Generating binary partial Hadamard matrices

V. Álvarez [a], J.A. Armario [a], R.M. Falcón [a], M.D. Frau [a], F. Gudiel [a], M.B. Güemes [b], A. Osuna [a]

[a] *Dpto. Matemática Aplicada I, Universidad de Sevilla, Spain*
[b] *Dpto. Álgebra, Universidad de Sevilla, Spain*

Keywords:
Partial Hadamard matrix
Hadamard graph
Clique
Constraint satisfaction problem

A B S T R A C T

This paper deals with partial binary Hadamard matrices. Although there is a fast simple way to generate about a half (which is the best asymptotic bound known so far, see de Launey (2000) and de Launey and Gordon (2001)) of a full Hadamard matrix, it cannot provide larger partial Hadamard matrices beyond this bound. In order to overcome such a limitation, we introduce a particular subgraph G_t of Ito's Hadamard Graph $\Delta(4t)$ (Ito, 1985), and study some of its properties, which facilitates that a procedure may be designed for constructing large partial Hadamard matrices. The key idea is translating the problem of extending a given clique in G_t into a Constraint Satisfaction Problem, to be solved by MINION (Gent et al., 2006). Actually, iteration of this process ends with large partial Hadamard matrices, usually beyond the bound of half a full Hadamard matrix, at least as our computation capabilities have led us thus far.

1. Introduction

Binary Hadamard matrices consist in $\{1, -1\}$-square $n \times n$ matrices H such that $HH^T = nI_n$. It may be straightforwardly checked that such a matrix H must be of size 1, 2 or a multiple of 4, as soon as three rows are assumed to be mutually orthogonal. Actually, it seems that Hadamard matrices might exist for every order multiple of 4, as the Hadamard Conjecture claims, although this fact remains still open for more than a century (see [10] for further details).

From a practical point of view, taking into account possible applications, sometimes there is no need to consider a full Hadamard matrix. In fact, it suffices to meet a large amount of pairwise orthogonal rows. This has originated the interest in constructing binary *partial Hadamard matrices*, that is, $m \times 4t$ $(1, -1)$-matrices PH satisfying $PH \cdot PH^T = 4tI_m$, for $m \leq 4t$. We call m the *depth* of PH.

For instance, existence of large partial Hadamard matrices implies large lower bounds on the size of several interesting combinatorial objects, as pointed out in [5]. Unfortunately it seems that their explicit construction is equally hard as well.

De Launey proved in [4] that partial Hadamard matrices of size about a third of a full $4t \times 4t$ Hadamard matrix exist for large t. The proof gives a polynomial time algorithm in t for constructing such a matrix. Furthermore, De Launey and Gordon proved in [5] that about a half of a Hadamard matrix $4t \times 4t$ exists for large t, assuming that the Riemann hypothesis is true. The idea was decomposing $2t - i$ as the sum of i odd prime numbers p_i, $2 \leq i \leq 3$, so that the juxtaposition of the corresponding Paley conference matrices provides a partial Hadamard matrix of depth $2\min\{p_i\} + 2$.

From a theoretical point of view, one could proceed even in a simpler way, avoiding this assumption concerning the Riemann hypothesis, which is just used in order to provide an explicit constructive algorithm working in polynomial time.

Actually, given any positive integer t, consider the set $\mathcal{PH}_t$ of pairs of integers (t_1, t_2) such that $t_1 \leq \lfloor \frac{t}{2} \rfloor \leq t_2$, $t_1 + t_2 = t$ and some full Hadamard matrices H_{4t_1} and H_{4t_2} exist of orders $4t_1$ and $4t_2$, respectively.

Let $m = \max_{(t_1,t_2)\in\mathcal{PH}_t} t_1$. Then a partial Hadamard matrix PH of order $4m \times 4t$ may be straightforwardly constructed, as soon as any full Hadamard matrix of order $4m$ and a collection of whatever $4m$ rows of any full Hadamard matrix of order $4(t - m)$ are juxtaposed. Notice that, as defined, $4m$ is necessarily close to $2t$, since it may rarely occur that systematically no full Hadamard matrices exist of orders $4t_1$ and $4t_2$, for $t_1 \leq t_2$, $t_1 \sim t_2$, $t_1 + t_2 = 2t$, no matter the Hadamard Conjecture has not been proved yet. As a matter of fact, a look at the updated list of integers $t < 500$ for which no Hadamard matrices of order $4t$ are known (namely, 167, 179, 223, 283, 311, 347, 359, 419, 443, 479, 487, 491 [7]), supports this idea.

Unfortunately, none of these methods can provide a partial Hadamard matrix of depth greater than half of a full Hadamard matrix. The aim of this work is to describe an alternative procedure providing large partial Hadamard matrices as well, hopefully beyond this bound. Actually, this will be the case for the examples we have worked out.

The work may be summarized as follows. Partial Hadamard matrices will be naturally identified as *cliques* of a suitable subgraph of Ito's Hadamard Graph [11]. This subgraph and its properties will be analyzed in Section 2. From this information, in Section 3 the problem of adding a new vertex to a given clique in G_t will be described as a Constraint Satisfaction Problem, to be solved by means of MINION [8]. Some examples will be provided. Last section will be devoted to conclusions and comments about further work.

A brief sketch of the work has been recently exposed in [1], as a result of a substantial progress on a previous work of some of the authors [2].

2. The graph G_t

Hadamard Graphs were introduced by Ito in [11]. Originally they referred to the graph $\Delta(4t)$ whose vertices are the $(1, -1)$-vectors of length $4t$ consisting of an even number of 1s. The adjacency relation consists in orthogonality.

We call Hadamard graph to the subgraph G_t of $\Delta(4t)$ induced by the $(1, -1)$-vectors simultaneously orthogonal to the three first rows of a normalized Hadamard matrix,

$$\left(\begin{array}{ccc|ccc|ccc|ccc} 1 & \dots & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & \dots & 1 \\ 1 & \dots & 1 & 1 & \dots & 1 & -1 & \dots & -1 & -1 & \dots & -1 \\ 1 & \dots & 1 & -1 & \dots & -1 & 1 & \dots & 1 & -1 & \dots & -1 \\ & \dots & & & \dots & & & \dots & & & \dots & \end{array}\right).$$

These orthogonality conditions characterize straightforwardly the form of the vertices in G_t, as follows.

Lemma 1. *The vertices of G_t consist of $(1, -1)$-vectors of length $4t$ where the $2t$ negative entries are distributed so that exactly k, $t - k$, $t - k$ and k negative entries occur respectively among the ranges $[1, \dots, t]$, $[t + 1, \dots, 2t]$, $[2t + 1, \dots, 3t]$ and $[3t + 1, \dots, 4t]$, for some $0 \leq k \leq t$.*

We may then classify the set of vertices in G_t attending to the number k of negative entries which appear in positions 1 to t. In what follows, a k-vertex (or k-vector) in G_t refers to a vertex with precisely k negative entries among positions 1 to t. Actually, for the remainder of the paper, it can be assumed that $0 \leq k \leq \lfloor \frac{t}{2} \rfloor$, since any vector $\mathbf{v}$ and its negated $-\mathbf{v}$ share a common set of adjacent vertices.

It is readily checked that the size of G_t (both in vertices and edges) grows exponentially on t.

Lemma 2. *In particular, the number of vertices in G_t is $|G_t| = \sum_{k=0}^{t} \binom{t}{k}^4$.*

Hereafter, for brevity, we will adopt the additive notation for representing Hadamard matrices, so that the 1s turn to 0s and the -1s turn to 1s. This way, k-vectors in G_t are now described as $(0, 1)$-vectors of length $4t$ consisting of precisely $2t$ ones (and hence $2t$ zeros), which are distributed in the following way: there are exactly k ones in positions 1 to t, other $t - k$ ones in positions $t + 1$ to $2t$, another $t - k$ ones in positions $2t + 1$ to $3t$, the last k ones being located in positions $3t + 1$ to $4t$.

Each of these k-vectors may be straightforwardly codified as an integer, assuming that the k-vector is the binary representation of a decimal number. Therefore, cliques are codified as lists of integers, each of them being the decimal representation of a binary number consisting of $2t$ ones and length less or equal to $4t$.

Since cliques of size m in G_t translate to partial Hadamard matrices $(m + 3) \times 4t$, we would like to search for large cliques in G_t. Since the largest clique in $\Delta(4t)$ is at most of size $4t$, the largest clique in G_t is at most of size $4t - 3$. Cliques meeting this upper bound would correspond, in turn, to full $4t \times 4t$ Hadamard matrices.

A *maximum clique* is a clique with the maximum cardinality (which is called the maximum clique number). This notion is different from that of maximal clique, which refers to a clique which is not a proper subset of any other clique. Thus, maximal cliques do not need to be maximum ones, though the converse is always true. Concerning cliques in G_t, this means that a partial Hadamard matrix does not need to be a submatrix of a full Hadamard matrix.

Given a graph, the *maximum clique problem* consists in finding a maximum clique, and it is NP-complete [3]. Furthermore, it is known that there is no polynomial-time algorithm for approximating the maximum clique within a factor of $n^{1-\epsilon}$ unless P = NP [9], where n is the number of the vertices of the graph. Moreover, there is no polynomial-time algorithm approximating the clique number within a factor of $\frac{n}{(\log n)^{1-\epsilon}}$ unless NP = ZPP [12].

Fortunately, the aim of the paper is not to describe a general purpose algorithm for solving the maximum clique problem. The aspiration is to design an ad hoc algorithm for constructing sufficiently large cliques in G_t. To this end, we need to study the properties of G_t in a more detailed way.

Firstly, we describe a procedure for determining a set $\delta_{\mathbf{v}}$ of generators for the adjacency list related to a fixed k-vector $\mathbf{v}$, that is, for those s-vectors $\mathbf{w}$ sharing exactly $2t$ entries with $\mathbf{v}$.

Lemma 3. *At the first and fourth (resp., the second and third) quarters the number i of coincidences in* 1*s (resp., in* 0*s) runs in the range* $i \in [0, \min(k, s)]$.

Lemma 4. *At each quarter, the number* α_i *of total coincidences (both in* 1*s and* 0*s) satisfies* $\alpha_i = t - s - k + 2i$, *and runs in the range* $\alpha_i \in [t - s - k, t - |s - k|]$.

Corollary 1. $\alpha_{i+1} = \alpha_i + 2$.

Let $n = \min(k, s)$. In the conditions above, the set of total coincidences is given by $t - s - k = \alpha_0 < \cdots < \alpha_n = t - |s - k|$. We may now describe the set of s-vectors adjacent to a given k-vector.

Proposition 1. *The set of vectors that are orthogonal to a given k-vector corresponds to the full set of distributions of vectors satisfying tuples of total coincidences* $(\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}, \alpha_{i_4})$ *such that* $\alpha_{i_1} + \alpha_{i_2} + \alpha_{i_3} + \alpha_{i_4} = 2t$.

Proposition 2. *The set of tuples* $(\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}, \alpha_{i_4})$ *which give rise to orthogonal s-vectors are characterized as the solutions of the following system of diophantine equations*

$$\begin{cases} x_0\alpha_0 + \ldots + x_n\alpha_n = 2t \\ x_0 + \ldots + x_n = 4 \\ x_i \in \mathbb{Z} : \ 0 \le x_i \le 4 \end{cases} \tag{2.1}$$

Here, $n = \min(k, s)$ *and* x_i *indicates how many coincidences of the type* α_i *must occur among the four quarters.*

We now give a constructive way to solve the system above.

Proposition 3. *There exists a solution for the system (2.1) if and only if* $4\alpha_0 \le 2t \le 4\alpha_n$.

Corollary 2. *Fixed* t *and* $0 \le k, s \le \lfloor \frac{t}{2} \rfloor$, *the set sol of solutions to the system (2.1) may be constructed in the following way:*

$sol \leftarrow \emptyset$
$\alpha_0 \leftarrow t - k - s$
$\alpha_n \leftarrow t - |k - s|$
`for` i_1 `from` $\max\{\alpha_1, 2t - 3\alpha_n\}$ `to` $\min\{\alpha_n, \lfloor \frac{2t}{4} \rfloor\}$ `with step 2 do`
  `for` i_2 `from` $\max\{i_1, 2t - i_1 - 2\alpha_n\}$ `to` $\min\{\alpha_n, \lfloor \frac{2t - i_1}{3} \rfloor\}$ `with step 2 do`
    `for` i_3 `from` $\max\{i_2, 2t - i_1 - i_2 - \alpha_n\}$ `to` $\min\{\alpha_n, \lfloor \frac{2t - i_1 - i_2}{2} \rfloor\}$ `with step 2 do`
      $sol \leftarrow sol \cup \{\{i_1, i_2, i_3, 2t - i_1 - i_2 - i_3\}\}$
    `od`
  `od`
`od`

Given a tuple $(\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3}, \alpha_{i_4})$ solution to (2.1), construct the four matrices N_k whose rows are those vectors satisfying α_{i_k} total coincidences with the corresponding quarter of $\mathbf{v}$. By construction, the juxtaposition of any of the rows of these matrices gives a vector orthogonal to $\mathbf{v}$.

Proposition 4. *A set* $\delta_{\mathbf{v}}$ *of generators for the adjacency list of* $\mathbf{v}$ *may be straightforwardly constructed in terms of matrices of the type above.*

Proposition 5. *Fixed a k-vector* $\mathbf{v}$, *there exist s-vectors orthogonal to* $\mathbf{v}$ *if and only if* $s \in [\lceil \frac{t}{2} \rceil - k, \lfloor \frac{t}{2} \rfloor]$.

Furthermore, we may straightforwardly precise the number of s-vectors orthogonal to a given k-vector, for some fixed $s \in [\lceil \frac{t}{2} \rceil - k, \lfloor \frac{t}{2} \rfloor]$.

Table 2.1
Rows in Hadamard matrices are s-rows, for $s \in \{\lfloor \frac{t}{2} \rfloor - 1, \lfloor \frac{t}{2} \rfloor\}$.

t	3	4	5	6	7	8	9
row	4	6	5	4	5	4	4
$\#\{s = \lfloor \frac{t}{2} \rfloor\}$	8	8	15	11	21	12	26
$\#\{s = \lfloor \frac{t}{2} \rfloor - 1\}$	0	4	1	9	3	16	6

Lemma 5. *Fixed a valid distribution (i_1, i_2, i_3, i_4), the number of s-vectors orthogonal to a given k-vector is given by the expression:*

$$\binom{k}{i_1}\binom{t-k}{s-i_1}\binom{k}{i_2}\binom{t-k}{s-i_2}\binom{k}{i_3}\binom{t-k}{s-i_3}\binom{k}{i_4}\binom{t-k}{s-i_4}$$

In particular, this might suggest that large cliques in G_t should more likely consist of k-vectors, for large values of k, close to $\lfloor \frac{t}{2} \rfloor$. This seems to be the case in practice, as the following calculations suggest.

For each $3 \leq t \leq 9$, we choose at random a Hadamard matrix of order $4t$ from Sloane's online library [14], say `had.12`, `had16.4`, `had20.hall.n`, `had24.pal`, `had28.pal2`, `had32.pal`, `had36.pal2`.

We now normalize these matrices, by means of the following algorithm. Notice that since just negation and permutation of columns are used, the Hadamard character of the matrix is preserved.

Algorithm 1. *Hadamard normalization*

- Negate those columns consisting of a first negative entry.
- Now, locate those columns i, $1 \leq i \leq 2t$, consisting of a second negative entry. Similarly, locate those columns j, $2t + 1 \leq j \leq 4t$, consisting of a second positive entry. Interchange them.
- Proceed as the step before, now by quarters. As a result, you will obtain a normalized Hadamard matrix.

Fixed a $\lfloor \frac{t}{2} \rfloor$-vector among the rows of these matrices (as indicated in Table 2.1), one may have a look on the range in which the values s run, for the s-vectors defining the remaining rows.

Table 2.1 suggests that one should focus on s-vectors for $s \in \{\lfloor \frac{t}{2} \rfloor - 1, \lfloor \frac{t}{2} \rfloor\}$. We take advantage of this fact for speeding the algorithm extending cliques in G_t, as described in the following section.

3. A CSP for extending cliques in G_t

As we commented before, finding the maximum clique of a graph is a NP-Hard problem, and consequently exact algorithms for this purpose are infeasible even in case of moderately large problem instances. Therefore most of the efforts to give practical solutions to the maximum clique problem are based on heuristic approaches. The authors themselves tried out some of them in a preliminary work [2]. Unfortunately, no matter the choice of the heuristic is, all of them require to explicitly construct the adjacency lists of the vertices which are considered along the computation. And it is apparent from the precedent section that such a task is intractable for the case of the graph G_t, as soon as t increases.

We describe here a novel and completely different approach. The idea is translating the problem of extending a given clique in G_t, into a Constraint Satisfaction Problem [6] (CSP in brief, hereafter).

The constraint satisfaction paradigm pursues solving a problem as the set of simultaneous solutions to a series of constraints completely characterizing the former. As noticed in [6], *"constraints identify the impossible and reduce the realm of possibilities to effectively focus on the possible, allowing for a natural declarative formulation of what must be satisfied, without expressing how"*.

Any model characterizing a CSP consists of a finite set of variables, their finite domains and the constraints to be satisfied. As usual, it often occurs that the same problem may be modeled in different ways, each of them involving possibly different complexities. A careful study of the particular circumstances of the problem should permit discriminating between them.

A solution to an instance of CSP is an assignment to each variable, such that all constraints are simultaneously satisfied. Solvers typically find all (or optionally just one) solutions, if any does exist.

In our case, an explicit formulation of the CSP relays on the knowledge of the properties of G_t described in the previous section. More specifically, given a clique C in G_t, we look for an s-vector (condition (C2) below) $\mathbf{x} = (x_1, \ldots, x_{4t})$, for the two most promising values of s (condition (C1) below), which is simultaneously orthogonal to every vertex already in the clique (condition (C3) below). Therefore, a model for the CSP consists in the following constraints:

(C1) $\lfloor \frac{t}{2} \rfloor - 1 \leq s \leq \lfloor \frac{t}{2} \rfloor$.
(C2) The number of -1s in the ranges $(x_1, \ldots, x_t)$, $(x_{t+1}, \ldots, x_{2t})$, $(x_{2t+1}, \ldots, x_{3t})$ and $(x_{3t+1}, \ldots, x_{4t})$ are s, $t - s$, $t - s$ and s, respectively.
(C3) The number of coincidences of $\mathbf{v}$ with each of the vertices already in C is $2t$.

Once a model is fixed, the following step consists in carrying it into a solver. A translation of the constraints defining the model into the syntaxis of the solver is needed. In this paper we make use of MINION [8], one of the fastest and most scalable constraint solvers using the "model and run" methodology. Actually, this solver is based on the common technique of alternating between splitting and propagation. The key point here is that splitting is minimized in practice by effective propagation, since the former inherits an exponential-time solution method.

Nevertheless, unlike most other constraint solvers, MINION does not break up constraints into smaller pieces, neither introduce new variables nor simplify or manipulate constraints. Typically, a depth-first chronological backtracking is performed by default. But few further information is available about the internals of MINION's performance. In essence, it is a black box from the user point of view, deliberately providing few options, but guaranteeing raw speed in return.

MINION expects to be provided with the name of an input file `.min` as an argument. This file contains a specification of the CSP to be solved (in terms of variables and constraints, to be declared in separated sections) as well as settings that the search process should use (a number of switches are supported to augment default behavior).

The constraints section consists of any number of constraint declarations on separate lines, constructed from the limited range of elementary constraints which are available. Four different variable types may be declared. Sorted by speed of performance, boolean (`BOOL`) variables operate faster, followed by discrete (`DISCRETE`), delimited domains (`BOUND`), and arbitrary ranges of integers (`SPARSEBOUND`). If no variable ordering is explicitly stated, then one is generated based on the order the variables are declared. Obviously, depending on the choice of the ordering, the time required for finding a solution may vary in a significant way.

The interested reader is referred to [8,13] for more information concerning MINION.

We now describe the way in which the CSP for extending a given clique C in G_t has been formalized into MINION syntax.

Three variables are considered. The desired vertex $\mathbf{x} = (x_1, \ldots, x_{4t})$ which might potentially extend the clique C is codified as a boolean matrix `x` of $4 \times t$ unknowns, the row i related to the range of coordinates $x_{(i-1)t+1}, \ldots, x_{it}$, for $1 \leq i \leq 4$.

Assuming the advantages commented in the precedent section, a discrete variable s
`DISCRETE s` $\{\lfloor \frac{t}{2} \rfloor - 1 .. \lfloor \frac{t}{2} \rfloor\}$ provides straightforwardly a simple way to fulfill the constraint (C1).

Conditions (C2) are codified in terms of the constraint `occurrence`(**vec**, *elem*, *count*), which ensures that there are *count* occurrences of the value *elem* in the vector **vec**. This way, the set of constraints
`ocurrence`([**x**[0, _]], 1, s)
`ocurrence`([**x**[1, _]], 0, s)
`ocurrence`([**x**[2, _]], 0, s)
`ocurrence`([**x**[3, _]], 1, s)
characterizes that the vertex **x** which we are looking for defines an s-vector.

Finally, translating the relations (C3) requires a more sophisticated elaboration, which combines the use of the constraint `element`(**vec**, i, e) and the constraint `reify`(*cons*, r). The former specifies that, in any solution, **vec**$[i] = e$. The latter ensures that the boolean variable r is set to 1 if and only if the constraint *cons* is satisfied.

Consider a boolean matrix `coin[|C|,4t]`, whose ith-row will keep trace of the entrywise coincidences of **x** and the ith vertex of C, so that **coin**$[i-1, j-1] = 1$ if and only if **x**$[\lfloor \frac{j-1}{t} \rfloor, (j-1) \mod t] = C(i, j)$. As soon as the ith-row of `coin` consists of exactly $2t$ ones, then the vertex **x** will be orthogonal to the vertex i in C. Therefore, the relations (C3) are codified as $|C|$ blocks of $4t + 1$ constraints of the type:
`reify(element(`**x**$[0, \_], 0, C(i, 1)$`),`**coin**$[i-1, 0]$)
$\vdots$
`reify(element(`**x**$[0, \_], t-1, C(i, t)$`),`**coin**$[i-1, t-1]$)
$\vdots$
`reify(element(`**x**$[3, \_], 4t-1, C(i, 4t)$`),`**coin**$[i-1, 4t-1]$)
`ocurrence(`[**coin**$[i-1, \_]], 1, 2t$)

Obviously, it is far from being operative that all these constraints are introduced by hand, step by step. Nevertheless, it is preferable to make use of some programming language and design a small executable file which, provided C as input data, straightforwardly generates the full code of the MINION file .min to be executed in turn.

We have performed 10 searches for each t in the range $3 \leq t \leq 11$. In all cases, starting from a vertex randomly generated, the procedure has consisted in iteratively trying to add a new vertex to the structure already constructed, until the CSP fails to provide a new vertex. Table 3.2 resumes the average time (Av.T.) for each of these calculations, as well as the smallest (Sm.) and largest (Lg.) sizes of the cliques found so far.

Notice that for small t full or almost full Hadamard matrices have been found, and for large t cliques have been found around the bound $2t$.

4. Conclusions and further work

In this paper a new way of generating (possibly large) partial Hadamard matrices $m \times 4t$ has been described. Two are the main novelties of the approach. On one hand, the problem has been translated to the context of Graph Theory, involving the construction of cliques in certain subgraph G_t of Ito's Hadamard Graph $\Delta(4t)$. Secondly, the problem itself of extending a clique has been described as a Constraint Satisfaction Problem.

Table 3.2
A summary of the results, involving about 1500 runs of the procedure.

t	Av.T.	Sm.	Lg.
3	$1''$	9	9
4	$1''$	12	13
5	$1''$	9	17
6	$2''$	9	21
7	$8''$	10	17
8	$39''$	12	21
9	$7'\ 50''$	14	16
10	$1^h\ 32'\ 10''$	15	17
11	$8^h\ 38'\ 9''$	14	17

Although the size of the graph G_t certainly makes the problem untractable even for not so large values of t, in comparison to the work in [2], this approach facilitates extending cliques for larger values of t. Furthermore, for common values of t, it improves either the size of the output clique, or the required time for execution, or even both.

As a matter of fact, it is worth noting that the approach itself might be used to extend those cliques described in the introduction, rounding the bound of $2t$ vertices, obtained by the juxtaposition of most of the rows of two full Hadamard matrices of appropriated sizes $t_1 + t_2 = t$. For instance, progressing from the juxtaposition of two Hadamard matrices of order 16, iteration of the procedure ends providing a full Hadamard matrix of order 32. It takes barely 8 s to add a vertex in each step. Unfortunately, the sizes of the objects which might be of real interest (those related to orders for which no full Hadamard matrices are known) are out of the scope of the actual capabilities for which the community has access to, for the moment.

However, as a secondary benefit, the procedure itself as defined might shed light on a new practical way to afford hard problems on Graph Theory, in terms of Constraint Satisfaction Problems accurately designed ad hoc, far beyond the traditional and didactical use of the graph coloring problem for illustrating the typical performance of a CSP. The door is open to many other interesting and potential applications.

References

[1] V. Álvarez, J.A. Armario, R.M. Falcón, M.D. Frau, F. Gudiel, M.B. Güemes, A. Osuna, Generating partial Hadamard matrices as solutions to a constraint satisfaction problem characterizing cliques. in: Proceedings of X Encuentro Andaluz de Matemática Discreta, pp. 17–20 ISBN: 978-84-697-4743-8, (La Línea, Cádiz, July 10–11 2017).

[2] V. Álvarez, J.A. Armario, M.D. Frau, F. Gudiel, M.B. Güemes, E. Martín, A. Osuna, Searching for partial Hadamard matrices, 2012, arXiv:1201.4021 [math.CO].

[3] I.M. Bomze, M. Budinich, P.M. Paradalos, M. Pelillo, in: D.Z. Du, P.M. Paradalos (Eds.), The maximum clique problem, in: Handbook of Combinatorial Optimization, vol. 4, Kluwer, Norwell, MA, 1999.

[4] W. de Launey, On the assymptotic existence of partial complex hadamard matrices and related combinatorial objects, Discrete Appl. Math. 102 (2000) 37–45.

[5] W. de Launey, D.M. Gordon, A comment on the hadamard conjecture, J. Combin. Theory Ser. A 95 (1) (2001) 180–184.

[6] R. Dechter, Constraint Processing, Morgan Kaufmann, 2003.

[7] D.v. Đokovic, O. Golubitsky, I.S. Kotsireas, Some new orders of hadamard and skew-hadamard matrices, J. Combin. Des. 22 (6) (2014) 270–277.

[8] I.P. Gent, C. Jefferson, I. Miguel, MINION: a fast scalable constraint solver, in: G. Brewka, S. Coradeschi, A. Perini, P. Traverso (Eds.), ECAI, IOS, Amsterdam, 2006, pp. 98–102.

[9] J. Hastad, Clique is hard to approximate within $n^{1-\epsilon}$, in: Proc. 37th Annu. Symp. Found. Comput. Sci. Burlington, 1996, pp. 627–636.

[10] K.J. Horadam, Hadamard Matrices and their Applications, Princeton University Press, 2007.

[11] N. Ito, Hadamard graphs i, Graphs Combin. 1 (1) (1985) 57–64.

[12] S. Khot, Improved inapproximability results for maxclique, chromatic number and approximate graph coloring, in: Proceedings of 42nd Annual IEEE Symposium on Foundations of Computer Science, FOCS, 2001, pp. 600–609.

[13] MINION official site, 2017, https://constraintmodelling.org/minion/.

[14] N.J.A. Sloane, The on-line encyclopedia of integer sequences, 2017, http://www2.research.att.com/~njas/hadamard.