

For Research Purposes Only

On Automatic Derivation of Fast Hadamard Transform Using Genetic Programming

S. Samadi, Y. Suzukake and H. Iwakura

Department of Communications and Systems
The University of Electro-Communications, Tokyo, JAPAN

Abstract

Automatic generation of fast Hadamard transform (FHT) algorithms through genetic programming is studied. The design of FHT algorithms is viewed as a search in the space of linear algorithms employing restricted types of coefficients. A generic circuit that can realize all algorithms belonging to this space is proposed. The goal is to automatically find an algorithm that can perform Hadamard transform properly using a specified number of multiply/add operations. It is shown that genetic programming is able to discover a 4-point FHT that requires only 7 multiply/add operations.

1 Introduction

Development of algorithms in the field of digital signal processing has been generally performed by human designers. The mathematical foundations of digital signal processing provide useful tools when a designer searches for an algorithm to perform a fast orthogonal transform or filter out noise from a signal. However, the mathematical tools are limited, applications and requirements are numerous, and the space to be searched for possible algorithms is so vast. Furthermore, emergence of novel computational environments and stricter efficiency requirements can make the development process a very complex and time-consuming task. Considering the preceding argument, it is quite attractive to study the possibility of automatic design of signal processing algorithms. Many approaches to automatic design have been concentrated on lower level issues related to hardware or software implementation of human-designed higher-level algorithms. Our interest lies in investigating the possibility of generating algorithms at higher levels in an automatic manner.

In this paper we are concerned with the automatic design of fast algorithms to perform 1-D Hadamard transform (HT) [1] using a given number of multiply/add operations. We view the design task as a search in the space of algorithms that can potentially solve our design problem. These are linear algorithms

that utilize a predefined set of coefficients and a specified number of multiply/add operations.

The following basic preparatory steps should be taken in developing an automatic design environment for fast linear algorithms including FHT. The first step is the determination of the computational resources available for implementation of the algorithm. We then need an algorithm representation method capable of incorporating the available resources. An important consideration in adoption of a representation method is the intended automatic computer search method. Mathematical properties of the problem also affect the selection of the representation method. For example, matrix and vector representations are natural and efficient methods for digital signal processing algorithms. We propose a matrix representation for FHTs that incorporates fused multiply/add operations. This matrix representation corresponds to a generic circuit for FHT algorithms.

Our search in the space of HT algorithms is mainly guided by the level of success of prospective algorithms in solving the design problem. This is needed to minimize the requirement of human knowledge for derivation of the algorithms. To this end, we adopt evolutionary algorithms as the search method. Evolutionary search methods have been successful in finding solutions to problems in optimization, artificial intelligence, machine learning, control, and many other areas of engineering. Genetic algorithms are a well-known example of evolutionary algorithms [11, 12]. For problems like ours, where a suitable representation is available and the computational complexity of evaluating a candidate in the space of possible solutions is affordable, an evolutionary algorithm is an attractive choice. In this paper we conduct the evolutionary search using genetic programming (GP) [13], a technique for automatic synthesis of computer programs coded as LISP S-expressions. An important feature of GP is the flexibility of its chromosome structure which is represented by one or more labeled trees. The problem of designing FHT algorithms through evolutionary methods is also considered in [14], where genetic algorithms are investigated. Ref. [14] is concerned with automatic derivation of in-place transforms using addition and subtraction operations.

For Research Purposes Only

The organization of this paper is as follows. After providing a brief background on HTs in Section 2, we state the design problem and give a representation scheme for FHTs in Section 3. Coding of FHT algorithms as tree structures for use in GP is discussed in Section 4, followed by simulation results in Section 5. Finally, Section 6 concludes the paper.

2 Theoretical Background

By the term Hadamard transform is meant any transformation of an $N \times 1$ vector \mathbf{x}_N by an $N \times N$ matrix \mathbf{H}_N with elements -1 and $+1$ satisfying

$$\mathbf{H}_N \mathbf{H}_N^T = N \mathbf{I}_N, \quad (1)$$

where \mathbf{I}_N is the identity matrix of order N . The vector of transform coefficients \mathbf{y}_N is related to the signal \mathbf{x}_N as

$$\mathbf{y}_N = \mathbf{H}_N \mathbf{x}_N. \quad (2)$$

Construction of \mathbf{H}_N for arbitrary values of N that are divisible by 4 is not a trivial task [2]. However, Sylvester-type Hadamard matrices with $N = 2^n$ can be easily constructed using the simple procedure

$$\mathbf{H}_{2^n} = \underbrace{\mathbf{H}_2 \otimes \cdots \otimes \mathbf{H}_2}_{n \text{ times}}, \quad (3)$$

where

$$\mathbf{H}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \quad (4)$$

The existing algorithms for FHT compute \mathbf{y}_N using $N \log_2 N$ additions [4, 5, 6]. They are in-place algorithms differing in the final ordering of transform coefficients and do not make use of multiplications. Lack of multiplications may be considered as an attractive feature of these algorithms. However, in some computational environments it may be more beneficial to further reduce the number of additions at the cost of allowing simple multiplications [8] or making use of the coherence in the structure of signal [7].

It is generally difficult to determine the minimum number of additions necessary to compute a linear algorithm for arbitrary data vectors [10]. Morgenstern has shown that there is a lower bound for the number of necessary additions, provided that an upper bound c can be assigned to the modulus of the coefficients involved in the algorithm [9]. The bound is inversely proportional to c . This suggests that faster HT algorithms may be obtained by allowing multiplier coefficients with magnitudes larger than unity. An example of this can be found in [8] where the number of additions is reduced by allowing multiply/add operations.

3 Statement of Problem

The general form of the FHT design problem is stated below.

Problem 1 (General Form) *For a given \mathbf{H}_N , find an algorithm to compute \mathbf{y}_N using the minimum possible number of additions.*

Searching for a solution to the above problem is a hard task even for small values of N . This is in part due to the fact that the only available measure of optimality of a prospective solution is to compare it with the lower bound given in [9]. Therefore, even with a very efficient search algorithm and small values of N , it is difficult, if not impossible, to verify whether an automatically generated HT satisfies the minimality requirement of Problem 1. In this paper, we concentrate on more practical forms of the problem.

Problem 2 (Practical Form) *Find an algorithm that yields an HT of \mathbf{x}_N using A multiply/add operations. The multiplier coefficients should be selected from \mathcal{C} , where \mathcal{C} denotes the set of admissible coefficients.*

Note that we have not specified the actual form of \mathbf{H}_N in Problem 2 and any type of HT is acceptable as long as the number of operations is at most A . There is no guarantee that an HT exists for an arbitrary value of A . The actual value of A should be determined according to the computational requirements and the values for existing FHT algorithms. As mentioned in Section 1, a representation method is needed to incorporate the computational requirements. Also note that storage requirements are not specified and thus we can search in a wider space than the space of conventional in-place algorithms.

We propose the use of a representation of FHTs that incorporates the requirements of fused multiply/add operations. This representation is used in [9] to characterize the family of linear algorithms. As a simple example for this representation consider the circuit of Fig. 1. At each stage, a single multiply/add operation of the form $X_p + mX_q$ ($m \in \mathcal{C}$) is performed on a pair of signals X_p and X_q that are selected from the available input signals. Therefore the number of stages is equal to the total number of operations. Furthermore, this circuit can represent all linear algorithms that utilize 3 multiply/add operations. For example, if we set the number of stages to 8, and the number of input signals to 4, the conventional 4-point, in-place FHT algorithms can be realized.

The matrix transfer function of the generic circuit of Fig. 1 is of the form

$$\mathbf{G}_2 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{S}_3^{(2)} \mathbf{S}_2^{(2)} \mathbf{S}_1^{(2)}, \quad (5)$$

where

$$\mathbf{S}_i^{(N)} = \begin{pmatrix} \mathbf{I}_{N+i-1} \\ \mathbf{v}_{N+i-1} \end{pmatrix}. \quad (6)$$

For Research Purposes Only

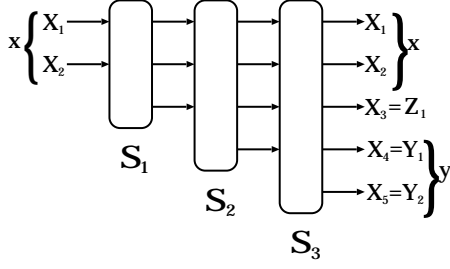


Figure 1: Generic circuit for computing linear transforms using 3 multiply/add operations.

The vectors \mathbf{v}_{N+i-1} should contain at most two non-zero elements one of them selected from the set \mathcal{C} . All other elements of \mathbf{v}_{N+i-1} are equal to zero. In general, an N -point FHT with A multiply/add operations can be represented with a generic circuit having a transfer function of the form

$$\mathbf{G}_N = (\mathbf{O}_{N \times A} \quad \mathbf{I}_N) \prod_{i=1}^A \mathbf{S}_{A-i+1}^{(N)}, \quad (7)$$

where $\mathbf{O}_{N \times A}$ denotes a zero matrix. Using the generic circuit, the mathematical form of the problem is given as follows.

Problem 3 (Mathematical Form) Find the vectors \mathbf{v}_{N+i-1} , $i = 1, \dots, A$, so that $\mathbf{G}_N = \mathbf{H}_N$. The non-zero elements of \mathbf{v}_{N+i-1} should be selected from \mathcal{C} .

4 Genetic Programming and Tree Coding of the FHTs

Using the generic representation (7), a search can be performed in the space of algorithms. In a search based on GP, the generic circuit should be coded as a tree structure [13]. Let the \mathbf{S} matrices for the generic circuit of Fig. 1 be specified as $\mathbf{S}_1^{(2)} = \begin{pmatrix} \mathbf{I}_2 \\ (1, 1) \end{pmatrix}$,

$$\mathbf{S}_2^{(2)} = \begin{pmatrix} \mathbf{I}_3 \\ (1, 0, -2) \end{pmatrix} \text{ and } \mathbf{S}_3^{(2)} = \begin{pmatrix} \mathbf{I}_4 \\ (0, 0, 1, -1) \end{pmatrix}.$$

We can code this circuit using tree structures consisting of 3 sub-trees each corresponding to one of the stages. An example is shown in Fig. 2. In this coding method the coefficient vectors \mathbf{v}_{N+i-1} are represented using subtrees denoting the corresponding mathematical operations. The details of the proposed coding method can be found in Table 1.

A tree-coded generic circuit with A stages can realize $k^A \prod_{i=N}^{N+A-1} \binom{i}{2}$ algorithms, where k is a number depending on the type of function nodes used in the tree. Some of these circuits result in identical matrices \mathbf{G}_N and a very small portion of them are HTs. The number of possible algorithms is very large even for small values of A and N .

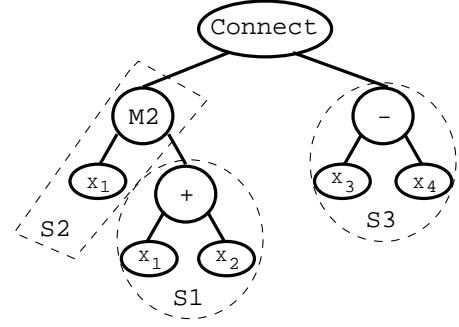


Figure 2: A tree coding for the circuit of Fig. 1. The realized algorithm is $\{Z_1 = X_1 + X_2, Y_1 = X_1 - 2X_3 = -X_1 - 2X_2, Y_2 = X_3 - X_4 = 2X_1 + 3X_2\}$.

The GP search is guided by a fitness function that assumes the values 1 for a solution. The fitness function f is the average of N simple functions f_i each evaluating the i th row of matrix \mathbf{G}_N in terms of orthogonality and the number of non-zero elements. The functions f_i vary in the interval $(0, \frac{1}{N}]$. Let $\mathbf{R} = \mathbf{G}_N \mathbf{G}_N^T$, and denote its i th row by \mathbf{r}_i . Also denote the i th row of the identity matrix \mathbf{I}_N by \mathbf{e}_i . The fitness function is defined as

$$f = \frac{1}{N} \sum_{i=1}^N f_i \quad (8)$$

$$f_i = \frac{1 - \alpha_1}{1 + \beta_1 (N - \sum_{j=1}^N \text{sgn}(\mathbf{G}_N[i, j]))} + \frac{1 - \alpha_1}{1 + \beta_2 \|\mathbf{r}_i - N\mathbf{e}_i\|_2}$$

where $\text{sgn}(\cdot)$ denotes the signature function.

5 Simulation Results

The results of a GP run for $N = 4$, $A = 7$ and specifications of Table 1 is given in Figs. 3 and 4. The circuit of Fig. 4 realizes the 4-point HT using 7 multiply/add operations. It is one of the $7^7 \prod_{i=4}^{10} \binom{i}{2} = 706026708072000$ algorithms utilizing 7 multiply/add operations that are realizable by the tree-coded generic circuit and is closely related to the circuit proposed in [8]. However, there is no systematic method to derive such algorithms. The algorithm in [8] is the result of a human search while the algorithm of Fig. 4 is the result of an automatic computer search. We also performed GP search to find faster algorithms for $N = 8$, but no superior algorithm has been obtained yet.

6 Conclusions

Performing an evolutionary search for an algorithm with some desired specifications may result in a solution. Because of the nature of evolutionary search methods, and the accumulated experience regarding their actual application to various problems, we can not expect an optimal solution in some formal sense. However, we can expect a solution that satisfies the

For Research Purposes Only

Table 1: GP tableau for FHT.

| | |
|-------------------|--|
| Objective | Problem 3 with $N = 4$, $A = 7$, $\mathcal{C} = \{\pm 1, \pm 2\}$. |
| Terminal set | $\{\mathbf{X}_1, \dots, \mathbf{X}_A\}$ |
| Function set | $\{\mathbf{Connect}, + : (X_p + X_q), \mathbf{P2} : (X_p + 2X_q), - : (X_p - X_q), \mathbf{M2} : (X_p - 2X_q)\}$ |
| Fitness | Eqn. (8) with $\alpha_1 = 0.8, \beta_1 = 0.05, \beta_2 = 0.5$. |
| Parameters | $M = 3000$ (Individuals/Generation) |
| Success predicate | Fitness of a tree is unity. |

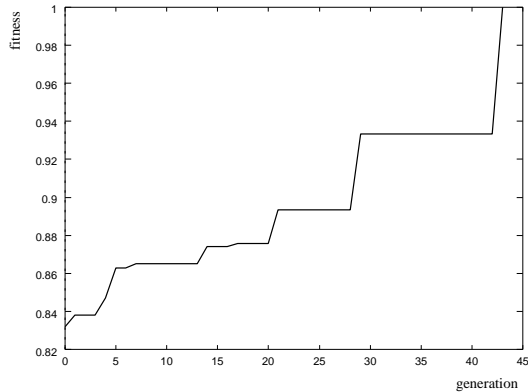


Figure 3: Best fitness versus generation in a successful run of GP for FHT.

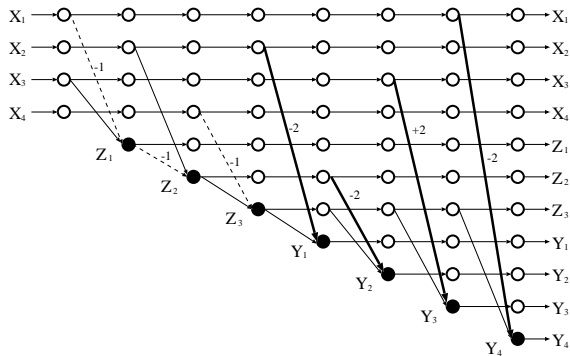


Figure 4: The FHT discovered during the run of Fig. 3.

specifications to a reasonable degree, a solution that can actually perform the task. In many situations, the form of optimal solution and its mathematical properties are not known. Even in such cases, an acceptable conventional solution usually exists. Any automatic solution that outperforms the conventional solution is clearly advantageous. The success of GP in generating a non-standard algorithm for 4-point FHT is encouraging and further research is required to clarify its applicability to the more complex and demanding tasks of generating fast algorithms for higher order HTs.

References

[1] R. K. Yarlagadda and J. E. Hershey, *Hadamard Matrix Analysis and Synthesis, With Applications*

to Communications and Signal/Image Processing. Boston: Kluwer Academic Publishers, 1996.

- [2] W. D. Wallis, A. P. Street and J. S. Wallis, *Combinatorics: Room Squares, Sum-Free Sets, Hadamard Matrices*. New York: Springer-Verlag, 1972.
- [3] H. Larsen, "An algorithm to compute the sequency ordered Walsh transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 335-336, 1976.
- [4] Y. A. Geadah and M. J. Corinthios, "Natural, dyadic and sequency order algorithms and processors for the Walsh-Hadamard Transform," *IEEE Trans. Comput.*, vol. C-26, pp. 435-442, 1977.
- [5] R. D. Brown, "A recursive algorithm for sequency ordered fast Walsh transform," *IEEE Trans. Comput.*, vol. C-26, no. 8, pp. 819-822, 1977.
- [6] M. H. Lee and M. Kaveh, "Fast Hadamard transform based on a simple matrix factorization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, no. 6, pp. 1666-1667, 1986.
- [7] M. M. Anguh and R. R. Martin, "A truncation method for computing Walsh transforms with applications to image processing," *CGIP*, vol. 55, no. 6, pp. 482-493, 1993.
- [8] D. Coppersmith, E. Faig and E. Linzer, "Hadamard transforms on multiply/add architectures," *IEEE Trans. Signal Processing*, vol. 42, no. 4, pp. 969-970, April 1994.
- [9] J. Morgenstern, "Note on a lower bound of the linear complexity of the fast Fourier transform," *Journal of the ACM*, vol. 20, no. 2, pp. 305-306, 1973.
- [10] J. Morgenstern, "The linear complexity of computation," *Journal of the ACM*, vol. 22, no. 2, pp. 184-194, 1975.
- [11] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.
- [12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
- [13] J. R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection," Cambridge, MA: MIT Press, 1992.
- [14] D. Beasley, D. R. Bull and R. R. Martin, "Complexity reduction using expansive coding," (*Evolutionary Computing: AISB Workshop, Leeds, U.K.*) *Lecture notes in computer science, Vol. 865*, pp. 304-319, 1994.