International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

Comprehensive review on fast Fourier transform types and their applications

Ahmed Faris Hameed [1] and Osama Qasim Jumah Al-Thahab [2,*]

[1] *Ministry of Electricity, State Company for Electric Power Production, Euphrates Middle Region, Iraq.*
[2] *Department of Electric, College of Engineering, University of Babylon, Iraq.*

International Journal of Science and Research Archive, 2025, 16(03), 1146-1152

Publication history: Received on 16 August 2025; revised on 23 September 2025; accepted on 25 September 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.16.3.2663

Abstract

Here in this paper a review of some Fast Fourier Transform (FFT) algorithms are presented. FFT employs the idea of dividing the Discreet Fourier Transform (DFT) to smaller ones and compute each transform separately, in addition to transform the signals in time domain to frequency domain. There are several types of FFT, some of them deals with even samples (Radix II), while the other deals with any even samples. Off course there are types that deal with odd samples, but they are somehow complicated. Than we introduce other algorithm Sparse Fast Fourier Transform (SFT) which outperform the old FFT and have better performance when we have very large input sequence. Than the multidimensional FFT is reviewed. In addition to summarized some FFT's applications.

Keywords: FFT Algorithms; DSP; 2DFFT; SFT Algorithms; Applications of FFT and SFT

1. Introduction

DFT (*Discreet Fourier Transform*) is considered an important part of digital signal processing, and in various applications, we may need to compute the DFT, which is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{nk} \quad (1)$$

And the Invers DFT:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, W_N^{-nk} \quad (2)$$

Where $W_N = e^{-j\frac{2\pi}{N}}$

And because of the complexity of the equations, we have (N^2) complex multiplication and ($N^2 - N$) complex addition, it's necessary to find more efficient and fast algorithms to compute the DFT. Such algorithms are called FFT (*fast Fourier transform*). So in essence FFT algorithms are simply the efficient way to compute the DFT [1]. The different methods that can be used to compute the DFT efficiently are depends on fundamental idea which is, first dividing the transformation to simple smaller ones, and second compute the transform to this simple problem. [2]. We will see several algorithms that employ this idea

2. Radix-2 Algorithms

The basic idea behind these algorithms is the DFT of a Length-N sequence, which is $N = 2^i$. It can be simply calculated from the two Length-N/2 DFT's of the even index terms and the odd index terms. This is then applied to the two half-

* Corresponding author: Osama Qasim Jumah Al-Thahab

length DFT's to give four quarter-length DFT's, and repeated until we have DFT of length 2 which is considered the FFT butterfly, so N DFT values can be calculated. If we alternately take the even and odd index terms of the input sequence, the algorithm called DIT (Decimation in Time), while, if we take the input sequence as the transform sequence, and we take the even term and the odd term alternately, then this algorithm is called DIF Decimation in Frequency. The dividing process is repeated $\log_2 N$ times and requires N multiplications each time. This gives the famous formula for the computational complexity of the FFT of ($N \log_2 N$) [3]. We can drive the Radix-2 equation DIT using the definition of DFT in eq (1) as follow:

$$X(k) = \sum_{even} x(n)W_N^{nk} + \sum_{odd} x(n)W_N^{nk}$$

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)W_{\frac{N}{2}}^{nk} \text{.....(3)}$$

Eq (3) will repeated until we have two-point DFT. we can drive the DIF equations using the same simple idea [3].

3. Radix-3 Algorithms

Radix-3 algorithms can be used if $N=3^i$, and basically the idea is the same as radix-2. The objective is to divide the transform to simple smaller ones. But the deferent here is the sequence will divided by 3, and we take the transform until we have just three points DFT. Here also we have, DIT and DIF algorithms which depend on application off course, and the deferent between them is the transform sequence that consider generally as multiple of three. The radix-3 equation decimation in time can be derived as follows [4]:

$$X(k) = \sum_{n=0}^{\frac{N}{3}-1} x(3n)W_N^{3nk} + \sum_{n=0}^{\frac{N}{3}-1} x(3n+1)W_N^{(3n+1)k} + \sum_{n=0}^{\frac{N}{3}-1} x(3n+2)W_N^{(3n+2)k}$$

Which simplify to:

$$(k) = x1_k + W_N{}^k x2_k + W_N{}^{2k} x3_k \quad \text{..... (4)}$$

Where x1, x2 and x3 are each N/3 DFT with the sequence (0,3, 6,...,N-3) and (1,4,7,...,N-2) and (2,5,8,...,N-1), Respectively. We repeat eq (4) until we have just three points DFT which considered the butterfly of radix-3.

4. Radix-4 Algorithms

When $N = 4^i$, we can always use radix-2 butt in this case, it's more efficient to use radix-4 algorithm. A radix-4 FFT is easily developed from the basic radix-2 structure by replacing the length-2 butterfly by a length-4 butterfly and making a few other modifications. Increasing the radix to 8 gives some improvement but not as much as radix-4. Increasing it to 16 is theoretically promising but the small decrease in multiplications is somewhat offset by an increase in additions and the program becomes rather long. Other radices are not attractive because they generally require a substantial number of multiplications and additions in the butterflies [5].

Like other algorithm here we also have DIT and DIF depending on how we divide an assign our data. The definition of DFT in eq (1) can be used to drive the equation of Radix-4 DIT as follow [6].

$$X(k) = \sum_{n=0}^{\frac{N}{4}-1} x(4n)W_N^{4nk} + \sum_{n=0}^{\frac{N}{4}-1} x(4n+1)W_N^{(4n+1)k} + \sum_{n=0}^{\frac{N}{4}-1} x(4n+2)W_N^{(4n+2)k} + \sum_{n=0}^{\frac{N}{4}-1} x(4n+3)W_N^{(4n+3)k}$$

$$X(k) = x1_k + W_N^k x2_k + W_N^{2k} x3_k + W_N^{3k} x4_k \text{.....(5)}$$

And we repeat eq (5) until we have DFT of 4 point, which in this case considere the building block and the butterfly of this algorithm.

5. Split-Radix FFT

The basic idea behind the split-radix FFT (SRFFT) is the combination of radix-2 and radix-4 algorithms together. Which yelled to the lowest number of multiplication and addition. It relates a Length-N DFT to one Length-N/2 DFT and two Length-N/4 DFT's. Repeating this process for the half and quarter length DFT's until scalars result gives the SRFFT algorithm in the same way the DIF radix-2 [7,8].

The resulting flow graph for the algorithm calculated in place looks like a radix-2 FFT. The L- shaped SRFFT butterfly. Advances the calculation of the top half, while the lower half, like a radix-4 butterfly. The algorithm is based on observation that at each stage a radix-4 is better for the odd DFT coefficients and a radix-2 is better for the even DFT coefficients. The algorithm is derived by using decimation in frequency DIT as follow: [6].

The N point DFT of $N=2^i$ can be decomposed as follow:

$$X(2k) \quad k=0,1,2,\ldots,N/2-1$$

$$X(4k+1) \text{ and } X(4k+3) \quad k=0,1,2,\ldots,N/4-1$$

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{2kn} + \sum_{n=\frac{N}{2}}^{N-1} x(n) W_N^{2nk}$$

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} [x(n) + x(n+\frac{N}{2})] W_{\frac{N}{2}}^{kn} \text{ where } k=0,1,2,\ldots N/2-1 \qquad \ldots\ldots \quad (6)$$

Here eq (6) is N/2 radix-2 DIF FFT. Similarly, the next Two part can be written as:

$$X(4k+1) = \sum_{n=0}^{\frac{N}{4}-1} ([x(n) - jx(n+\frac{N}{4}) - x\left(n+\frac{N}{2}\right) + jx(n+\frac{3N}{4})] W_N^n) W_{\frac{N}{4}}^{nk} \qquad (7)$$

$$X(4k+3) = \sum_{n=0}^{\frac{N}{4}-1} ([x(n) + jx(n+\frac{N}{4}) - x\left(n+\frac{N}{2}\right) - jx(n+\frac{3N}{4})] W_N^{3n}) W_{\frac{N}{4}}^{nk} \qquad (8)$$

Eq (7) and eq (8) are radix-4 DIF algorithm for k=0,1,2,...,N/4 -1. And we repeat eq (6,7,8) to further divide the transform to smaller part [9,10].

6. Sparse Fast Fourier Transform Applications:

There are some applications for SFT, and can be summarized as follows:

6.1. Multidimensional FFTs

The multidimensional (M-D) Fast Fourier Transforms (FFTs in 2D or more dimensions) are used in many applications such as image processing, applied physics. These applications require large amount of computations. The general form of the multidimensional FFT is as follows:

$$X(u_1, u_2, \ldots, u_m) = \sum_{v_1=0}^{N_1-1} \sum_{v_2=0}^{N_2-1} \cdots \sum_{v_m=0}^{N_m-1} W_{N_1}^{u_1 v_1} W_{N_2}^{u_2 v_2} \cdots W_{N_m}^{u_m v_m} x(v_1, v_2, \ldots, v_m) \qquad \ldots\ldots\ldots (9)$$

where $W_N^k = \exp(-N2\pi k j)$, $uk = 0, 1, \ldots, uk-1$; uk is the length of the kth dimension $k = 1, 2, \ldots, m$ and $x(v1, v2, \ldots, vm)$ are the complex input data sequences. Equation (10) is converted into m one-dimensional FFTs as follows:

$$X(u_1, u_2, \ldots, u_m) = \sum_{v_1=0}^{N_1-1} W_{N_1}^{u_1 v_1} \sum_{v_2=0}^{N_2-1} W_{N_2}^{u_2 v_2} \cdots \sum_{v_m=0}^{N_m-1} W_{N_m}^{u_m v_m} x(v_1, v_2, \ldots, v_m) \quad \ldots\ldots\ldots(10)$$

This provides the simplest algorithm where each one-dimensional FFT can be computed by the Cooley–Tukey FFT [11], or SFT algorithm. So this algorithm is known as row– column algorithm [12]. Several algorithms have been proposed for the multidimensional FFTs such as the Vector-Radix Algorithms (VRA), the polynomial transform algorithms and the Split Vector-Radix Algorithms (SVRA) [13]. These algorithms reduce the complexity over row– column algorithm. In [14], a fast algorithm has been derived based on vector coding for multidimensional integral points. This algorithm has reduced the multiplication complexity and the number of recursive stages without increasing the number of additions. However, the most popular one among these algorithms is the row–column decomposition algorithm, due to its simple structure and easy to program.

6.2. Spectrum Sensing and Decoding

The ever-increasing demand for wireless connectivity has led to a spectrum shortage which prompted the FCC to release multiple new bands for dynamic spectrum sharing. This is part of a bigger vision to dynamically share much of the currently under-utilized spectrum, creating a wide spectrum superhighways that can be shared by different types of wireless services. However, a major technical obstacle precluding this vision is the need for receivers that can capture and sense the GHz of spectrum in real-time in order to identify unoccupied bands quickly. Such receivers consume a lot of power because they need to sample the wireless signal at Giga Sample/sec. To overcome this challenge, we leverage the fact that the wireless spectrum is sparsely utilized and use the Sparse Fourier Transform to build a receiver that can capture and recover GHz of spectrum in real-time, while sampling only at Mega Sample/sec [15,16].

6.3. GPS Receivers

GPS is one of the most widely used wireless systems. In order to calculate its position, a GPS receiver has to lock on the satellite signals by aligning the received signal with each satellite's code. This process requires heavy computation, which consumes both time and power. As a result, running GPS on a phone can quickly drain the battery. GPS synchronization can be reduced into a sparse computation problem by leveraging the fact that only the correct alignment between the received GPS signal and the satellite code causes their correlation to spike. We can develop a GPS receiver that exploits the Sparse Fourier Transform to quickly lock on the satellite signal and identify its location. With significant reduction in localization delay and power consumption [17,18].

7. General Applications Uses FFT Algorithms:

There are some applications for FFT that can be summarized as follows:

7.1. Digital Signal Processing Applications

Spectral analysis of signals is one of the most important and core application FFT. By analyzing the spectra of input signals in frequency domain, the unknown parameters such as frequency, amplitude, power spectra, and phase parameters of a signal can be observed that are not easily detectable in time domain waveform [19].

7.2. Applications to Communication

FFT is an important functional block in modern communication systems, specifically for applications in Orthogonal Frequency Division Multiplexing (OFDM) systems, such as Digital Broadcasting [14], Worldwide Interoperability for Microwave Access (WiMAX) [20,21], IEEE 802.11 standards [22], Long-Term Evolution (LTE) [23].

7.3. Image Processing Application

FFT is used in medical imaging [24] for image filtering, image analysis and image reconstruction. In the Fourier representation of images using FFT, spectral magnitude, and phase tend to play different roles. In [25], Oppenheim and Lim demonstrated the importance of phase spectrum over magnitude spectrum. Further in [26], it is shown that the phase information plays a more important role than the magnitude. Some of the important applications based on the FFT-based image matching include face recognition, iris recognition, palm print recognition, fingerprint matching and

waveform matching. In [26, 27], a new image quality assessment algorithm based on the phase and magnitude of the two dimensional DFT has been proposed, which pointed out that both amplitude and phase spectrum are required for perfect image reconstruction.

Fourier Transform in the West (FFTW), which contain a FFT library (C codes) is used for computing the DFT in 1- D or more dimensions of various input size for both real- and complex-valued input data. Image processing applications make use of FFTW. However, today's applications need to process a huge amount of data sets demands fast computation of FFT. The Sparse FFT algorithm [28] addresses this problem by providing a sublinear complexity which utilizes the sparsity in the Fourier domain (considering only frequencies with information). The fast computation makes the sparse FFT a promising tool for many data-intensive applications such as 4D light fields [29] and 2D Magnetic Resonance Spectroscopy (MRS) [30].

8. Sparse Fast Fourier Transform (SFT) [15,31]

All the algorithm previously mentioned depends on fundamental idea which is, dividing the big transform to smaller ones that we can compute separately. And since the sixty's this was the idea to the method of computing the DFT, and it was adopted since then in all the applications that need the computation of the DFT. This FFT's algorithm are efficient and fast if the number of sample are relatively low, but when N become very large even these algorithms become inefficient and slow. So because of the bad performance of the FFT's algorithm at large N, it's necessary to develop better and faster algorithm. And this why Sparse FFT (SFT) was developed. It basically uses the sparsity of the signal in clever way to reduce the computation.

The meaning of sparse signal is that if we use eq (1) to compute the DFT most of the value are either zero or close to zero. So SFR algorithm is use this fact to compute the DFT of the signal from subset of the input sequence. The algorithm framework is as follow:

8.1. Frequency Bucketization

The Sparse Fourier Transform starts by hashing the frequency coefficients of x(n) into buckets such that the value of the bucket is the sum of the values of the frequency coefficients that hash into the bucket. Since x(n) is sparse, many buckets will be empty and can be simply discarded. The algorithm then focuses on the nonempty buckets and computes the positions and values of the large frequency coefficients in those buckets in what we call the frequency estimation step. The process of frequency bucketization is achieved through the use of filters. A filter suppresses and zeroes out frequency coefficients that hash outside the bucket while passing through frequency coefficients that hash into the bucket. The simplest example of this is the aliasing filter. Recall the following basic property of the Fourier transform: subsampling in the time domain causes aliasing in the frequency domain.

8.2. Frequency Estimation

In this step, the Sparse Fourier Transform estimates the positions and values of the non-zero frequency coefficients which created the energy in each of the non-empty buckets. Since x(n) is sparse, many of the non-empty buckets will likely have a single non-zero frequency coefficient hashing into them, and only a small number will have a collision of multiple non-zero coefficients. We first focus on buckets with a single non-zero frequency coefficients and estimate the value and the position of this non-zero frequency, i.e., X(k) and the corresponding k. In the absence of a collision, the value of the non-zero frequency coefficient is the value of the bucket it hashes to since all other frequencies that hash into the bucket have zero values. Hence, we can easily find the value of the non-zero frequency coefficient in a bucket. However, we still do not know its frequency position k, since frequency bucketization mapped multiple frequencies to the same bucket. The simplest way to compute k is to leverage the phase-rotation property of the Fourier transform, which states that *a shift in time domain translates into phase rotation in the frequency domain*. Specifically, we perform the process of bucketization again, after a circular shift of x by τ samples. Since a shift in time translates into a phase rotation in the frequency domain, the value of the bucket changes Hence, using the change in the phase of the bucket, we can estimate the position of the non-zero frequency coefficient in the bucket.

8.3. Collision Resolution

Non-zero frequency coefficients that are isolated in their own bucket can be properly estimated as described above. However, when non-zero frequencies collide in the same bucket, we are unable to estimate them correctly. Hence, to recover the full frequency spectrum, we need to resolve the collisions. To resolve collision, we need to repeat the frequency bucketization in a manner that ensures the same nonzero frequencies do not collide with each other every time. The manner in which we achieve this depends on the type of filter used for bucketization. For example, with the

aliasing filters described above, we can bucketize the spectrum multiple times using aliasing filters with co-prime sampling rates. Co-prime aliasing filters guarantee that any two frequencies that collide in one bucketization will not collide in the other bucketization. Iterating between the different bucketizations is done by estimating the frequencies from buckets where they do not collide and subtracting them from buckets where they do collide, also ensuring that each non-zero frequency will be isolated in its own bucket during some iteration of the algorithm. This allows us to estimate each non-zero frequency correctly. Thus, at the end of the of the collision resolution step, we have recovered all non-zero frequencies and hence have successfully computed the Fourier transform of the signal.

9. Conclusion

DFT is very important transform in Digital signal processing, and it has many applications in different areas. So it's desirable to find efficient way to compute it. The paper starts with the idea of basic FFT algorithms, which do well when N is not very large. For large N the old FFT algorithms become inefficient so we introduced the SFT which assume sparse input signal and compute the DFT from subset of the input signal. This will outperform the old FFT's and finally we introduce the multidimensional DFT which simply can convert it to multi one dimension FFT and all one Dimension algorithms can be applied to compute the transform.

Recommendations

In the research of this field of Digital Signal Processing (DSP), it is favourable to begin the study with DFT first, secondly we study the FFT algorithms, and why they used instead of DFT. Finally we can study the SFT and Multidimensional FFT with it effect on time domain and frequency domain.

Compliance with ethical standards

Disclosure of conflict of interest

The author declares that there is no conflict of interest, either current or past, with any affiliated institutions.

Declaration

This work was conducted independently as part of the author's research activities. All results, proposals, and findings are based on the cited literature. The paper is done as a part of independent research. Here, all results, proposals and findings are from the cited literature.

References

[1] B. G. Osgood, Lectures on the Fourier Transform and Its Applications, American Mathematical Society, 2019.

[2] R.M. Jiang, "An area-efficient FFT architecture for OFDM digital video broadcasting", IEEE Trans. Consum. Electron. Vol. (**53) No.** (4), 2007, pp. 1322–1326.

[3] A. Oppenheim, R. Schafer, Discrete-Time Signal Processing, Pearson; 3rd edition 2009.

[4] K. R. Rao, D. Nyeon Kim and J. Jeong Hwang, Fast Fourier Transform - Algorithms and Applications, Springer, 2010.

[5] D. Manolakis, J. G Proakis, Digital Signal Processing: Principles, Algorithms, and Applications, Pearson; 4th edition 2006.

[6] C. S. Burrus and T. W. Parks, "DFT/FFT and Convolution Algorithms and Implementation", Wiley-Interscience; 1st edition, 1985.

[7] C. M. Rader. "Discrete Fourier transforms when the number of data samples is prime". Proceedings of the IEEE, Vol. (56), 1968.

[8] P. Duhamel and H. Hollmann, "Split Radix FFT Algorithm," Electronic Letters, Vol 20, 1984, pp 14-16.

[9] P. Duhamel and M. Vetterli, "Fast Fourier transforms A tutorial review and a state of the art Signal Process", Elsevier, Vol. (19), 1990, pp.259–299.

[10] E. Brigham, Fast Fourier Transform and Its Applications, Pearson; 1st ed. 1988.

[11] F. Cooley, J. Tukey, "An algorithm for the machine calculation of complex Fourier series", Math. Comput., **Vol. (19) No.** (90), 1965, pp. 297–301.

[12] C. Van Loan, Computational Frameworks for the Fast Fourier Transform, SIAM, Philadelphia, 1992.

[13] M. Clellan, D. Chan and H. Schuessler, "Vector radix fast Fourier transform", in IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, 1977, pp. 548–55.

[14] Z. Chen, L. Zhang, "Vector coding algorithms for multidimensional discrete Fourier transform", J. Comput. Appl. Math, **Vol. (212**), No. (1), 2008, pp. 63–74.

[15] H. Al-Hassanieh, "The Sparse Fourier Transform: theory & practice", thesis submitted to Massachusetts Institute of Technology department of electrical engineering and computer science; MIT, 2016.

[16] M. Turrillas, A. Cortés, I. Vélez, J. F. Sevillano and A. Irizar, "An FFT core for DVB-T2 receivers", 16th IEEE International Conference on Electronics, Circuits, and Systems, 2009, pp. 120–123.

[17] M. Garrido, S. J. Huang, S. G. Chen and O. Gustafsson, "The serial commutator FFT", IEEE Trans. Circuits Syst. Vol. (**63), No.** (10), 2016, pp.974–978.

[18] M. J. Narasimha, "Modified overlap-add and overlap-save convolution algorithms for real signals", IEEE Signal Process. Lett., Vol. (**13**), No. (11), 2006, pp.669–671.

[19] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications ACM, Vol. (21), No. (2), 1978, pp. 120–126.

[20] Cortés, I. Vélez, J. Sevillano and M. Turrillas, "Fast Fourier Transform Processors: Implementing FFT and IFFT Cores for OFDM Communication Systems", INTECH, London, 2012.

[21] P. Fan, M. S. Lee and G. A. Su, "A low multiplier and multiplication costs 256-point FFT implementation with simplified radix-24 SDF architecture", IEEE Asia Pacific Conference on Circuits and Systems", APCCAS , 2006, pp. 1935–1938.

[22] T. Cho, H. Lee, J. Park, C. Park, "A high-speed low-complexity modified radix-25 FFT processor for gigabit WPAN applications", IEEE International Symposium on Circuits and Systems, (2011), pp. 1259–1262.

[23] S. Y. Peng, K. T. Shr, C. M. Chen and Y. H. Huang, "Energy-efficient 128 2048/1536-point FFT processor with resource block mapping for 3GPP-LTE system", International Conference on Green Circuits and Systems, IEEE, 2010, pp. 14–17.

[24] M. N. Haque, M. S. Uddin, M. Abdullah-Al-Wadud, and Y. Chung, "Fast reconstruction technique for medical images using graphics processing unit", Signal Processing, Image Processing and Pattern Recognition conference, 2011, pp. 300–309.

[25] A. Oppenheim, J. S. Lim, "The importance of phase in signals", IEEE, Vol. (69), No. (5), 1981, pp.529–541.

[26] X. S. Ni and X. Huo, "Statistical interpretation of the importance of phase information in signal and image reconstruction", Stat. Probab. Lett. Vol. (77), No. (4), 2007, pp. 447–454.

[27] M. Narwaria, W. Lin, I. V. McLoughlin, S. Emmanuel and L. T. Chia, "Fourier transform-based scalable image quality measure", IEEE Trans. Image Process., Vol. (21), No. (8), 2012, pp.3364–3377.

[28] 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.

[29] W. W. Smith and J. M. Smith, "Handbook of Real-Time Fast Fourier Transforms", IEEE Press, NewYork, 1995.

[30] L. P. Yaroslavsky, "Fast transforms in image processing: compression, restoration, and resampling", Adv. Electr. Eng., 2014.

[31] H. Hassanieh, "Simple and Practical Algorithm for Sparse Fourier Transform", Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, 2011.