Fast Algorithm of Checking the Equivalence of Hadamard Matrices

Min-Ho Shin, Hong-Yeop Song

Dept. of Electrical and Computer Engineering, Yonsei University

This work was supported by the Basic Research Program of the Korea Science and Engineering Foundation under the grant number 97-0100-0501-3.

Abstract

In this paper, an algorithm of checking equivalence of Hadamard matrices is proposed. This algorithm is based on the relation between equivalence operation of Hadamard matrices and sorting. We present some computational results of checking equivalence of Hadamard matrices using this algorithm.

Ⅰ. Introduction

Hadamard matrices have some interesting structures and thus have been widely studied and used in the area of algebraic coding theory, communication systems engineering and image processing. In the area of wireless communication system design, one of the recent successful applications of the Hadamard matrices is their use of well known as Walsh codes of length 64, which are exactly the same as all the rows of a Hadamard matrix of order 64 recursively constructed by Kronecker product[1],[2],[4].

A Hadamard matrix H of order n is defined as an $n \times n$ matrix with all entries +1, or −1 such that[1]

$$H \cdot H^T = nI. \quad (1)$$

This implies any two rows of H are orthogonal. The orthogonal property of Hadamard matrices does not change by any of the following four operations:

(ⅰ) multiply some columns by −1
(ⅱ) multiply some rows by −1
(ⅲ) column permutations
(ⅳ) row permutations

Two Hadamard matrices related by some combination of the above four operations are called *equivalent*[1]. For a given Hadamard matrix we can find an equivalent one whose first row and first column contains +1 only. Such a Hadamard matrix is called *normalized*[1]. Hadamard matrices were first introduced by Jacques Hadamard in 1893. And yet, despite much attention by numerous researchers, the central question of the existence has not been completely answered. At present it is the order 428 for which no example is known and no non-existence is yet proved [4].

And still less is known about the classification of Hadamard matrices for general n - so far, the number of inequivalent Hadamard matrices of order n is known only for $n \leq 28$. These results are summarized as in the following table[3],[6].

Table 1. Number of inequivalent classes of Hadamard matrices of order n ($n \leq 28$)

order	$n \leq 12$	16	20	24	28
number of inequivalent classes	1	5	3	60	487

In this paper, computational complexity of checking the equivalence of Hadamard matrices are described in SectionⅡ. In Section Ⅲ, we present a fast equivalence-checking algorithm which lays its key idea on the relation between equivalence operation and sorting. We then present some computational results in Section Ⅳ. Finally in Section Ⅴ, this paper is concluded with some discussions on the results and with some remarks on further research.

Ⅱ. Computational Complexity

For a given Hadamard matrix of order n, we can find equivalent ones by using some combinations of four equivalence operation described in Section Ⅰ.

The number of such matrices are upper bounded by $(2^n \times n!)^2$, since there are 2^n ways of column(and row) multiplication by -1 and $n!$ ways of column(and row) permutation(see Fig.1.). Since such matrices are all candidates for the equivalence-check algorithm, the complexity of the algorithm depends on the number of candidates.

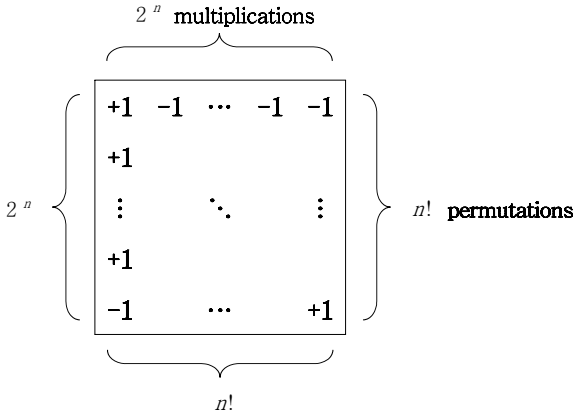

Fig.1. Number of candidates by 4-equivalence operation : $(2^n \times n!)^2$ cases

Since we can always make a given Hadamard matrix H to be normalized, we can confine equivalence operations only to row(and/or column) permutations. This reduces the number of candidates for equivalence-check to be $((n-1)!)^2$ as shown in Fig.2.

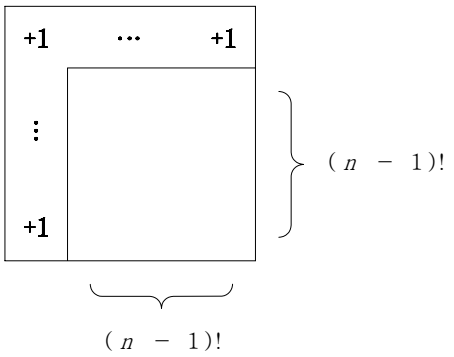

Fig.2. Number of candidates with normalized form : $(n-1)! \times (n-1)!$ cases

Making the natural transition from the multiplicative group of order two, $\{-1, +1\}$, to the additive group, $\{0, 1\}$, we can represent $n \times n$ Hadamard matrices with n binary column vectors or row vectors(n-tuple). Then as shown in Fig.3. by reading binary column(or row) vectors as decimal values and by applying normal sorting columnwise (or rowwise), we can reduce the number of candidates for equivalence-check by $(n-1)!$.

In the remaining of this paper, we will describe a proposed equivalence-checking algorithm, and then analyze its computational complexity in terms of the number of candidates for the equivalence-check. For example if a normalization is considered(as in Fig.2.) we say the algorithm has $((n-1)!)^2$ -complexity, and in case of Fig.3. $(n-1)!$ -complexity.

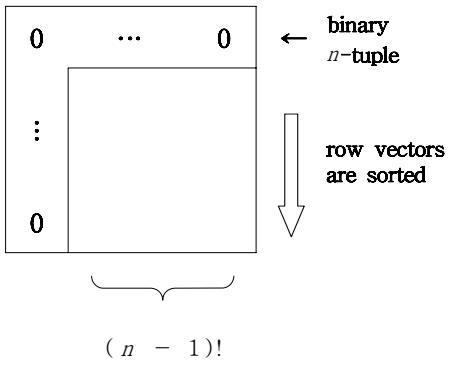

Fig.3. Number of candidates by sorting row vectors : $(n-1)!$ cases

Ⅲ. Main Algorithm

In this section, we will describe our main equivalence-check algorithm. For convenience, we will consider binary Hadamard matrices over $GF(2)$. The idea that the specific column(or row) permutation can be done by sorting is applied both columnwise and rowwise, i.e. for a given Hadamard matrix H we first normalize it and then carry out row-sorting and column-sorting successively and repeatedly until both rows and columns are completely sorted. This procedure reduces the complexity of the checking algorithm.

For example let's apply this procedure to a Hadamard matrix H_8 shown in Fig.4. which is constructed by Sylvester method[1][4].

0	0	0	0	0	0	0	0	← 0
0	1	0	1	0	1	0	1	← 85
0	0	1	1	0	0	1	1	← 51
0	1	1	0	0	1	1	0	← 102
0	0	0	0	1	1	1	1	← 15
0	1	0	1	1	0	1	0	← 90
0	0	1	1	1	1	0	0	← 60
0	1	1	0	1	0	0	1	← 105
↑	↑	↑	↑	↑	↑	↑	↑	
0	85	51	102	15	90	60	105	

Fig.4. H_8 constructed by Sylvester method

In Fig.4, the decimal value of each 8-tuple binary vector is shown. We then apply above procedure: column-sorting and row-sorting iteratively. This gives the following transformed matrix H'_8 a sorted one(see Fig.5.).

0	0	0	0	0	0	0	0	←	0
0	0	0	0	1	1	1	1	←	15
0	0	1	1	0	0	1	1	←	51
0	0	1	1	1	1	0	0	←	60
0	1	0	1	0	1	0	1	←	85
0	1	0	1	1	0	1	0	←	90
0	1	1	0	0	1	1	0	←	102
0	1	1	0	1	0	0	1	←	105
↑	↑	↑	↑	↑	↑	↑	↑		
0	15	51	60	85	90	102	105		

Fig.5. H'_8 equivalently transformed(sorted)

Observing the resulting matrices, one can notice that this procedure transforms given Hadamard matrices into those which are as symmetric as possible. Using this transform-like procedure, we then propose the following equivalence-check algorithm of Hadamard matrices.

Making candidates (Procedure A):

(a) choose p [permutation number]
(b) given H, test_num=0
(c) normalize H
(d) row-sorting
(e) column-sorting
(f) If not row sorted then [Go to (d)]
(g) store the resulting matrix as candidate H', [test_num = test_num+1]
(h) If test_num<p
then [permute H, Go to (c)]
(i) candidates are determined. STOP

Equivalence check :

(1) given H_a, H_b , test_num=0
(2) make all candidates of H_a using the *procedure A*
(3) make a candidate of H_b using only steps from (c) to (g) of *procedure A*
(4) If one of candidates of H_a is the same as H'_b
then [H_a and H_b are equivalent, STOP]
(5) If test_num < p
then [test_num=test_num+1, Go To (3)]
(6) H_a and H_b are inequivalent. STOP

As described in the algorithm, to make candidates using sorting, we should check if the algorithm terminate.

Theorem. Given a Hadamard matrix H, the Step(g) of the *procedure A* can be reached in a finite number of steps. That is, *Making Candidates* algorithm terminates.

proof : After normalized in Step (c), the algorithm repeats row-sorting(Step (d))→column-sorting(Step (e)), until both rows and columns are completely sorted. In Step (f), the resulting matrix is column sorted but possibly not row sorted, say decimal value of *i'th* row is greater than one of decimal values of following rows, say decimal value of *j'th* row($j>i$) is minimum among such rows.(see Fig.6.) Then at Step (d) row sorting will permute row vectors as shown in Fig.7.

Since one more column-sorting(Step (e)) does not affect the first i row vectors, we can say at least the first i row vectors are sorted in Step (f). Hence the algorithm terminates. ■

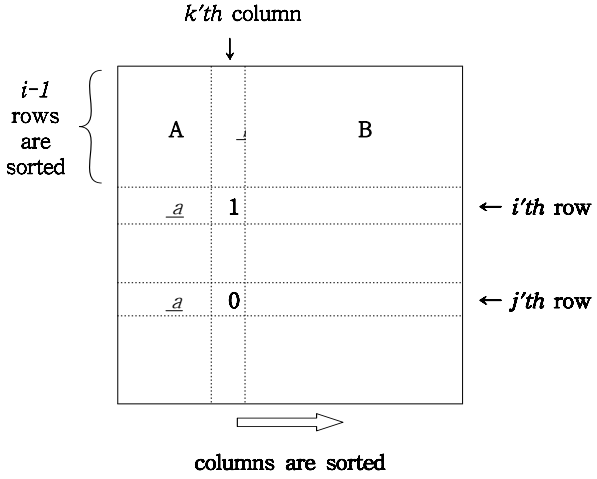

Fig.6. Case of in Step (f) that rows are not sorted completely

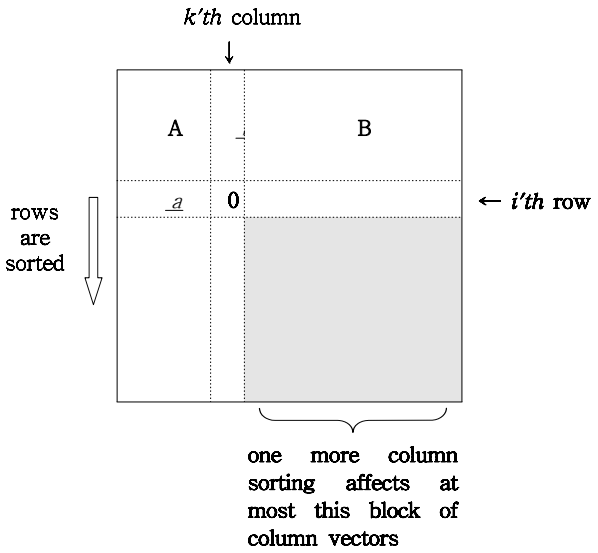

Fig.7. Matrix form when Step (d) is over

Ⅳ. Some Computational Results

In this section we present some computational results of our main algorithm. For this, we construct possibly inequivalent Hadamard matrices of order 16 by the following method.[5].

Consider the Sylvester type Hadamard matrix H_{16}. We can construct such a matrix with H_8 (also a Sylvester-type) as follows[1],[4].

$$H_{16} = H_8 \otimes H_2 = \begin{bmatrix} H_8 & H_8 \\ H_8 & \overline{H}_8 \end{bmatrix} \quad (2)$$

Then we can construct $8!$ different Hadamard matrices of order 16 by row permutation of H_8 in the first block[5]. As introduced in Section I there are five inequivalent classes of Hadamard matrices of order 16. Applying our algorithm with the permutation number in Step (a) to be 16, and permutation in Step (h) to be cyclic rotation of column vectors, we have classified all $8!$ such matrices into 4 inequivalent classes. These results are summarized as in the following table.

Table2. Computational Results of checking equivalence of the 16×16 Hadamard matrices

Equivalence class	Computer search results
H_1^1	1,344
H_2^2	9,408
H_3^3	18,816
H_4^5	0
H_5^4	10,752

Here, each equivalence class in the table also gives information about the equivalence class of its transpose: the superscript denotes the matrix equivalence class of the transpose. Thus H_5^4 indicates its transpose is in the class of H_4^5[11].

Ⅴ. Concluding Remarks

We have proposed an equivalence-check algorithm of Hadamard matrices. The algorithm is based on a relation between equivalence operation and sorting: using this idea we reduced the computational complexity by reducing the number of candidates to be checked. Some computational results with this algorithm of n-complexity was presented in Section Ⅳ, which has been done in 40[sec] with Pentium MMX-266 processor.

As a future work, some algebraic analysis of this algorithm perhaps by design theory or graph theory is desirable to apply this algorithm for a higher order of Hadamard matrices with lesser complexity.

REFERENCES

[1] J. H. Van Lint & R. M. Wilson, *A Course In Combinatorics*, Cambridge University Press, 1992.

[2] CIA/TIA/IS-95 Mobile Station -- Base Station Compatibility Standard for Dual-Mode Sideband Spread Spectrum Cellular System, Published by Tele communication Industry Association as a North American 1.5MHz Cellular CDMA Air - Interface Standard, July, 1993.

[3] H. Kimura, "Classification of Hadamard matrices of order 28", *Discrete Math.*, Vol. 133, pp. 171-180, 1994

[4] M. Hall Jr, *Combinatorial Theory, Second Ed,* John Wiley and Sons, 1986.

[5] J.-S. No & H.-Y. Song, "Generalization of Sylvester-Type Constructions for Hadamard Matrices", preprint, 1999.

[6] N. J. A. Sloane, *On-Line Encyclopedia of Integer Sequences,* sequence no : 1,1,1,5,3,60,487 http://www.research.att.com/~njas/sequences/index.html, 1999.