Walsh-Hadamard Transform Notes

Jeremy Johnson
Department of Computer Science
Drexel University
Philadelphia, PA 19104
jjohnson@cs.drexel.edu

August 12, 2005

1 The Walsh-Hadamard Transform

The Walsh-Hadamard transform of a signal $x$, of size $N = 2^n$, is the matrix-vector product $\text{WHT}_N \cdot x$, where

$$\text{WHT}_N = \bigotimes_{i=1}^{n} \text{DFT}_2 = \text{DFT}_2 \otimes \cdots \otimes \text{DFT}_2.$$ 

The matrix

$$\text{DFT}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

is the 2-point DFT matrix, and $\otimes$ denotes the tensor or Kronecker product. The tensor product of two matrices is obtained by replacing each entry of the first matrix by that element multiplied by the second matrix. Thus, for example,

$$\text{WHT}_4 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$ 

Algorithms for computing the WHT can be derived using properties of the tensor product [2, 1]. A recursive algorithm for the WHT is obtained from the factorization

$$\text{WHT}_{2^n} = (\text{WHT}_2 \otimes I_{2^{n-1}})(I_2 \otimes \text{WHT}_{2^{n-1}}) \quad (1)$$

This equation corresponds to the divide and conquer step in a recursive FFT. An iterative algorithm for computing the WHT is obtained from the factorization

$$\text{WHT}_{2^n} = \prod_{i=1}^{n} (I_{2^{i-1}} \otimes \text{WHT}_2 \otimes I_{2^{n-i}}), \quad (2)$$
which corresponds to an iterative FFT. More generally, let  
\( n = n_1 + \cdots + n_t \), then

\[
\text{WHT}_{2^n} = \prod_{i=1}^{t} (I_{2^{n_1} + \cdots + n_{i-1}} \otimes \text{WHT}_{2^{n_i}} \otimes I_{2^{n_{i+1}} + \cdots + n_t})
\]  

(3)

This equation encompasses both the iterative and recursive algorithm and provides a mechanism for exploring different breakdown strategies and combinations of recursion and iteration. Alternative algorithms are obtained through different sequences of the application of Equation 3. Each algorithm obtained this way can be represented by a tree, called a composition tree. The root of the composition tree corresponding to an algorithm for computing \( \text{WHT}_N \), where \( N = 2^n \) is labeled with \( n \). Each application of Equation 3 corresponds to an expansion of a node into children whose sum equals the node. Figure 1 shows the trees for a right-recursive and iterative algorithm for computing \( \text{WHT}_{16} \).

![Partition Trees for Iterative and Recursive WHT Algorithms](image)

A fully expanded composition tree is a composition tree whose leaf nodes are equal to 1.

**Theorem 1** Let  
\( n = n_1 + \cdots + n_t \), then

\[
\text{WHT}_{2^n} = \prod_{i=1}^{t} (I_{2^{n_1} + \cdots + n_{i-1}} \otimes \text{WHT}_{2^{n_i}} \otimes I_{2^{n_{i+1}} + \cdots + n_t})
\]

Proof.

First observe, using associativity of \( \otimes \), that

\[
\text{WHT}_{2^n} = \text{DFT}_2 \otimes \cdots \otimes \text{DFT}_2 \otimes \cdots \otimes \text{DFT}_2 = \text{WHT}_{2^{n_1}} \otimes \cdots \otimes \text{WHT}_{2^{n_t}}
\]

For fixed, but arbitrary, \( n \) we prove the theorem using induction on \( t \). In the base case, when \( t = 1 \), the theorem is trivially true provided we interpret the empty sums \( n_1 + \cdots + n_0 = 0 \) and \( n_2 + \cdots + n_1 \) to be zero. In this case \( I_{2^n} = 1 \) and \( 1 \otimes A = A = A \otimes 1 \).

Assume that the theorem is true for \( t - 1 \). The inductive proof proceeds using the property \( A \otimes B = (A \otimes I)(I \otimes B) \). Using this property,

\[
\text{WHT}_{2^n} = (\text{WHT}_{2^{n_1}} \otimes I_{2^{n_2} + \cdots + n_t}) (I_{2^{n_1}} \otimes \text{WHT}_{2^{n_2} + \cdots + n_t}),
\]
and using the inductive hypothesis on $\text{WHT}_{2^n+i}$,

$$\text{WHT}_{2^n} = (\text{WHT}_{2^n_1} \otimes I_{2^n_2+\ldots+n_t}) \left( I_{2^n_1} \otimes \prod_{i=2}^{t} (I_{2^n_2+\ldots+n_{i-1}} \otimes \text{WHT}_{2^n_i} \otimes I_{2^n_{i+1}+\ldots+n_t}) \right)$$

Using the tensor product properties $(I \otimes AB) = (I \otimes A)(I \otimes B)$ and $I_m \otimes I_n = I_{mn}$,

$$\text{WHT}_{2^n} = (\text{WHT}_{2^n_1} \otimes I_{2^n_2+\ldots+n_t}) \prod_{i=2}^{t} (I_{2^n_2+\ldots+n_{i-1}} \otimes \text{WHT}_{2^n_i} \otimes I_{2^n_{i+1}+\ldots+n_t}),$$

which is equal to

$$\text{WHT}_{2^n} = \prod_{i=1}^{t} (I_{2^n_2+\ldots+n_{i-1}} \otimes \text{WHT}_{2^n_i} \otimes I_{2^n_{i+1}+\ldots+n_t})$$

\section{Computation of the WHT}

Let $N = N_1 \cdots N_t$, where $N_i = 2^{n_i}$, and let $x_{b,s}^{M_i}$ denote the vector $(x(b), x(b+s), \ldots, x(b+(M-1)s))$. Then evaluation of $x = \text{WHT}_N \cdot x$ using Equation 3 is performed using

$$R = N; \quad S = 1;$$

for $i = 1, \ldots, t$

$$R = R/N_i;$$

for $j = 0, \ldots, R - 1$

for $k = 0, \ldots, S - 1$

$$x_{jN_i, S+k, S}^{N_i} = \text{WHT}_N \cdot x_{jN_i, S+k, S}^{N_i};$$

$$S = S * N_i;$$

This scheme assumes that the algorithm works in-place and is able to accept stride parameters. Moreover, the recursive computations of $\text{WHT}_{N_i}$ are performed using the same triply-nested loop. In the base case straight-line code is used to evaluate $\text{WHT}_{N_i}$. The recursive computation structure is guided by a specified composition tree. If the tree is fully expanded all base case computations correspond to $\text{DFT}_2$.

The computation of $\text{WHT}_N$ using any fully-expanded composition tree requires exactly $N \log N$ arithmetic operations. It is easy to check this for right recursive composition trees. Let $M(n)$ be the number of arithmetic operations required to compute $x = \text{WHT}_N$, where $N = 2^n$, using a right-recursive composition tree. In this case the number of arithmetic operations satisfies the recurrence relation

$$M(n) = \begin{cases} 
2M(n-1) + 2^n & n > 1 \\
2 & n = 1 
\end{cases}$$

and $M(n) = n2^n$. 

3
Theorem 2 All algorithms to compute $x = \text{WHT}_N x$, where $N = 2^n$, using fully expanded composition trees require exactly $N \lg N$ arithmetic operations.

Proof. The proof uses induction on $n$. In the base case, $n = 1$, since there are two arithmetic operations to compute $\text{WHT}_2$, the theorem is true.

In the general case, assume $\text{WHT}_N$ is computed using $n = n_1 + \cdots + n_t$ in Equation 3, and assume that the theorem is true for all positive integers less than $n$.

In the evaluation of

$$x = (I_{2^{n_1}} \otimes \cdots \otimes I_{2^{n_{t-1}}} \otimes \text{WHT}_{2^{n_t}} \otimes I_{2^{n_{t+1}}}) x$$

$\text{WHT}_{2^{n_t}}$ is evaluated $2^n - n_t$ times. Therefore, the number of arithmetic operations in the evaluation of Equation 3 is equal to

$$\sum_{i=1}^t 2^{n-n_i} M(n_i)$$

where $M(n_i)$ is equal to the number of operations required to evaluate $\text{WHT}_{2^{n_i}}$. By induction this is equal to

$$\sum_{i=1}^t 2^{n-n_i} n_i 2^{n_i},$$

which is equal to

$$\sum_{i=1}^t n_i 2^n = 2^n \sum_{i=1}^t n_i = n 2^n.$$

References
