The Search for Hadamard Matrices of Order 428

The Road Leading Towards an Algorithm Construction

Scott King

Computer Algebra Research Group
of
Wilfrid Laurier University

Contact Information:
Physics and Computer Science
Wilfrid Laurier University
75 University Ave. W, ON, Canada

Email: king3730@mylaurier.ca

Abstract

I will be proving the result of a paper for finding a Hadamard matrix of order 428. Using four base sequences, that are Turyn-type, of lengths $n, n, n, n-1$, they are able to find four new sequences. These new sequences of lengths $2n-1$, $2n-1$, n and n are then used to generate a numerous amount of sequences which they filter to find the appropriate solution.

Motivation

The methodologies and algorithms behind this paper[1], yield an important result of being able to filter sequences to find solutions for Hadamard matrices.

- Naturally, the algorithm was not thoroughly detailed and thus important to fully determine *why* and *how* it works
- With this understanding comes the ability to manipulate and extend it to larger orders
- Larger orders are increasingly more difficult to prove and using this method may give new insights into solving them

Introduction

Proving the existence of Hadamard matrices is a problem with an increasing complexity and amount of data to handle.

We will need to know some of the mathematical constructs and devices use to make this possible.

Let's start by defining the **Hadamard matrix**; which is a $n \times n$ square matrix of order n, denoted as $H = (h_{ij})$. Each element of H is ± 1 such that $HH^T = I_n$. Thus, H is orthogonal.

$$\begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ 1 & 1 & -1 \end{pmatrix}$$

Next, let's look at **Turyn-type sequences**. They are quadruples of ± 1-sequences, (A, B, C, D), where each $A, ..., D$ have lengths of $n, n, n, n-1$ respectively.

For a given sequence of the quadruple of length n, there is an associated **nonperiodic autocorrelation function**. Denoted as $N_A(s), ..., N_D(s)$, where:

$$N_A(s) = \sum_{i=1}^{n-1-s} a_i a_{i+s} \tag{1}$$

where $s = 0, 1, ..., n-1$ and $N_A(s) = 0$ for $s \geq n$. These will eventually filter down the terms where $N_A(n-1)$ will give us one term.

Next, **Hall polynomials**, which are discrete generating functions. Denoted as:

$$h_A(t) = \sum_{i=0}^{n-1} a_i t^i \tag{2}$$

is a function returning constants.

From here, we are ready to start looking at the actual process of finding solutions.

Hadamard Matrices of Higher Orders

Hadamard matrices of certain orders are known to exist, but proving their existence is a non-trivial procedure. As the order increases, the time complexity and number of subsequent sequences increases.

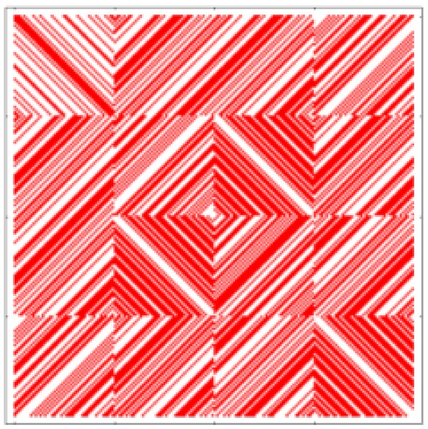

Figure 1: Hadamard matrix of order $4 \cdot 43 = 172$

Procedure & Algorithm

Problems of this magnitude and complexity cannot be done serially, due to hardware and especially time constraints. This is where parallel computing comes in handy.

Theta Filtering

Combining (2) with the 100 θ-filters, we are able to generate a list of constants:

$$h_Z(e^{i\theta}) = z_0 + z_1 e^{i\theta} + z_2 e^{2i\theta} + \cdots + z_{n-1} e^{(n-1)\cdot i\theta} \tag{3}$$

where $\theta \in [\frac{\pi}{100}, \frac{2\pi}{100}, ..., \frac{100\pi}{100}]$. From here we can combine (3) with the real function:

$$f_A(\theta) = |h_A(e^{i\theta})|^2 = Re(c)^2 + Im(c)^2 \tag{4}$$

And thus (4) will give us our magnitudes of the complex numbers and more importantly our large number of possible solution sequences.

These θ-filters are generated using meta-programming, using Maple. We can write a program to implement the filtering algorithm[1] to provide the code which saves certain sequences according to their identical first and last six entries. This is done serially.

Abstractions and Next Steps

- Upon the initial run of this filter, it produces a large file of 52GB
- Using a serial computer to find the specified sequence will take an unreasonably long time
- We will probably need to increase the number of filters

But we can parallelize some code to be able to do this in a timely manner. This can be done a few different ways:

- A majority of the code will be written in C using MPI, *Message Passing Interface*, which is a programming standard for high performance computing
- I am also doing some experiments with Mozilla's new systems language, Rust; which shows to have some performance gains completing regex operations

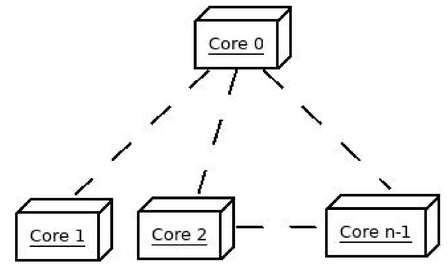

Figure 2: Illustration of parallel algorithm

Figure 2 shows us that we will divide the file between $n-1$ cores and each core will use a backtracking algorithm to determine whether the sequence is in solution space.

If this works for out the 4 sequences of the quadruple, then we have found our solution.

From our final results, we can translate these sequences to elements in the matrix and produce images such as Figure 1.

Forthcoming Research on Higher Orders

Following my confirmation of the algorithm for finding a Hadamard matrix of order 428, we will look at proving the existence of higher order matrices.

Acknowledgements

I would like to thank Dr. Ilias Kotsireas for giving me an opportunity in his lab and to develop methods for high performance, symbolic computing. Also, to SHARCNET for the ability to be able to run jobs on their machines. Credits to H.Kharagani and B.Tayfeh-Rezaie for publishing their findings.

References

[1] H. Kharaghani and B. Tayfeh-Rezaie. A hadamard matrix of order 428. pages 435–440, December 2004.