EasyChair Preprint

№ 825

The SAT+CAS Method for Combinatorial Search with Applications to Best Matrices

Curtis Bright, Dragomir Z Djoković, Ilias Kotsireas and Vijay Ganesh

March 12, 2019

The SAT+CAS Method for Combinatorial Search with Applications to Best Matrices

Curtis Bright[1], Dragomir Ž. Đoković[1], Ilias Kotsireas[2], and Vijay Ganesh[1]

[1] University of Waterloo, Canada
[2] Wilfrid Laurier University, Canada

Abstract. In this paper, we provide an overview of the SAT+CAS method that combines satisfiability checkers (SAT solvers) and computer algebra systems (CAS) to resolve combinatorial conjectures, and present new results vis-à-vis best matrices. The SAT+CAS method is a variant of the DPLL(T) architecture, where the T solver is replaced by a CAS. We describe how the SAT+CAS method has been used to resolve many open problems from graph theory, combinatorial design theory, and number theory, showing that the method has broad applications across a variety of fields. Additionally, we apply the method to construct the largest best matrices yet known and find new skew Hadamard matrices constructed from best matrices. As a consequence of this we show that a conjecture on the existence of best matrices that was previously known to hold for $r \leq 6$ also holds for $r = 7$.

Keywords: Satisfiability checking · Combinatorial search · Symbolic computation · SAT+CAS

1 Introduction

In recent years a number of search paradigms have emerged that allow the solving of extraordinarily large problems in combinatorial mathematics. One of the most successful techniques in recent years is the "SAT paradigm" of reducing a problem into Boolean logic and then searching for a solution using a SAT solver [7]. In fact, the SAT paradigm has been so successful that it is routinely used to solve problems in areas of mathematics which don't seem to be directly connected to Boolean logic at first glance. In 2017 Heule, Kullmann, and Marek [42] summarized the state-of-the-art in combinatorial searches as follows:

> Surprisingly, SAT solving is getting so strong that indeed [using SAT solvers] seems today the best solution in most cases.

Some enormous combinatorial problems have been resolved in this way. In particular, the cube-and-conquer SAT solving paradigm [40] by Heule and Kullmann has achieved a number of striking successes including solving the Boolean Pythagorean triples problem [41] and determining the value of the fifth Schur number—a problem that resisted solution for over 100 years [39].

Briefly, in the cube-and-conquer paradigm a "look-ahead" SAT solver [44] partitions the search space into a number of independent subspaces of roughly equal difficulty called "cubes". Each cube is then solved by a SAT solver using conflict-driven clause learning [59] (possibly employing parallelization across a large number of processors) to determine if a solution to the problem exists. See Section 2 for more background on the cube-and-conquer paradigm.

Despite these impressive successes, the cube-and-conquer paradigm is not appropriate for all kinds of combinatorial problems, and in particular it would be difficult to use in problems that have properties that *cannot easily be expressed in Boolean logic*. When dealing with such problems one common approach is to employ an SMT (SAT modulo theories) solver [4,21,27,31] based on the Davis–Putnam–Logemann–Loveland algorithm (modulo theories) denoted by DPLL(T) [61] where T is a theory of first-order logic. SMT solvers can solve many problems of interest in the context of automatic program verification [9] and automatic test case generation [17]. However, modern SMT solvers only support specific theories and in this paper we are interested in solving combinatorial problems that have properties that cannot be easily expressed in those theories.

By contrast, a huge number of mathematical properties of interest can easily be expressed in *computer algebra systems* (CAS) such as Maple [6], Mathematica [75], and SageMath [70]. Indeed, a common approach for solving problems that use advanced mathematics is to write a program in the programming language of a CAS. While CAS are very impressive in solving pure math problems, they are not optimized for combinatorial problems that require both math and search.

We therefore have developed a new "SAT+CAS" method, that combines the best of both the SAT world (for search) and CAS (for math) and used this method to solve several large combinatorial problems that rely on advanced mathematics. We do this by using a combination of SAT solvers and computer algebra systems and use each system in ways that exploit its strengths. Namely, we use the SAT solver as the combinatorial search engine and use the CAS to check properties that are too difficult or cumbersome to encode into Boolean logic. The SAT+CAS paradigm is a variant of DPLL(T) and can be captured as DPLL(CAS), however, it also uses the CAS more generally. For example, we use the CAS during preprocessing in addition to using it in the DPLL(CAS) context.

As concrete examples of this paradigm we mention three problems and the properties that we checked using a CAS. See Section 3 for background on these problems and Section 4 for details of how we used the SAT+CAS method to push the state-of-the-art in these problems. We briefly mention these below:

1. The Ruskey–Savage conjecture (see [78,79]). This conjecture states that any matching of a hypercube graph can be extended to a Hamiltonian cycle. We encode the property that a set of edges is a matching in a SAT instance and check that the edges extend to a Hamiltonian cycle using a CAS.
2. Enumerating Williamson matrices (see [13,14]). Williamson matrices are square $\{\pm 1\}$-matrices that satisfy a simple arithmetical property. They also satisfy a more complicated property based on the discrete Fourier transform that we check using a CAS.

3. Enumerating complex Golay pairs (see [15,16]). Complex Golay pairs are two polynomials with coefficients in $\{\pm 1, \pm i\}$ that satisfy a simple arithmetical property. The norm of the polynomials satisfy certain bounds that we check using the nonlinear programming optimizer of a CAS.

In this paper, we further extend the success of the SAT+CAS paradigm to another class of matrices studied in combinatorial design theory known as best matrices [34]. This case study is similar to the Williamson example from above because best matrices are also known to satisfy a strict condition based on the discrete Fourier transform. However, best matrices tend to be much rarer than Williamson matrices. In particular, it is known that if circulant best matrices of order n exist then n must be of the form $r^2 + r + 1$ for some $r \geq 0$ [25].

Before this work it was known that best matrices exist in all these orders for r up to and including $r = 5$ [34] and best matrices were recently found for $r = 6$ [25]. This makes it tempting to conjecture that best matrices actually exist for all orders of the form $r^2 + r + 1$. See Section 5 for a detailed discussion of how we applied the SAT+CAS paradigm to this problem and Section 6 for details on our implementation and results.

Contributions. The main new result of this paper is that we show for the first time that best matrices exist for $r = 7$ by explicitly constructing best matrices of order 57—the largest best matrices currently known. Additionally, we use these matrices to construct new skew Hadamard matrices and perform the first published verification of the counts of best matrices given in [25,34] (see Section 6). A secondary contribution is a demonstration that the SAT+CAS method is applicable to a wide variety of fields including graph theory, combinatorial design theory, and number theory (see Section 4).

2 Previous work

Algorithmic advances such as conflict-driven clause learning (CDCL) and variable branching heuristics have spurred the "SAT revolution" [71] and resulted in a huge number of applications of SAT solvers ranging from formal verification of hardware [72] to solving Sudoku puzzles [57]. In this paper we focus on applications of SAT solvers to combinatorial search problems.

The first application of SAT solvers to combinatorial problems appears to be by McCune [60], and Stickel and Zhang [67] who in the 1990s used SAT solvers to solve a number of open Latin square and quasigroup problems. Zhang developed a solver called SATO [76] and applied it to numerous Latin square problems and observed that such an approach was just as effective as using a special-purpose solver [77]:

> In the earlier stage of our study of Latin square problems, the author wrote two special-purpose programs. After observing that these two programs could not do better than SATO, the author has not written any special-purpose search programs since then.

In the 2000s SAT solvers were also successfully applied to the branch of combinatorics known as Ramsey theory and in particular to the problem of computing van der Waerden numbers. The mathematician van der Waerden proved [73] that any r-colouring of the natural numbers must contain k numbers in arithmetic progression that are all the same colour (monochromatic). A *van der Waerden number* is the smallest value of n such that all r-colourings of $\{1, \ldots, n\}$ have a monochromatic arithmetic progression of length k.

An initial result in 2003 was by Dransfield, Marek, and Truszczyński [26] who used a SAT solver to significantly improve the lower bounds on several van der Waerden numbers. Two years later Kouril and Franco [53] used a SAT solver to find a 2-colouring of $\{1, \ldots, 1131\}$ without any monochromatic arithmetic progressions of length 6 and conjectured that it was not possible to increase the size of this set. Three years later Kouril and Paul [54] used a SAT solver to prove this, in other words they showed that all 2-colourings of $\{1, \ldots, 1132\}$ contain monochromatic arithmetic progressions of length 6.

In 2011, Heule, Kullmann, Wieringa, and Biere [43] developed the cube-and-conquer paradigm in the process of solving SAT instances that arose from computing van der Waerden numbers [3]. They found that the cube-and-conquer method performed better than any other method on these instances:

> Results on hard van der Waerden benchmarks using our basic method show reduced computational costs up to a factor 20 compared to the fastest "pure" SAT solver.

The basic idea behind the cube-and-conquer method is to combine two different SAT solving strategies, the "lookahead" and "conflict-driven" strategies. Lookahead solvers are good at making decisions at a *global* level, i.e., finding the next decision that simplifies the problem as much as possible. In contrast, conflict-driven solvers are good at solving large problems that admit a relatively short solution, i.e., ones that can be solved by making specific *local* decisions that may not be good globally but happen to work in that specific case.

The crucial insight by Heule et al. is to employ lookahead solvers to split the problem into many subproblems and then switch to conflict-driven solvers once the subproblems become simple enough. In this way a cube-and-conquer solver performs better than either a pure lookahead or pure conflict-driven solver. Furthermore the method naturally admits parallelization as the subproblems can be solved using separate processors.

The cube-and-conquer paradigm has been enormously successful. In particular, Heule, Kullmann, and Marek [41] used it to solve the Boolean Pythagorean triples problem and Heule [39] used it to find the value of the fifth Schur number—both of these problems were well-known and went unsolved for decades. SAT solvers have also been used to compute Green–Tao numbers by Kullmann [55] and solve a special case of the Erdős discrepancy conjecture by Konev and Lisitsa [48].

3 Mathematical preliminaries

In this section we describe the mathematical preliminaries necessary to understand the problems discussed in this paper. Our goal is to demonstrate that the SAT+CAS method is broadly applicable so we review several different fields.

Graph theory. The *hypercube graph* of order n is a graph on 2^n vertices where the vertices are labelled with bitstrings of length n. Two vertices are adjacent in this graph exactly when their labels differ in a single bit. A *matching* of a graph is a subset of its edges such that no two edges share a vertex. A *Hamiltonian cycle* of a graph is a path through the graph that starts and ends at the same vertex and visits each vertex exactly once.

The Ruskey–Savage conjecture says that every matching of a hypercube graph of order $n \geq 2$ can be extended into a Hamiltonian cycle of the graph [64]. This conjecture has been open for over twenty-five years.

Combinatorial design theory. A *Hadamard matrix* is a square matrix with ± 1 entries such that any two distinct rows are orthogonal. Hadamard matrices have a long history (first constructed in 1867 [68]) and applications to error-correcting codes [58], image coding [62], and techniques for statistical estimation [63]. The *Hadamard conjecture* is that Hadamard matrices exist in all orders that are multiples of four and much work has gone into constructing Hadamard matrices in as many orders as possible [38,46,47,66]. One way of constructing a Hadamard matrix of order $4n$ is to use a set of Williamson matrices of order n [50].

To define Williamson matrices we require the definition of a circulant matrix. A matrix is *circulant* if each row is equal to the previous row shifted by one element to the right (with a wrap-around). Therefore, a circulant matrix can equivalently be identified with the sequence formed by its first row. Four circulant and symmetric matrices A, B, C, D of order n are *Williamson matrices* if they have ± 1 entries and $A^2 + B^2 + C^2 + D^2$ is the scalar matrix $4nI$. The *Williamson conjecture* is that Williamson matrices exist in all orders $n \geq 1$ [37].

Best matrices are similar to Williamson matrices and will be formally defined in Section 5. One of the major differences is that best matrices can be used to construct *skew* Hadamard matrices, i.e., ones whose off-diagonal entries are anti-symmetric. Much effort has also been spent constructing skew Hadamard matrices in as many orders as possible. They are conjectured to exist for all orders of the form $4n$ [20] but the current smallest unknown order is $n = 69$ and the previous smallest unknown order $n = 47$ was solved in 2008 [22]. Constructions for other combinatorial structures of interest that rely on best matrices are given in the original paper that defined them [34].

Number theory. Two polynomials A and B with coefficients in $\{\pm 1, \pm i\}$ and of degree $n - 1$ form a *complex Golay pair* if $|A(z)|^2 + |B(z)|^2 = 2n$ for all z on the unit circle. Such polynomials (with real coefficients) were first used by Golay to solve a problem in infrared multislit spectrometry [36]. They have since been applied to an enormous number of applications in engineering (particularly in

communications [69]). They also provide extremal examples for various problems in number theory [8].

4 The SAT+CAS paradigm

It is well known that one of the drawbacks of Boolean logic is that it is not expressive enough for many domains. This was a significant impetus for the development of *SAT modulo theories* (SMT) solvers and the DPLL(T) architecture [33] that can solve problems specified in more expressive theories. A few SMT solvers have the ability to work with more mathematically complex theories such as non-linear transcendental arithmetic [18]. However, to our knowledge no SMT solvers can compute fast Fourier transforms or are optimized to handle the many fragments of mathematics supported by computer algebra systems.

Conversely, *computer algebra systems* (CAS) from the field of symbolic computation are optimized to solve hard non-linear algebraic problems, among many other fragments of mathematics. In 2015, Ábrahám [1] pointed out that the fields of symbolic computation and SMT solving have similar aims but the fields have developed mostly independently of each other:

> The research areas of SMT solving and symbolic computation are quite disconnected. On the one hand, SMT solving [...] makes use of symbolic computation results only in a rather naive way. [...] On the other hand, symbolic computation [...] does not exploit the achievements in SMT solving for efficiently handling logical fragments, using heuristics and learning to speed-up the search for satisfying solutions.

Furthermore, she made the case that these communities could mutually benefit from exploiting the achievements of the other field. To this end the SC^2 project (for symbolic computation and satisfiability checking) was started to bridge the gap between these communities [2].

Independently of the work of Ábrahám, we started developing a system called MathCheck in 2014 inspired by the DPLL(T) algorithm but replacing the theory solver with a computer algebra system. We used MathCheck to show that certain graph theoretic conjectures held up to bounds that had not previously been verified [79]. Later we applied MathCheck to find (and disprove the existence of in certain orders) Williamson matrices [14], complex Golay pairs [16], and good matrices [10].

The SAT instances that were generated in the graph theory case studies were small enough that the instances could be solved without splitting them into subproblems. However, in every subsequent problem we found that splitting was essential in order to solve the largest cases. We observed very similar behaviour to what was described by Heule [39] in the cube-and-conquer paradigm, namely, that instances could be solved much quicker once they had been split into independent subproblems. This held true even if only using a single processor and even when accounting for the time it takes to split the problem.

Our SAT+CAS method can be viewed as a special case of the cube-and-conquer paradigm with two major differences. First, we don't divide the problem into subproblems specified by cubes (a conjunction of literals). Instead, we allow our subproblems to be specified by clauses in conjunctive normal form. Second, we use a computer algebra system during both the divide and conquer phases. During the dividing phase the computer algebra system can often discard entire subproblems without even sending them to a SAT solver. For example, if the CAS finds that two subproblems are isomorphic then one can safely be discarded. However, devising a splitting method that takes advantage of the computer algebra system and performs well requires significant knowledge of the domain.

We now describe in detail the problems outlined in Sections 1 and 3 and how we applied the SAT+CAS paradigm to derive new results about each problem. We leverage some of these ideas in Section 5 and use them to construct the largest best matrices currently known.

Graph theory. The Ruskey–Savage conjecture (that every matching of a hypercube can be extended into a Hamiltonian cycle) was previously known in the orders $n = 2$, 3, and 4 [29]. Using MathCheck we showed that the conjecture also holds in the order $n = 5$ for the first time [79].

The constraint that a subset of the edges of a hypergraph forms a matching is encoded directly into Boolean logic. However, the constraint that says that such a matching can be extended into a Hamiltonian cycle is not straightforward to encode in Boolean logic—but a computer algebra system can easily test this. Therefore, whenever a satisfying assignment of the propositional constraints (i.e., a matching of the graph) is found by the SAT solver the matching is passed to a CAS to verify that it can be extended into a Hamiltonian cycle.

If the CAS finds that the given matching can not extend to a Hamiltonian cycle this provides a counterexample of the conjecture. Otherwise, the CAS provides to the SAT solver a clause that blocks this matching from being considered in the future. It's also possible for the CAS to provide clauses that block other similar matchings (e.g., matchings generated via an automorphism of the graph) and we showed that this was beneficial to the performance of the solver [78].

Combinatorial design theory. Prior to our work, exhaustive searches for Williamson matrices had been performed in all odd orders $n \leq 59$ [45] and all even orders $n \leq 18$ [49]. These searches discovered that Williamson matrices don't exist in the orders $n = 35$, 47, 53, and 59 but exist in all other orders that were searched. Using MathCheck we were able to provide exhaustive searches for all orders $n \leq 70$ divisible by 2 or 3 (finding over 100,000 new sets of Williamson matrices) [12,14] and verified the counterexample $n = 35$ [11].

Using arithmetic circuits it is possible to generate a SAT instance that specifies that Williamson matrices exist in order n. However, this approach was only able to find Williamson matrices for orders up to $n = 30$. To scale up to $n = 70$ we found that it was essential to use a divide-and-conquer approach and use CAS functionality in both the divide and conquer phases.

First, we give an overview of the divide phase. The first property that is useful in this regard is the fact that Williamson matrices satisfy

$$\mathrm{sum}(A)^2 + \mathrm{sum}(B)^2 + \mathrm{sum}(C)^2 + \mathrm{sum}(D)^2 = 4n$$

where $\mathrm{sum}(X)$ denotes the rowsum of the first row of X. (We associate a circulant matrix X with the sequence formed by its first row.) We use a computer algebra system to solve the equation $x^2 + y^2 + u^2 + v^2 = 4n$ in integers and each solution provides one subproblem, namely, the subproblem of finding a set of Williamson matrices of order n with rowsums (x, y, u, v). This typically splits each order into a few subproblems; to further divide the problem we use the properties of *sequence compression* [24].

For concreteness, suppose n is even and $n = 2m$. Then Williamson matrices can be "compressed" into matrices of order m by adding together the entries that are separated by exactly m entries in each row. For example, the compression of the row $[1, -1, 1, 1, -1, -1]$ is $[2, -2, 0]$. We generate further subproblems using compressions; each subproblem corresponds to finding a set of Williamson matrices that compresses to a given $\{\pm 2, 0\}$-sequence. The reason this method of dividing is so effective is because there are strong filtering theorems that a CAS can use to determine that most subproblems of this form are unsatisfiable without even using a SAT solver. An example of these filtering theorems is described below (in the context of the conquer phase).

Second, we give an overview of the conquer phase. In this phase a SAT solver receives a number of independent SAT instances that encode one subproblem that was generated in the divide phase. Using an off-the-shelf SAT solver in this stage allowed us to scale to order $n = 35$ (and in particular verify that 35 is the smallest counterexample of the Williamson conjecture [11]). However, using a CAS in the conquer phase was found to be orders of magnitude faster and allowed us to scale to $n = 70$.

The reason that using a CAS produces such a dramatic improvement is because it allows the usage of filtering theorems that are very strong—but the theorems cannot easily be directly encoded into Boolean logic. As an example, it is known that if $A = [a_0, \ldots, a_{n-1}]$ is the first row of a Williamson matrix then the bound

$$\left| \sum_{j=0}^{n-1} a_j \exp\bigl(2\pi\sqrt{-1}jk/n\bigr) \right|^2 \leq 4n$$

holds for all integers k. This is a very strict bound that the vast majority of A will fail to satisfy for some k. Furthermore, it is very efficient to test because the values on the left form the *power spectral density* of A and can be quickly computed using a fast Fourier transform (e.g., using the DFT and PowerSpectrum functions of the computer algebra system Maple).

In the conquer phase the SAT solver proceeds as normal until a partial satisfying assignment of the propositional constraints are such that the values of A can be determined. At this point A is passed to a CAS which computes its power spectrum. If the power spectral density bound is violated then a conflict

clause is returned to the SAT solver that blocks this A from being considered in the future. The same condition is also checked with B, C, and D.

Number theory. All complex Golay pairs up to $n = 19$ were enumerated in [19] and it was conjectured that such pairs do not exist for $n = 23$ based on a partial search. This conjecture was proven in [28] where a complete enumeration was performed up to $n = 28$. However, this result had never been independently verified. Using MathCheck we performed the first independent verification of this result [15] by explicitly finding all complex Golay pairs for $n \leq 28$, and further provided a complete enumeration of all complex Golay pairs up to $n = 28$.

At first it is not even obvious that a search for complex Golay pairs of order n could be translated into SAT, since there are an infinite number of z on the unit circle. In fact, using arithmetic circuits and other properties of complex Golay pairs it is possible to generate a SAT instance that specifies that complex Golay pairs exist in order n. However, in our experience these instances could only be solved up to $n = 16$ before incorporating a CAS. Similar to the previous case study we employ a divide and conquer approach and use CAS functionality in both phases.

In the divide phase we perform a search for all possible polynomials A that could appear as a member of a complex Golay pair. A number of properties that A must satisfy are used as filtering criteria, the main one being that $|A(z)|^2 \leq 2n$ for all z on the unit circle. To test this bound we find the maximum of the non-linear function $|A(z)|^2$ for z on the unit circle; for example, this can be done with the Maple command NLPSolve.

In the conquer phase we solve a SAT instance for each possible A that was found in the dividing phase; a satisfying assignment of such an instance will produce a B such that (A, B) form a complex Golay pair. To do this we use the relationship $N_A + N_B = [2n, 0, \ldots, 0]$ where N_X is the *autocorrelation* of the coefficients of X. For example, N_B can be computed with the Maple command AutoCorrelation once the coefficients of B are known. An important optimization is that most values of N_B can be computed with only partial knowledge of B; this allows one to learn shorter conflict clauses based on only a partial assignment of the SAT instance.

5 Best matrices

We now apply our experience using MathCheck on the three case studies described in Section 4 to a new problem, namely, the problem of finding best matrices from combinatorial design theory. Best matrices are similar to Williamson matrices but exist in fewer orders; in fact, if best matrices exist in order n then n must be of the form $r^2 + r + 1$. The best known result [25] is that best matrices exist for all $r \leq 6$ and we use MathCheck to extend this result to $r \leq 7$. It is unknown if best matrices exist for any $r \geq 8$.

5.1 Background

Let X be a square matrix of order n. Recall that X is *symmetric* if $x_{i,j} = x_{j,i}$ for all indices (i, j), *skew* if $x_{i,j} = -x_{j,i}$ for all indices $i \neq j$, and *circulant* if $x_{i,j} = x_{i+1,j+1}$ for all indices (i, j) (reduced mod n if necessary).

Four matrices A, B, C, D of order n with ± 1 entries and positive diagonal entries are *best matrices* if they satisfy the following three axioms:

(1) A, B, and C are skew and D is symmetric.
(2) A, B, C, and D are pairwise commutative.
(3) $AA^T + BB^T + CC^T + DD^T$ is the scalar matrix $4nI$.

An example of best matrices of order three (where "+" denotes 1 and "−" denotes −1) are

$$A = \begin{bmatrix} + & - & + \\ + & + & - \\ - & + & + \end{bmatrix} \quad B = \begin{bmatrix} + & - & + \\ + & + & - \\ - & + & + \end{bmatrix} \quad C = \begin{bmatrix} + & - & + \\ + & + & - \\ - & + & + \end{bmatrix} \quad D = \begin{bmatrix} + & + & + \\ + & + & + \\ + & + & + \end{bmatrix}.$$

In this paper we will only consider *circulant* best matrices in which case condition (2) is always satisfied. Furthermore, condition (1) is easy to enforce since, for example, once the first half of the entries in the first row are known they uniquely determine the values of the entries in the second half. This still leaves an enormous search space, however. Since there are $(n-1)/2$ undetermined entries in each matrix a naive brute-force search would check $2^{4(n-1)/2} = 4^{n-1}$ quadruples—making the search space for best matrices of order 57 about a quarter of a billion times larger than the search space for best matrices of order 43 (the previous largest best matrices known). Nevertheless, we were successful in our search for best matrices of order 57 by employing a number of powerful filtering theorems and using SAT solvers to search the remaining space.

5.2 Equivalence operations

There are three operations on best matrices A, B, C, D that can be used to produce a new equivalent set of best matrices:

1. Reorder A, B, and C in any way.
2. Apply the operation $i \mapsto -i \bmod n$ to the indices of the first row of A, B, or C. (Since D is symmetric such an operation has no effect on it.)
3. Apply an automorphism of the cyclic group $\mathbb{Z}_n$ to the indices of the first rows of A, B, C, and D simultaneously.

Such equivalence operations are well-known [25]. The "cyclic shift" operation is sometimes also considered an equivalence operation but we did not use it as it generally disturbs the symmetry and anti-symmetry of the matrices.

5.3 Divide phase

Our aim in this phase is to split the problem of finding best matrices of given order n into subproblems such that each subproblem is easy enough to be solved with a SAT solver (coupled with a CAS).

For concreteness we will focus on the case $n = 57$ and use the fact that $57 = 3 \cdot 19$ which allows us to "3-compress" the rows of best matrices to form compressed best matrices of order 19. If X is a sequence of length 57 then its compression $\bar{X}$ is a sequence of length 19 such that its kth entry is

$$\bar{x}_k \coloneqq x_k + x_{k+19} + x_{k+2\cdot 19} \qquad \text{for } 0 \le k < 19.$$

The reason why compression is so important is because of the following property that the compressions of best matrices must satisfy [24]:

$$\mathrm{PSD}_{\bar{A}}(k) + \mathrm{PSD}_{\bar{B}}(k) + \mathrm{PSD}_{\bar{C}}(k) + \mathrm{PSD}_{\bar{D}}(k) = 4 \cdot 57 \qquad \text{for all } k. \qquad (*)$$

Here $\mathrm{PSD}_{\bar{X}}$ denotes the *power spectral density* of $\bar{X}$ defined by

$$\mathrm{PSD}_{\bar{X}}(k) \coloneqq \left| \sum_{j=0}^{18} \bar{x}_j \exp\bigl(2\pi\sqrt{-1}jk/19\bigr) \right|^2.$$

Our implementation (see Section 6 for details) finds 15,178 inequivalent possible quadruples $(\bar{A}, \bar{B}, \bar{C}, \bar{D})$ that satisfy the necessary relationship $(*)$. For each quadruple we generate a SAT instance with the $2n - 2$ variables $\{\, a_i, b_i, c_i, d_i : 1 \le i < (n+1)/2 \,\}$. The remaining entries are determined via the relationships

$$a_k = -a_{n-k}, \quad b_k = -b_{n-k}, \quad c_k = -c_{n-k}, \quad d_k = d_{n-k} \qquad \text{for } k \neq 0.$$

For clarity we will use variables with indices greater than $(n+1)/2$ with the understanding they refer to variables in our SAT instance using these relationships. Variables will be assigned *true* when they represent the entry 1 and *false* when they represent the entry -1 (by abuse of notation we use the same variable name for both but it will be clear from context if the variable is an integer or a Boolean).

Each of the 15,178 possible compressions will specify a single independent SAT subproblem. This is achieved by encoding the compression constraints in conjunctive normal form. Because the sum of three ± 1 entries must be ± 3 or ± 1 these constraints come in four forms.

The first form is when $a_k + a_{k+19} + a_{k+2\cdot 19} = 3$. In this case, we add the cube $a_k \wedge a_{k+19} \wedge a_{k+2\cdot 19}$ to the SAT subproblem. The second form is when $a_k + a_{k+19} + a_{k+2\cdot 19} = 1$. In this case, we add

$$(a_k \vee a_{k+19}) \wedge (a_k \vee a_{k+2\cdot 19}) \wedge (a_{k+19} \vee a_{k+2\cdot 19}) \wedge (\neg a_k \vee \neg a_{k+19} \vee \neg a_{k+2\cdot 19})$$

to the SAT subproblem. The cases with -1 and -3 are handled in the same way with the polarity of the literals in the clauses reversed. We also add similar clauses for the entries of B, C, and D.

5.4 Conquer phase

Our aim in this phase is to solve the subproblems generated in the dividing phase. To do this, we employ a SAT solver with a programmatic interface [32] that allows it to learn conflict clauses by querying a CAS.[1] The property that the CAS checks is the uncompressed form of $(*)$, namely,

$$\mathrm{PSD}_A(k) + \mathrm{PSD}_B(k) + \mathrm{PSD}_C(k) + \mathrm{PSD}_D(k) = 4n \qquad \text{for all } k.$$

Note that although this condition can only be verified to hold once all entries of A, B, C, D are known, in many cases it can be verified to *not* hold with only partial information. In particular, since PSD values are non-negative we must have the *PSD criterion*

$$\sum_{X \in S} \mathrm{PSD}_X(k) \leq 4n$$

where S is any subset of $\{A, B, C, D\}$.

In particular, if a partial assignment specifies enough entries such that the PSD criterion is violated then a conflict clause is learned that tells the SAT solver to avoid that partial assignment in the future. An important optimization is to choose S in the PSD criterion to be as small as possible. For example, if both $S = \{A, B\}$ and $S = \{C\}$ violate the PSD criterion we prefer the latter because in that case we learn a shorter conflict clause. In the latter case the learned clause would say that at least one variable $\{\, c_i : 0 \leq i < n \,\}$ has to be assigned differently to its current assignment.

Additionally, the entries of best matrices can be shown to satisfy certain constraints similar to constraints that Williamson matrices [74], good matrices [10], and the coefficients of complex Golay pairs [15] satisfy. In the supplementary material we show that the entries of best matrices satisfy the relationship $a_k b_k c_k d_k a_{2k} b_{2k} c_{2k} = -1$ for $k \neq 0$ (with indices reduced mod n if necessary). Because of the anti-symmetry of A, B, and C when $k = n/3$ the product constraint reduces to $d_k = 1$ and in this case can be encoded as a unit clause. In general we encode the product constraint in SAT by breaking it up into the six constraints

$$x_0 = a_k b_k, \quad x_1 = x_0 c_k, \quad x_2 = x_1 d_k, \quad x_3 = x_2 a_{2k}, \quad x_4 = x_3 b_{2k}, \quad x_5 = x_4 c_{2k}$$

with $x_5 = -1$, where the x_i are new variables. For example the first of these constraints is represented in conjunctive normal form as

$$(x_0 \vee a_k \vee b_k) \wedge (\neg x_0 \vee \neg a_k \vee b_k) \wedge (x_0 \vee \neg a_k \vee \neg b_k) \wedge (\neg x_0 \vee a_k \vee \neg b_k)$$

and the others are represented similarly.

[1] A programmatic SAT solver is simply a variant of DPLL(T) and similar to a "SAT modulo SAT" solver [5]. The key difference between programmatic SAT and DPLL(T) is that the T solver in programmatic SAT can be specialized to individual formulas (like an advice string in a non-uniform computation model), whereas DPLL(T) was envisioned with the T solver being a decision procedure for an entire theory.

6 Implementation and results

We implemented the divide and conquer phases described in Section 5 in our SAT+CAS system MathCheck. Our code is available from our website `uwaterloo.ca/mathcheck` along with more details of our other case studies.

In the divide phase we wrote some custom C++ code to generate all possible compressions of best matrices. This code takes advantage of the well-known fact that the rowsums of the first rows of A, B, C must be 1 and the squared rowsum of the first row of D must be $4n-3$ [34]. It follows that the rowsum of the first row of D is $\pm(2r+1)$ where $n = r^2+r+1$. In fact, the sign of $\mathrm{sum}(D)$ is positive when $r \equiv 0, 1 \pmod 4$ and negative otherwise (see supplementary material for details). Thus, for $r = 7$ we have that $\mathrm{sum}(D) = -15$.

We now employ a brute-force method to find all possibilities for the first rows of best matrices of order n. Taking into account the matrices are skew or symmetric there are $2^{(n-1)/2}$ possibilities for each of A, B, C, and D. The majority of possibilities have a PSD value larger than $4n$ and can therefore be ignored. To further cut down on possibilities we also discard possibilities that will lead to equivalent best matrices using the equivalence operations of Section 5.2. In particular, we apply operation 2 to the possibilities for B and C and operation 3 to the possibilities for A.

We then 3-compress the possibilities for $n = 57$, finding 2748 possibilities for $\bar{A}$, 24,674 possibilities for $\bar{B}$ and $\bar{C}$, and 7999 possibilities for $\bar{D}$. These possibilities now need to be joined into quadruples. First, using brute-force we make a list of the possible pairs $(\bar{A}, \bar{B})$ and $(\bar{C}, \bar{D})$; we find about 12 million possibilities for the former and 40 million possibilities for the latter. Then using the string sorting and matching algorithm described in [51] we find all quadruples whose PSD values sum to $4n$. After this step has completed we find 91,190 possible quadruples $(\bar{A}, \bar{B}, \bar{C}, \bar{D})$ of which 15,178 are inequivalent using the equivalence operations of Section 5.2. Each of these quadruples will form one independent subproblem (using the SAT encoding described in Section 5.3) to be solved in the conquer phase.

For efficiency the PSD values were computed using the C library FFTW [30] that can very efficiently compute discrete Fourier transforms. Since FFTW uses floating-point arithmetic we would only discard possibilities whose PSD values could be shown to be larger than $4n + \epsilon$ where ϵ is larger than the precision of the fast Fourier transform that was used.

In the conquer phase we solved our SAT instances using a programmatic version of the SAT solver MapleSAT [56]. The "callback" function was implemented as described in Section 5.4 with a conflict clause being learnt whenever enough of a partial assignment is known so that the PSD criterion can be shown to be violated. Again for efficiency we used the C library FFTW for computing the PSD values.

As previously mentioned the orders of best matrices must be of the form $r^2 + r + 1$ for $r \geq 0$. The case $r = 7$ is currently the smallest open case and MathCheck is successfully able to solve this case. For completeness, we apply our method to smaller orders. Orders of the form $r^2 + r + 1$ are prime for

$r \in \{1, 2, 3, 5, 6, 8\}$ and therefore do not have a nontrivial compression factor. Despite this, the cases $r = 1$, 2, 3, 4 can be be solved in under a second using the method described in Section 5 with no compression (i.e., compression by 1). Furthermore, the case $r = 5$ can be solved in about 5 seconds and the case $r = 6$ can be solved in about 50 minutes.

We ran the case $r = 7$ on a cluster of 64-bit Opteron 2.2GHz and Xeon 2.6GHz processors running CentOS 6 using compression by a factor of 3. In this case MathCheck requires about 20 minutes to perform the dividing phase and about 162 hours to perform the conquer phase. These times measure the total amount of CPU time, though the conquer phase took under an hour of real time when parallelized across 200 cores.

Three inequivalent sets of best matrices of order 57 were found in the $r = 7$ case. We used the Goethals–Seidel construction [35] to construct new skew Hadamard matrices of order $4 \cdot 57$ using these best matrices and give one example in Figure 1. (Skew Hadamard matrices of order $4 \cdot 57$ have long been known [65] but these are the first ones constructed using best matrices.) Explicit representations of the new best matrices that we constructed can be found on our website `uwaterloo.ca/mathcheck`, are archived at `doi.org/10.5281/zenodo.2582592`, and are available in the supplementary material.

Let B_r denote the number of inequivalent sets of best matrices of order $r^2 + r + 1$. Our results determine the value of B_r for $r \leq 7$:

$$B_0 = 1, \quad B_1 = 1, \quad B_2 = 2, \quad B_3 = 2, \quad B_4 = 7, \quad B_5 = 2, \quad B_6 = 5, \quad B_7 = 3.$$

The value of B_7 is new, the value of B_6 was found in [25], and the other values were given in [34]. Our counts differ from those of [34] only because that work did not use equivalence operation 2. For example, for $r = 4$ they find twenty-one sets of best matrices but each of the seven sets that we found is equivalent to one of the twenty-one previously found. The counts up to $r = 5$ also appear in [23,52] but these works did not verify the counts. To our knowledge we have performed the first published verification.

7 Conclusions and future work

We have described a "SAT+CAS" paradigm, building on DPLL(T), that is able to solve hard combinatorial problems that require *both* clever search routines (à la SAT) and efficient procedures for complex mathematics outside the scope of traditional SMT theory solvers (e.g., Fourier transforms in CAS). As a demonstration of the power and flexibility of the method we have outlined how it was used to improve the state-of-the-art on three separate class of problems from graph theory, number theory, and combinatorics, as well as its application to construct new skew Hadamard matrices of order $4 \cdot 57$. The naive search space for such an object is $2^{57 \cdot 56/2} \approx 10^{480}$ which is totally impractical to search using brute-force. Instead, we use a number of mathematical properties of best matrices to greatly constrain the search space. However, it would be difficult to execute the search using either SAT solvers or computer algebra systems in isolation:

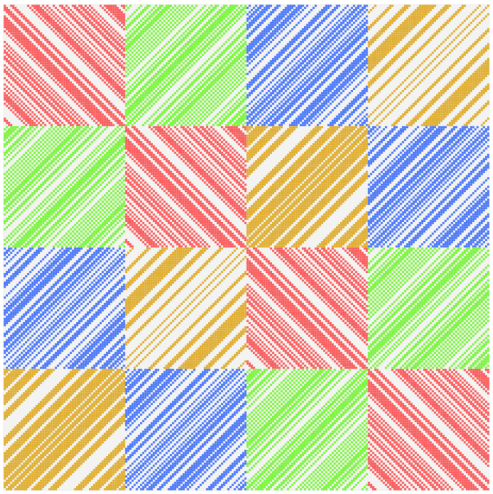

Fig. 1. A new skew Hadamard matrix of order $4 \cdot 57 = 228$ constructed using the Goethals–Seidel construction and best matrices of order 57. The coloured entries represent 1, the grey entries represent -1, and each best matrix is coloured differently to more clearly show the structure of the matrix. (The matrices B, C, and D appear reflected in the Goethals–Seidel construction.)

SAT solvers would not be able to exploit the complex mathematical properties and computer algebra systems would not be able to exploit the efficient search routines of SAT solvers, by themselves.

We additionally find inspiration from the cube-and-conquer paradigm of Heule et al. [43]. Since open problems (like finding best matrices of order 57) typically have extremely large search spaces they are usually not easy to solve using a sequential SAT solver. To deal with this we developed a method of dividing instances into multiple independent subproblems. In particular, we divide the search for best matrices of order 57 into 15,178 subproblems such that each subproblem can be solved in about a minute using a SAT solver augmented with a domain-specific method of generating conflicts.

Heule, Kullmann, and Marek [42] point out that there are essentially three kinds of solvers that are currently used for solving large combinatorial problems: special-purpose solvers, constraint satisfaction solvers, and SAT solvers. We believe that SAT+CAS solvers have now proven themselves as an effective way of introducing the reasoning of special-purpose solvers and computer algebra systems into SAT solving for hard problems from many areas of mathematics. Going forward, we expect that SAT+CAS solvers will become essential for solving the largest combinatorial problems that incorporate sophisticated mathematical properties. For example, [42] points out that searching for finite projective planes (a special kind of combinatorial design) has currently only been done using special-purpose solvers. These kinds of problems are ripe for attack using the SAT+CAS paradigm.

References

1. Ábrahám, E.: Building bridges between symbolic computation and satisfiability checking. In: Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation. pp. 1–6. ACM (2015)
2. Ábrahám, E., Abbott, J., Becker, B., Bigatti, A.M., Brain, M., Buchberger, B., Cimatti, A., Davenport, J.H., England, M., Fontaine, P., Forrest, S., Griggio, A., Kroening, D., Seiler, W.M., Sturm, T.: SC^2: Satisfiability checking meets symbolic computation. Intelligent Computer Mathematics: Proceedings CICM pp. 28–43 (2016)
3. Ahmed, T., Kullmann, O., Snevily, H.: On the van der Waerden numbers $w(2; 3, t)$. Discrete Applied Mathematics **174**, 27–51 (2014)
4. Barrett, C., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, chap. 26, pp. 825–885. IOS Press (2009)
5. Bayless, S., Val, C.G., Ball, T., Hoos, H.H., Hu, A.J.: Efficient modular SAT solving for IC3. In: 2013 Formal Methods in Computer-Aided Design. pp. 149–156. IEEE (2013)
6. Bernardin, L., Chin, P., DeMarco, P., Geddes, K.O., Hare, D.E.G., Heal, K.M., Labahn, G., May, J.P., McCarron, J., Monagan, M.B., Ohashi, D., Vorkoetter, S.M.: Maple programming guide (2018)
7. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
8. Borwein, P.: Barker polynomials and Golay pairs. In: Computational Excursions in Analysis and Number Theory, pp. 109–119. CMS Books in Mathematics, Springer-Verlag New York (2002)
9. Botinčan, M., Parkinson, M., Schulte, W.: Separation logic verification of C programs with an SMT solver. Electronic Notes in Theoretical Computer Science **254**, 5–23 (2009)
10. Bright, C., Đoković, D., Kotsireas, I., Ganesh, V.: A SAT+CAS approach to finding good matrices: New examples and counterexamples. In: Thirty-Third AAAI Conference on Artificial Intelligence. AAAI Press (2019)
11. Bright, C., Ganesh, V., Heinle, A., Kotsireas, I., Nejati, S., Czarnecki, K.: MathCheck2: A SAT+CAS verifier for combinatorial conjectures. In: International Workshop on Computer Algebra in Scientific Computing. pp. 117–133. Springer (2016)
12. Bright, C., Kotsireas, I., Ganesh, V.: Applying computer algebra systems and SAT solvers to the Williamson conjecture. arXiv preprint arXiv:1804.01172 (2018)
13. Bright, C., Kotsireas, I., Ganesh, V.: A SAT+CAS method for enumerating Williamson matrices of even order. In: Thirty-Second AAAI Conference on Artificial Intelligence. pp. 6573–6580. AAAI Press (2018)
14. Bright, C., Kotsireas, I., Ganesh, V.: The SAT+CAS paradigm and the Williamson conjecture. ACM Communications in Computer Algebra **52**(3), 82–84 (2018)
15. Bright, C., Kotsireas, I., Heinle, A., Ganesh, V.: Complex Golay pairs up to length 28: A search via computer algebra and programmatic SAT. `https://doi.org/10.13140/RG.2.2.29639.14246` (2018)
16. Bright, C., Kotsireas, I., Heinle, A., Ganesh, V.: Enumeration of complex Golay pairs via programmatic SAT. In: Proceedings of the 2018 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2018, New York, NY, USA, July 16–19, 2018. pp. 111–118 (2018)

17. Cadar, C., Ganesh, V., Pawlowski, P.M., Dill, D.L., Engler, D.R.: EXE: Automatically generating inputs of death. ACM Transactions on Information and System Security (TISSEC) **12**(2), 1–38 (2008)
18. Cimatti, A., Griggio, A., Irfan, A., Roveri, M., Sebastiani, R.: Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions. ACM Transactions on Computational Logic (TOCL) **19**(3), 19:1–19:52 (2018)
19. Craigen, R., Holzmann, W., Kharaghani, H.: Complex Golay sequences: structure and applications. Discrete mathematics **252**(1-3), 73–89 (2002)
20. Craigen, R., Kharaghani, H.: Hadamard matrices and Hadamard designs. In: Colbourn, C.J., Dinitz, J.H. (eds.) Handbook of Combinatorial Designs, pp. 273–280. Chapman and Hall, CRC (2007)
21. de Moura, L., Bjørner, N.: The Z3 theorem prover. `https://github.com/Z3Prover` (2008)
22. Đoković, D.Ž.: Skew-Hadamard matrices of orders 188 and 388 exist. International Mathematical Forum **3**(22), 1063–1068 (2008)
23. Đoković, D.Ž.: Supplementary difference sets with symmetry for Hadamard matrices. Operators and Matrices **3**(4), 557–569 (2009)
24. Đoković, D.Ž., Kotsireas, I.S.: Compression of periodic complementary sequences and applications. Designs, Codes and Cryptography **74**(2), 365–377 (2015)
25. Đoković, D.Ž., Kotsireas, I.S.: Goethals–Seidel difference families with symmetric or skew base blocks. Mathematics in Computer Science **12**(4), 373–388 (2018)
26. Dransfield, M.R., Marek, V.W., Truszczyński, M.: Satisfiability and computing van der Waerden numbers. In: Giunchiglia, E., Tacchella, A. (eds.) Theory and Applications of Satisfiability Testing. SAT 2003, pp. 1–13. Springer, Berlin, Heidelberg (2003)
27. Dutertre, B., de Moura, L.: The Yices SMT solver. `http://yices.csl.sri.com/` (2006)
28. Fiedler, F.: Small Golay sequences. Advances in Mathematics of Communications **7**(4) (2013)
29. Fink, J.: Perfect matchings extend to Hamilton cycles in hypercubes. Journal of Combinatorial Theory, Series B **97**(6), 1074–1076 (2007)
30. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proceedings of the IEEE **93**(2), 216–231 (2005)
31. Ganesh, V., Dill, D.L.: A decision procedure for bit-vectors and arrays. In: Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007, Proceedings. pp. 519–531 (2007)
32. Ganesh, V., O'Donnell, C.W., Soos, M., Devadas, S., Rinard, M.C., Solar-Lezama, A.: Lynx: A programmatic SAT solver for the RNA-folding problem. In: Cimatti, A., Sebastiani, R. (eds.) Theory and Applications of Satisfiability Testing – SAT 2012, pp. 143–156. Springer, Berlin, Heidelberg (2012)
33. Ganzinger, H., Hagen, G., Nieuwenhuis, R., Oliveras, A., Tinelli, C.: DPLL(T): Fast decision procedures. In: International Conference on Computer Aided Verification. pp. 175–188. Springer (2004)
34. Georgiou, S., Koukouvinos, C., Seberry, J.: On circulant best matrices and their applications. Linear and Multilinear Algebra **48**(3), 263–274 (2001)
35. Goethals, J., Seidel, J.: A skew Hadamard matrix of order 36. Journal of the Australian Mathematical Society **11**(3), 343–344 (1970)
36. Golay, M.J.: Multi-slit spectrometry. JOSA **39**(6), 437–444 (1949)
37. Golomb, S.W., Baumert, L.D.: The search for Hadamard matrices. The American Mathematical Monthly **70**(1), 12–17 (1963)

38. Hedayat, A., Wallis, W.D., et al.: Hadamard matrices and their applications. The Annals of Statistics **6**(6), 1184–1238 (1978)
39. Heule, M.J.H.: Schur number five. In: Thirty-Second AAAI Conference on Artificial Intelligence. pp. 6598–6606. AAAI Press (2018)
40. Heule, M.J.H., Kullmann, O.: The science of brute force. Communications of the ACM **60**(8), 70–79 (2017)
41. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer. In: Creignou, N., Le Berre, D. (eds.) Theory and Applications of Satisfiability Testing – SAT 2016. pp. 228–245. Springer International Publishing, Cham (2016)
42. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving very hard problems: Cube-and-conquer, a hybrid SAT solving method. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. pp. 4864–4868 (2017)
43. Heule, M.J.H., Kullmann, O., Wieringa, S., Biere, A.: Cube and conquer: Guiding CDCL SAT solvers by lookaheads. In: Haifa Verification Conference. pp. 50–65. Springer (2011)
44. Heule, M.J.H., van Maaren, H.: Look-ahead based SAT solvers. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of satisfiability, chap. 5, pp. 155–184. IOS Press (2009)
45. Holzmann, W.H., Kharaghani, H., Tayfeh-Rezaie, B.: Williamson matrices up to order 59. Designs, Codes and Cryptography **46**(3), 343–352 (2008)
46. Horadam, K.J.: Hadamard matrices and their applications. Princeton university press (2012)
47. Kharaghani, H., Tayfeh-Rezaie, B.: A Hadamard matrix of order 428. Journal of Combinatorial Designs **13**(6), 435–440 (2005)
48. Konev, B., Lisitsa, A.: A SAT attack on the Erdős discrepancy conjecture. In: Sinz, C., Egly, U. (eds.) Theory and Applications of Satisfiability Testing – SAT 2014, pp. 219–226. Springer, Cham (2014)
49. Kotsireas, I.S., Koukouvinos, C.: Constructions for Hadamard matrices of Williamson type. Journal of Combinatorial Mathematics and Combinatorial Computing **59**, 17–32 (2006)
50. Kotsireas, I.S., Koukouvinos, C.: Hadamard matrices of Williamson type: A challenge for computer algebra. Journal of Symbolic Computation **44**(3), 271–279 (2009)
51. Kotsireas, I.S., Koukouvinos, C., Seberry, J.: Weighing matrices and string sorting. Annals of Combinatorics **13**(3), 305–313 (2009)
52. Koukouvinos, C., Stylianou, S.: On skew-Hadamard matrices. Discrete Mathematics **308**(13), 2723–2731 (2008)
53. Kouril, M., Franco, J.: Resolution tunnels for improved SAT solver performance. In: Bacchus, F., Walsh, T. (eds.) Theory and Applications of Satisfiability Testing. SAT 2005. pp. 143–157. Springer, Berlin, Heidelberg (2005)
54. Kouril, M., Paul, J.L.: The van der Waerden number $W(2, 6)$ is 1132. Experimental Mathematics **17**(1), 53–61 (2008)
55. Kullmann, O.: Green-Tao numbers and SAT. In: Strichman, O., Szeider, S. (eds.) Theory and Applications of Satisfiability Testing – SAT 2010, pp. 352–362. Springer, Berlin, Heidelberg (2010)
56. Liang, J.H., Govind V.K., H., Poupart, P., Czarnecki, K., Ganesh, V.: An empirical study of branching heuristics through the lens of global learning rate. In: Gaspers, S., Walsh, T. (eds.) Theory and Applications of Satisfiability Testing – SAT 2017, pp. 119–135. Springer, Cham (2017)

57. Lynce, I., Ouaknine, J.: Sudoku as a SAT problem. In: 9th International Symposium on Artificial Intelligence and Mathematics (2006)
58. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes, vol. 16. Elsevier (1977)
59. Marques-Silva, J., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of satisfiability, chap. 4, pp. 131–153. IOS Press (2009)
60. McCune, W.: A Davis–Putnam program and its application to finite first-order model search: Quasigroup existence problems. Tech. rep., Argonne National Laboratory (1994)
61. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). Journal of the ACM **53**(6), 937–977 (2006)
62. Pratt, W.K., Kane, J., Andrews, H.C.: Hadamard transform image coding. Proceedings of the IEEE **57**(1), 58–68 (1969)
63. Rao, J., Shao, J.: On balanced half-sample variance estimation in stratified random sampling. Journal of the American Statistical Association **91**(433), 343–348 (1996)
64. Ruskey, F., Savage, C.: Hamilton cycles that extend transposition matchings in Cayley graphs of S_n. SIAM Journal on Discrete Mathematics **6**(1), 152–166 (1993)
65. Seberry, J.: On skew Hadamard matrices. Ars Combinatoria **6**, 255–275 (1978)
66. Seberry, J., Yamada, M.: Hadamard matrices, sequences, and block designs. Contemporary design theory: a collection of surveys pp. 431–560 (1992)
67. Stickel, M.E., Zhang, H.: First results of studying quasigroup identities by rewriting techniques. In: Proceedings of Workshop on Automated Theorem Proving in conjunction with FGCS. pp. 16–23 (1994)
68. Sylvester, J.J.: Thoughts on inverse orthogonal matrices, simultaneous signsuccessions, and tessellated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work, and the theory of numbers. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **34**(232), 461–475 (1867)
69. Taghavi, M., Zahraei, M.: On the autocorrelations of ± 1 polynomials. Journal of Mathematical Extension **1**(2), 139–147 (2007)
70. The Sage Development Team: Sage tutorial, release 8.6 (2019)
71. Vardi, M.Y.: Symbolic techniques in propositional satisfiability solving. In: Kullmann, O. (ed.) Theory and Applications of Satisfiability Testing - SAT 2009. pp. 2–3. Springer, Berlin, Heidelberg (2009)
72. Vizel, Y., Weissenbacher, G., Malik, S.: Boolean satisfiability solvers and their applications in model checking. Proceedings of the IEEE **103**(11), 2021–2035 (2015)
73. van der Waerden, B.: Beweis einer baudetschen vermutung. Nieuw Arch. Wisk. **19**, 212–216 (1927)
74. Williamson, J.: Hadamard's determinant theorem and the sum of four squares. Duke Mathematical Journal **11**(1), 65–81 (1944)
75. Wolfram, S.: The Mathematica book, fifth edition (2003)
76. Zhang, H.: Specifying Latin square problems in propositional logic. In: Veroff, R. (ed.) Automated Reasoning and Its Applications: Essays in Honor of Larry Wos, pp. 115–146. MIT Press (1997)
77. Zhang, H.: Combinatorial designs by SAT solvers. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, chap. 17, pp. 533–568. IOS Press (2009)

78. Zulkoski, E., Bright, C., Heinle, A., Kotsireas, I., Czarnecki, K., Ganesh, V.: Combining SAT solvers with computer algebra systems to verify combinatorial conjectures. Journal of Automated Reasoning **58**(3), 313–339 (2017)
79. Zulkoski, E., Ganesh, V., Czarnecki, K.: MathCheck: A math assistant via a combination of computer algebra systems and SAT solvers. In: International Conference on Automated Deduction. pp. 607–622. Springer, Cham (2015)

The SAT+CAS Method for Combinatorial Search with Applications to Best Matrices

Supplementary Material

Let A, B, C, D be a set of circulant best matrices of order $n = r^2 + r + 1$ (note that n must be odd). As described in Section 6 the rowsums of the first rows of A, B, C are 1 and the rowsum of the first row of D is $\pm(2r + 1)$ where the sign of $\mathrm{sum}(D)$ is positive when $r \equiv 0, 1 \pmod 4$ and negative otherwise.

Proof. Since the matrix A is skew we have $a_i + a_{n-i} = 0$ for $i \neq 0$. Thus

$$\mathrm{sum}(A) = a_0 + \sum_{i=1}^{(n-1)/2} (a_i + a_{n-i}) = 1$$

and similarly for B and C. Taking the relationship $AA^T + BB^T + CC^T + DD^T = 4nI$ and multiplying by the row vector of ones (on the left) and the column vector of ones (on the right) we obtain

$$\mathrm{sum}(A)^2 + \mathrm{sum}(B)^2 + \mathrm{sum}(C)^2 + \mathrm{sum}(D)^2 = 4n$$

and therefore $\mathrm{sum}(D)^2 = 4n - 3 = (2r + 1)^2$ and $\mathrm{sum}(D) = s(2r + 1)$ where $s = \pm 1$.

Since D is symmetric and $2d_i \equiv 2 \pmod 4$

$$\mathrm{sum}(D) = 1 + 2 \sum_{i=1}^{(n-1)/2} d_i \equiv n \pmod 4.$$

Therefore $r^2 + r + 1 \equiv s(2r + 1) \pmod 4$. Since

$$r^2 + r + 1 \equiv (-1)^{\lfloor (r+1)/2 \rfloor} \pmod 4 \qquad \text{and} \qquad 2r + 1 \equiv (-1)^r \pmod 4$$

we have $s = 1$ when $r \equiv 0, 1 \pmod 4$ and $s = -1$ otherwise. □

As described in Section 5.4 we have that the entries of these matrices satisfy the relationship

$$a_k b_k c_k d_k a_{2k} b_{2k} c_{2k} = -1$$

for $k \neq 0$ (with indices reduced mod n if necessary).

Proof. We can equivalently consider circulant best matrices to be polynomials given by the generating function of the entries of their first rows. In this formulation A, B, C, D are polynomials with ± 1 coefficients and of degree $n - 1$ that satisfy

$$A(x)A(x^{-1}) + B(x)B(x^{-1}) + C(x)C(x^{-1}) + D(x)D(x^{-1}) = 4n \qquad (1)$$

in the ideal generated by $x^n - 1$ (all computations will take place in this ideal).

Let A_+ denote the polynomial containing the terms of A with positive coefficients and let $|A_+|$ denote the number of terms in A_+. Then $A = 2A_+ - T$ where $T(x) \coloneqq \sum_{i=0}^{n-1} x^i$. Since $x^i T = T$ we have $A_+ T = |A_+|T$ and $T^2 = nT$.

Since A is anti-symmetric (i.e., $A(x) + A(x^{-1}) = 2$) we have $A(1) = 1$ and $|A_+| = (T(1) + A(1))/2 = (n+1)/2$. Furthermore,

$$\begin{aligned} A(x)A(x^{-1}) &= 2A - A^2 \\ &= 2(2A_+ - T) - (2A_+ - T)^2 \\ &= 4A_+ - 4A_+^2 - (2T - 4|A_+|T + nT) \\ &= 4A_+ - 4A_+^2 + nT \end{aligned} \tag{2}$$

and similarly for B and C.

Since D is symmetric (i.e., $D(x) = D(x^{-1})$) we have

$$D(x)D(x^{-1}) = (2D_+ - T)^2 = 4D_+^2 + (n - 4|D_+|)T. \tag{3}$$

By the symmetry of D we have $D = 1 + 2\sum_{i=1}^{(n-1)/2} d_i x^i$ and thus $|D_+|$ is odd.

Equating (1)–(3) and dividing by four we have

$$A_+ - A_+^2 + B_+ - B_+^2 + C_+ - C_+^2 + D_+^2 + (n - |D_+|)T = n. \tag{4}$$

Since $A_+ = \sum_{a_i=1} x^i$ we have $A_+^2 \equiv \sum_{a_i=1} x^{2i} \pmod 2$ and (4) reduces to

$$\sum_{a_i=1} (x^{2i} + x^i) + \sum_{b_i=1} (x^{2i} + x^i) + \sum_{c_i=1} (x^{2i} + x^i) + \sum_{d_i=1} x^{2i} \equiv 1 \pmod 2$$

since both n and $|D_+|$ are odd.

Since n is odd the congruence $i \equiv 2y \pmod n$ has exactly one solution $0 \le y < n$ for each $0 \le i < n$. Denoting this solution by $i/2$ we have

$$\sum_{a_{i/2}=1} x^i + \sum_{a_i=1} x^i + \sum_{b_{i/2}=1} x^i + \sum_{b_i=1} x^i + \sum_{c_{i/2}=1} x^i + \sum_{c_i=1} x^i + \sum_{d_i=1} x^{2i} \equiv 1 \pmod 2.$$

In other words, we have that the number of entries in $\{a_{i/2}, a_i, b_{i/2}, b_i, c_{i/2}, c_i, d_{i/2}\}$ that are positive is 1 (mod 2) for $i = 0$ and 0 (mod 2) for $i \neq 0$. Letting $k = i/2$ for $i \neq 0$ this means $a_k a_{2k} b_k b_{2k} c_k c_{2k} d_k = -1$ as required. □

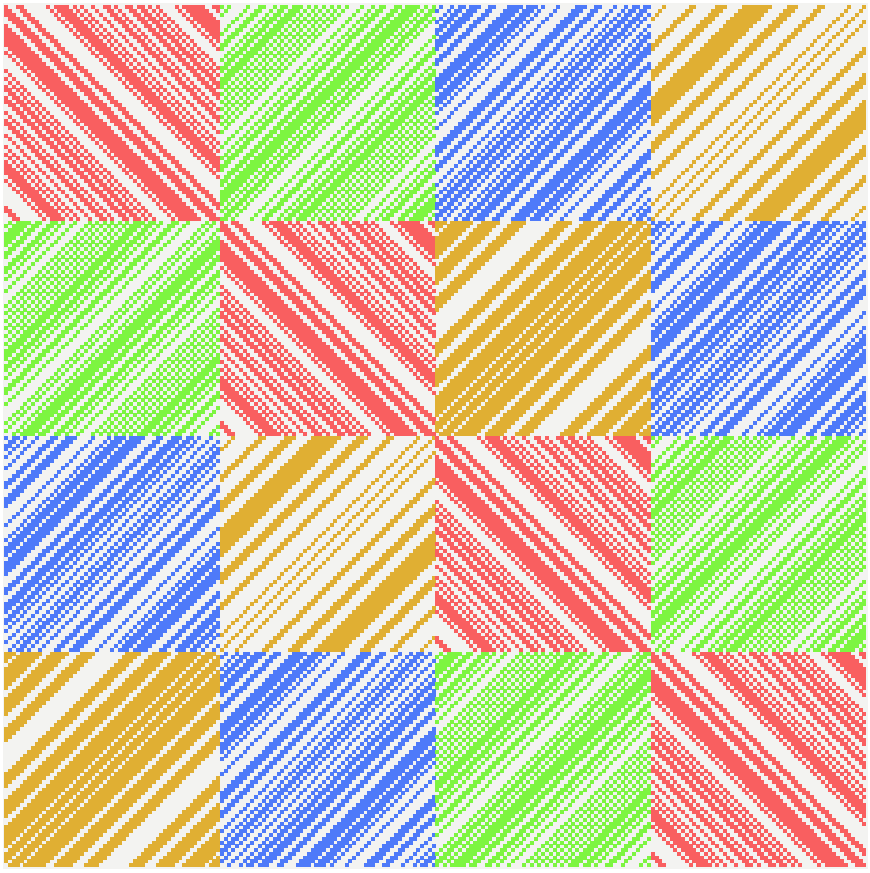

Fig. 1. A new skew Hadamard matrix of order $4 \cdot 57$ constructed using best matrices of order 57 with the following first rows:

```
+--++------+-++++-+-+--+--++-+--++-++-+-+----+-++++++--++
+-+-++++-++--++-+--+---+-+-+-+-+-+-+++-++-+--++--+----+-+
++-++-+--+---++--+++----+---+-+++-++++---++--+++-++-+--+-
+---+------++-----+++----++++++++----+++-----++------+---
```

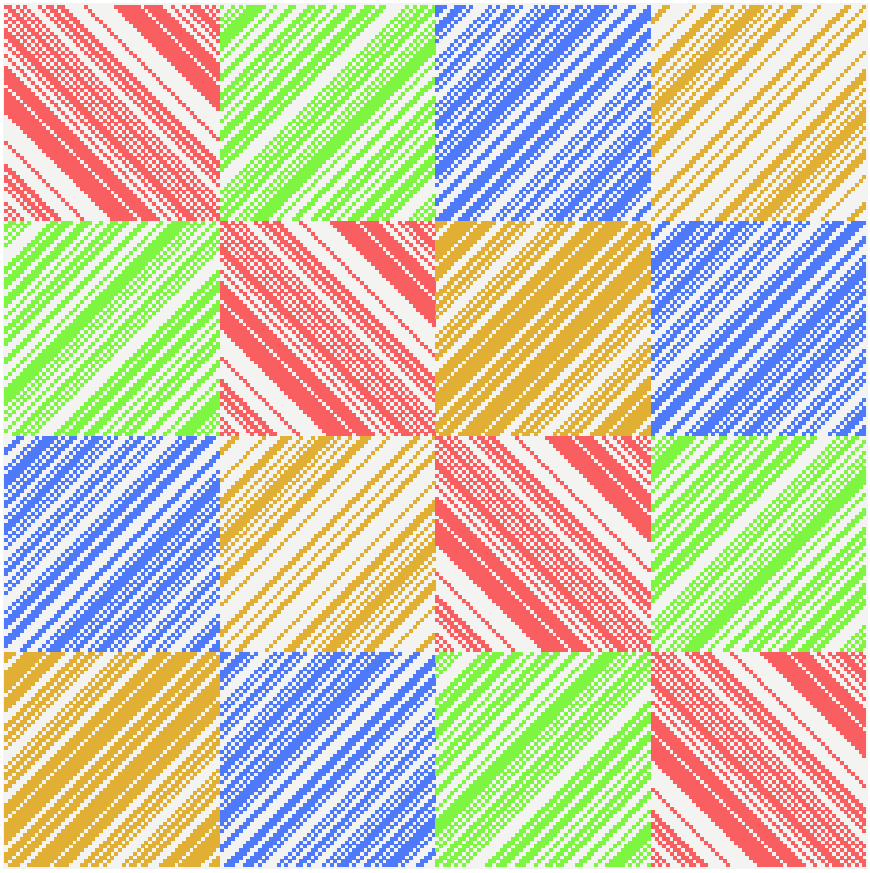

Fig. 2. A new skew Hadamard matrix of order $4 \cdot 57$ constructed using best matrices of order 57 with the following first rows:

```
++-+-+---+-++-++-----+-------+++++++-+++++--+--+-+++-+-+-
+-++-+-+-----++-+++--+++--+--++-++---++---+--+++++-+-+--+
+---++---+-++++-+-+++-+-+++-+-+---+-++--+-+----+-+++--+++
+---++------+----+++--+--+-++++-+--+--+++----+------++---
```

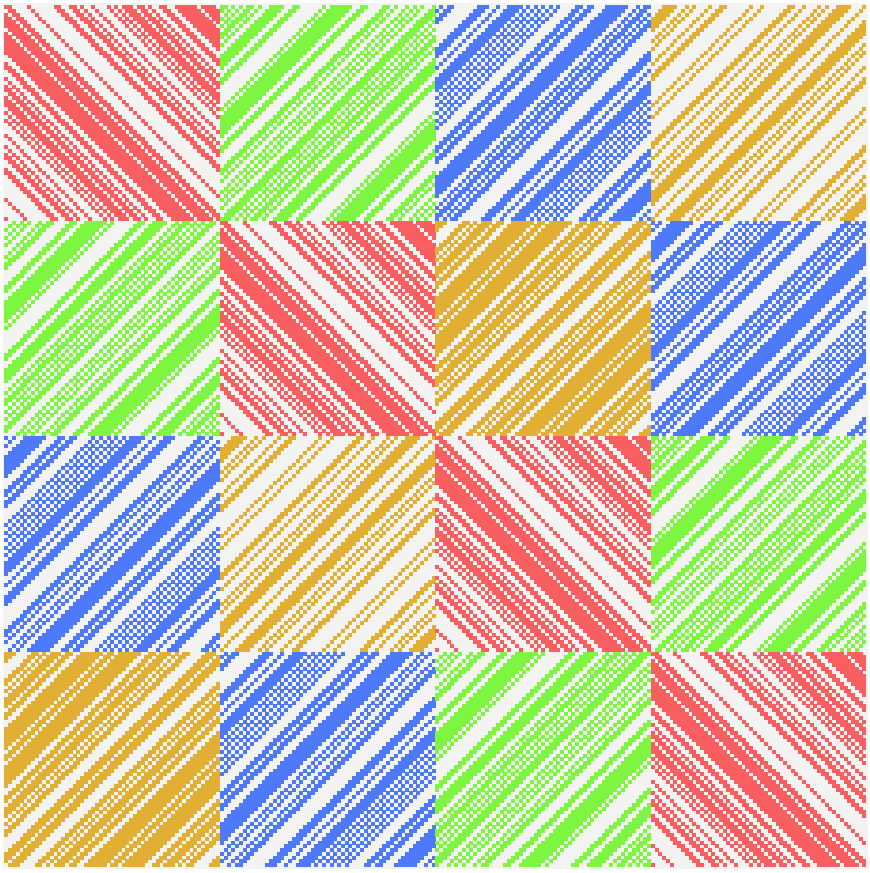

Fig. 3. A new skew Hadamard matrix of order $4 \cdot 57$ constructed using best matrices of order 57 with the following first rows:

```
++----+------++--++-+--+---+-+-+++-++-+--++--++++++-++++-
+--+-+-+--+-++----++--++++++-+------++--++++--+-++-+-+-++
++-+--++++++---++-++----+-+-+-+-+-++++--+--+++------++-+-
+---++++--+-+---+--++------+--+------++--+---+-+--++++---
```