

where $r(\nu, d)$ is the minimum number of required iterations which is a function of the oversampling ratio and the number of stages used, we will see that this function dominates the cost. If one of the (k) components is detected it will be subtracted from all the DFTs in all the d stages and this is assumed to cost $2.d$ operations. Subtracting the components requires knowing their locations and this requires calculating $(d-1)$ modulo operators, each assumed to cost one operation:

$$C_3 = k \cdot (3d - 1) \tag{12}$$

and this leads to the overall cost of the algorithm:

$$\begin{aligned} C_{\text{total}} &= C_1 + C_2 + C_3 \\ &= m \log\left(\frac{m}{2 \cdot d}\right) + 5 \cdot r(\nu, d) \cdot m + k \cdot (3d - 1) \end{aligned} \tag{13}$$

V. Discussion

For a given sparsity (k) the algorithm requires a minimum number of time domain samples to converge $(m = \eta k)$, hence, if the sparsity level is high the term C_1 is mildly superlinear in m and it will dominate. C_2 dominates the choice of optimal oversampling ratio in most cases. It depends on the number of time domain samples used (m) and the number of iterations required for convergence $(r(\nu, d))$ which both depend on two parameters. The first parameter is the oversampling ratio $\eta \approx 2d\nu$, using η just above the minimum oversampling ratio required will make the algorithm behave in the way shown in Fig. 3. As the contour taken by the algorithm passes closer to the line corresponding to $P_r = P_{r-1}$ a larger number of iterations will be required to achieve convergence, however, increasing the oversampling ratio just a little will move the contour away and this will allow convergence using fewer number of iterations. The second parameter is the number of stages used, d. Writing $P_r = f(P_{r-1})$ we note that for $d = 2$ the gradient $\partial f(0)/\partial P$ is non-zero and this indicates linear convergence, hence, $d = 2$ is not competitive. On the other hand, for $d \geq 3$ the gradient is zero and $d = 3$ has the minimum oversampling requirement. The term C_3 represents the cost of removing the recovered components and this cost is fixed and depends on the number of components that need to be removed which is related to the sparsity level, it also depends on the number of stages from which the components will be removed. Fig. 6 shows an example for $(d = 4)$ stages, it can be noticed that if the oversampling ratio is near the minimum the cost is huge, for instance, if the sparsity $k = 400$ operating at an oversampling ratio of $(\eta = 2.593)$ will require (608000) operations, on the other hand, increasing the oversampling to $(\eta = 4.422)$ will cause the complexity to drop to (53550) operations, which is an order of magnitude smaller and this is at the expense of an increase in the total number of time domain samples used from $m \approx (1038)$ to $m \approx (1769)$.

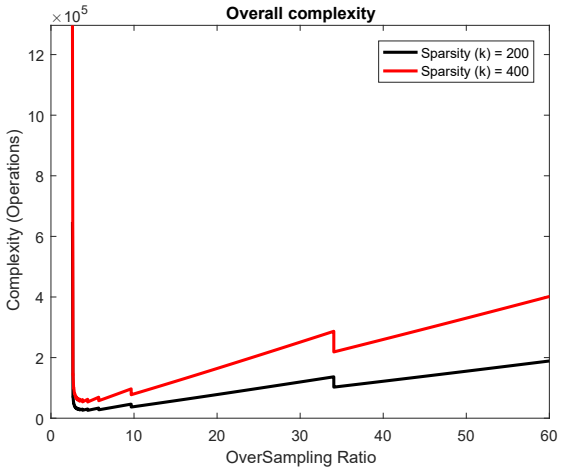

Fig. 6. Overall complexity, operating just above the minimum oversampling ratio will allow achieving minimum complexity while still using a small number of time domain samples.

VI. Conclusion

The computational complexity of the Aliasing Based SFT has been investigated and it was shown that in contrast to minimal sampling requirements an optimum operating point that has both low sample complexity and low computational complexity is above the minimum oversampling. Without such operating point the algorithm will operate either near the critical oversampling which is of a huge cost or above the optimum oversampling which is also of a large cost. The

ISBN 978-0-9928626-7-1 © EURASIP 2017

considered signals are assumed *noiseless* and *exactly sparse* and the time-data tradeoffs for more general SFT problems will be considered in future work.

REFERENCES

[1] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss, "Near-optimal sparse fourier representations via sampling," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, 2002, pp. 152–161.

[2] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A tutorial on fast fourier sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 57–66, March 2008.

[3] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Nearly optimal sparse fourier transform," in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM, 2012, pp. 563–578.

[4] A. C. Gilbert, P. Indyk, M. Iwen, and L. Schmidt, "Recent developments in the sparse fourier transform: A compressed fourier transform for big data," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 91–100, Sept 2014.

[5] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete fourier transform using at most 4k samples and o(k log k) complexity," in *2013 IEEE International Symposium on Information Theory*, July 2013, pp. 464–468.

[6] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi, "Sample-optimal average-case sparse fourier transform in two dimensions," in *the 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Oct 2013, pp. 1258–1265.

[7] S. H. Hsieh, C. S. Lu, and S. C. Pei, "Sparse fast fourier transform by downsampling," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 5637–5641.

[8] P. Feng and Y. Bresler, "Spectrum-blind minimum-rate sampling and reconstruction of multiband signals," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, May 1996, pp. 1688–1691 vol. 3.

[9] S. Pawar, "Pulse: Peeling-based ultra-low complexity algorithms for sparse signal estimation," Ph.D. dissertation, EECS Department, University of California, Berkeley, Dec 2013. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-215.html (Acceded: March, 2017).

[10] X. Li, S. Pawar, and K. Ramchandran, "Sub-linear time compressed sensing using sparse-graph codes," in *2015 IEEE International Symposium on Information Theory (ISIT)*, June 2015, pp. 1645–1649.

ISBN 978-0-9928626-7-1 © EURASIP 2017