Interesting Open Problem Related to Complexity of Computing the Fourier Transform and Group Theory

Nir Ailon
CS, Technion
nailon@cs.technion.ac.il

February 4, 2019

Abstract

The Fourier Transform is one of the most important linear transformations used in science and engineering. Cooley and Tukey's Fast Fourier Transform (FFT) from 1964 is a method for computing this transformation in time $O(n \log n)$. From a lower bound perspective, relatively little is known. Ailon shows in 2013 an $\Omega(n \log n)$ bound for computing the normalized Fourier Transform assuming only unitary operations on pairs of coordinates is allowed. The goal of this document is to describe a natural open problem that arises from this work, which is related to group theory, and in particular to representation theory.

1 Introduction

The (discrete) normalized Fourier transform is a complex linear mapping sending an input $x \in \mathbb{C}^n$ to $y = Fx \in \mathbb{C}^n$, where F is an $n \times n$ unitary matrix defined by

$$F(k, \ell) = n^{-1/2} e^{-i2\pi k\ell/n} \ . \tag{1}$$

If n is a power of 2, then Walsh-Hadamard transform is a real, orthogonal mapping H, with the element in position (k, ℓ) given by:

$$H(k, \ell) = n^{-1/2}(-1)^{\langle k,\ell \rangle} \ , \tag{2}$$

where $\langle k, \ell \rangle$ is the dot-product modulo 2 of the binary representations of the integers $k - 1$ and $\ell - 1$.

More genererally, both F and H (resp.) are defined by the characters of underlying abelian groups $\mathbb{Z}/n\mathbb{Z}$ (integers modulo n, under addition) and the $\log n$ dimensional binary cube $(\mathbb{Z}/2\mathbb{Z})^{\log n}$ ($\log n$ dimensional vector space over bits). Given an input vector $x \in \mathbb{C}^n$, it is possible to compute Fx and Hx (resp.) in time $O(n \log n)$ using the Fast Fourier-Transform [Cooley and Tukey(1964)] , or the Walsh-Hadamard transform (resp.).

As for computational lower bounds, it is trivial that computing both Fx and Hx requires a linear number of steps, because each coordinate of the output depends on all the input coordinates.

There has not been much prior work on better bounds. We refer the reader to [Ailon(2013)] for a brief history of this line of work, and concentrate on a recent lower bound.

The work [Ailon(2013)] provides a lower bound of $\Omega(n \log n)$ operations for computing Fx (or Hx) given x, assuming that at each step the computer can perform a unitary operation affecting at most 2 rows. In other words, the algorithm, running in m steps, is viewed as a product

$$R_m \cdot R_{m-1} \ldots R_2 \cdot R_1$$

of matrices R_t, each R_t a block-diagonal matrix with $n-2$ blocks equalling 1, and one block equalling a 2×2 unitary matrix A_t:

$$R_t = \begin{array}{c} \\ \\ \\ i_t \\ \\ \\ \\ j_t \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & & & i_t & & & & j_t & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & A_t(1,1) & & & A_t(1,2) & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & A_t(2,1) & & & A_t(2,2) & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{array} \end{array} . \tag{3}$$

The justification for this model of computation is threefold:

- Similarly to matrices of the form (3), any basic operation of a modern computer (e.g., addition of numbers) acts on only a fixed number of inputs.
- The Fast Fourier-Transform, as well as the Walsh-Hadamard transform, operate in this model, and
- The set of matrices of the form (3) generate the group of unitary matrices.

Thus the question of computational complexity of of the Fourier transform becomes that of computing distances between elements of a group, namely the unitary group, with respect to a set of generators that is computationally simple.

Obtaining the lower bound of $\Omega(n\log n)$ in [Ailon(2013)] is done by defining a potential function Φ for unitary matrices, as follows:

$$\Phi(U) = -\sum_{i,j} |U(i,j)|^2 \log |U(i,j)|^2 \ .$$

With this potential function, one shows that:

(a) $\Phi(\mathrm{Id}) = 0$

(b) $\Phi(F) = \Phi(H) = \Omega(n\log n)$

(c) $|\Phi(M_t) - \Phi(M_{t-1})| \le 2$, where $M_t = R_t \cdot R_{t-1} \ldots R_1$ is the *state* of the algorithm after t steps.

Indeed, if the potentail Φ grows from 0 to $\Omega(n\log n)$ changing (in absolute value) by no more than 2 at each step, then the number of steps must be $\Omega(n\log n)$. Showing (c) is done using two observations. The first is that M_t defers from M_{t-1} in at most 2 rows i_t and j_t, and that for each column k, due to unitarity of A_t,

$$|M_t(i_t,k)|^2 + |M_t(j_t,k)|^2 = |M_{t-1}(i_t,k)|^2 + |M_{t-1}(j_t,k)|^2 \ .$$

The next observation is that any 4 numbers x,y,z,w satisfying $x^2+y^2 = z^2+w^2 =: r^2$, also satisfy

$$|(x^2\log x^2 + y^2 \log y^2) - (z^2\log z^2 + w^2\log w^2)| \le r^2 \ .$$

Combining the observations, we conclude that the total change in the potential function can be at most

$$\sum_k (|M_t(i_t,k)|^2 + |M_t(j_t,k)|^2) = 2 \ .$$

2 A Perplexing Problem

The advantage of the method just described is that it reduces a computational problem to that of computing distance between two elements of a group, with respect to a chosen set of generators of the group. We now define a more general problem within the same group theoretical setting. Consider the $2n \times 2n$ matrix G defined as

$$G = \begin{pmatrix} 0 & F \\ -F^* & 0 \end{pmatrix} \ .$$

(One may replace F with H, but we work with F henceforth.) The matrix G is skew-Hermitian. Let Id denote the $2n \times 2n$ identity matrix, and finally define for a real angle α the following matrix:

$$X_\alpha = (\cos\alpha)\,\mathrm{Id} + (\sin\alpha)G \ .$$

It is easy to verify that X_α is unitary for all α. It is also easy to verify that

$$X_{\alpha'} X_\alpha = X_{\alpha+\alpha'} \ . \tag{4}$$

Also, using the potential function Φ defined above, we see that

$$\Phi(X_\alpha) = \Theta(\alpha^2 n \log n) \ .$$

Hence, using the argument as above, the number of steps required to compute X_α must be at least $\Omega(\alpha^2 n \log n)$. However, it is unreasonable that it should be possible to compute X_α faster than the time it takes to compute F, by a factor of $1/\alpha^2$. Indeed, given an input $x \in \mathbb{C}^n$, we could simply embed it as $\tilde{x} \in \mathbb{C}^{2n}$ by padding with n 0's, then compute $\tilde{y} = X_\alpha \tilde{x}$ and then retrieve $y = Fx$ from $\tilde{y}$ and $\tilde{x}$ by a simple arithmetic manipulation. Hence, we conjecture that the number of steps required to compute X_α should be not much smaller than $\Omega(n \log n)$. [1]

2.1 A slight improvement: Lower bound of $\Omega(\alpha n \log n)$.

It is possible to get a better bound than $\Omega(\alpha^2 n \log n)$, as follows. Instead of starting the computation at state Id and finishing at X_α, we can opportunistically choose a starting point M (and finish at $X_\alpha M$).

If we choose the state $M = X_{\pi/4-\alpha/2}$ then it is trivial to verify that the computation ends at state $X_\alpha M$, which equals $X_{\pi/4+\alpha/2}$ by (4). We then observe that

$$\Phi(X_{\pi/4-\alpha/2}) = \Theta\left(\sin^2(\pi/4-\alpha/2)\cdot n\log n\right) \qquad \Phi(X_{\pi/4+\alpha/2}) = \Theta\left(\sin^2(\pi/4+\alpha/2)\cdot n\log n\right) \ ,$$

and hence,

$$\begin{aligned} \left|\Phi(X_{\pi/4+\alpha/2}) - \Phi(X_{\pi/4-\alpha/2})\right| &= \Omega\left((\sin^2(\pi/4+\alpha/2) - \sin^2(\pi/4-\alpha/2))n\log n\right) \\ &= \Omega\left(\alpha n \log n\right) \ . \end{aligned}$$

[1] The author conjectures $\Omega((n\log n)/\log(1/\alpha))$ to be the correct bound.

2.2 Stronger improvements?

It is tempting take the following approach. Assume α is small (say, in the range $(0, \pi/4)$), and also that $\pi/2$ is divisible by α. Then by (4),

$$X_\alpha^{\pi/(2\alpha)} = X_{\pi/2} = G \ .$$

Viewing X_α as an operator, the last equation tells us that X_α composed with itself $\pi/(2\alpha)$ times gives G. Therefore, using the "squaring trick" should imply that the number of steps to compute G is at at most $O(\log(1/\alpha))$ times the time it takes to compute X_α. But since we need $\Omega(n \log n)$ steps to compute G, we conclude that we need at least $\Omega((n \log n)/\log(1/\alpha))$ steps to compute X_α. [2] However, it turns out that the squaring trick doesn't work here, because we are not computing matrices and their exponents, but rather their action on a given input vector.

Is it possible to get a stronger lower bound than $\alpha n \log n$? One approach for solving this problem might be using group representation theory. If $\Psi : U(n) \mapsto U(n')$ is any unitary representation of $U(n)$, then we could define a new potential function $\Phi \circ \Psi$ on $U(n)$, and use it to obtain possibly better lower bounds.

A simple representation that I've tried is the tensor representation $\Psi : U(n) \mapsto U(n^2)$ which can be written, using a coordinate system indexed by pairs (i_1, i_2), as

$$(\Psi U)((i_1, i_2), (j_1, j_2)) = U(i_1, j_1)U(i_2, j_2) \ .$$

Unfortunately this representation doesn't give us anything because it can be easily shown that $\Phi \circ \Psi$ is simply $2n\Psi$, in other words, the potential function is simply multiplied by $2n$.

Are there other more interesting representations of $U(n)$ that could give something more useful? I have tried various standard representations derived from matrix tensors, but they have not given any interesting bound. For more details, please contact me.

References

[Ailon(2013)] Nir Ailon. A lower bound for Fourier transform computation in a linear model over 2x2 unitary gates using matrix entropy. *Chicago J. of Theo. Comp. Sci.*, 2013.

[Cooley and Tukey(1964)] J. W Cooley and J. W Tukey. An algorithm for the machine computation of complex Fourier series. *J. of American Math. Soc.*, pages 297–301, 1964.

[2] The "squaring trick" is what we use to 'raise to the power k', using $O(\log k)$ iterations.