A SAT Certification for the Nonexistence of Weight 16 Codewords in a Projective Plane of Order Ten

Content Areas: Constraint Satisfaction and Optimization (Search, Satisfiability, Solvers and Tools)

Abstract

In the 1970s and 1980s, searches performed by L. Carter, C. Lam, L. Thiel, and S. Swiercz showed that projective planes of order ten with weight 16 codewords do not exist. These searches required highly specialized and optimized computer programs and required about 2,000 hours of computing time on mainframe and supermini computers. In 2010, these searches were verified by D. Roy using an optimized C program and 16,000 hours on a desktop machine. In this paper, we reduce the search problem to the Boolean satisfiability problem (SAT) and construct a collection of SAT instances that we use to verify their searches in 130 hours on a desktop machine. This was accomplished using the state-of-the-art cube-and-conquer SAT solving paradigm in combination with a computer algebra system. Our searches uncovered four partial projective planes missed by previous searches and produced a nonexistence proof of almost 300 gigabytes in size.

1 Introduction

Geometry is one of the oldest branches of mathematics, being first axiomatically studied by Euclid in the 3rd century BC. Given a line and a point not on it, Euclid's "parallel postulate" implies that there exists exactly one line through the point and parallel to the given line. For 2000 years mathematicians tried to prove this axiom but eventually geometries that did not satisfy the parallel postulate were discovered. For example, in the early seventeenth century G. Desargues studied *projective geometry* where parallel lines do not exist. Projective geometry became widely studied in the nineteenth century, leading to the discovery of projective geometries containing a finite number of points.

Despite a huge amount of study for over 200 years, some basic questions about finite projective geometries remain open—for example, how many points can a finite projective plane contain? It is well-known (Kåhrström 2002) that this number must be of the form $n^2 + n + 1$ for some natural number n (known as the *order* of the plane) and certain orders such as $n = 6$ have been ruled out by theoretical arguments. For every other n up to ten a finite projective plane of order n can be shown to exist through an explicit construction. No theoretical explanation is known that answers the question if ten can be the order of a projective plane, but in the 1970s and 1980s an enormous amount of computing was used to show that no such plane exists (Lam 1991).

The computations were based on the existence of codewords in the error-correcting code generated by a projective plane of order ten. It was shown (Hall 1980) that such a code must contain codewords of weights 15, 16, or 19. Exhaustive searches using the most powerful computers of the day showed that such codewords do not exist (MacWilliams, Sloane, and Thompson 1973; Carter 1974; Lam, Thiel, and Swiercz 1986; 1989). Thus, a projective plane of order ten does not exist.

Each search required more advanced search techniques and orders of magnitude more computational power than the previous search—the weight 15 search being the easiest and the weight 19 search being the most challenging. In this paper we focus on the weight 16 search that originally required about 2,000 hours on supercomputers and a VAX-11 supermini machine. Additionally, in 2010, using an optimized C implementation the weight 16 search was verified in 16,000 hours on a desktop machine (Roy 2010).

We provide for the first time a reduction of the weight 16 codeword existence problem to the Boolean satisfiability problem (SAT) and a SAT certification that a projective plane of order ten generates no weight 16 codewords. This is done using the cube-and-conquer SAT solving paradigm (Heule et al. 2011) in addition to using functionality from a computer algebra system (i.e., the SAT+CAS paradigm). See Section 2 for background on the cube-and-conquer and SAT+CAS paradigms, and Section 3 for a description of our SAT encoding. Our search completed in about 130 hours on a desktop machine, significantly faster than any previous search.

Furthermore, no previous search was able to provide a certificate on successful completion. Thus, an independent party had to take on faith that the searches did in fact complete. The lack of a certificate can have serious consequences: our search found four partial projective planes that had been claimed to not exist following the 1986 search. Thus, the original search was incomplete but because of the lack of a verifiable certificate this went unnoticed for over thirty years. In contrast, our search produces an unsatisfiability certificate that an independent party can use to verify that our searches were successfully run to completion. The proof of nonexistence generated by the SAT solver amounts

to almost 300 gigabytes in the binary DRAT format. See Section 4 for details on our implementation and results.

2 Background

We now describe the background necessary to understand the nonexistence results of this paper. First, we describe the cube-and-conquer paradigm that we used to solve the SAT instances. Second, we describe the SAT+CAS paradigm that was necessary to improve the performance of the SAT solver. Lastly, we give the mathematical background on projective planes and their symmetry groups that is necessary to understand our SAT reduction and the symmetry breaking techniques that we used to make the search efficient.

2.1 The cube-and-conquer paradigm

The cube-and-conquer paradigm was first developed by Heule, Kullmann, Wieringa, and Biere (Heule et al. 2011) for computing van der Waerden numbers, a notoriously difficult computational problem from combinatorics. They reported that the cube-and-conquer method performed up to twenty times faster than any other known method. Moreover, in recent years the cube-and-conquer method has been used to resolve long-standing combinatorial problems such as the Boolean Pythagorean triples problem (Heule, Kullmann, and Marek 2017) and computing the fifth Schur number (Heule 2018).

The idea behind the cube-and-conquer method is to split a SAT instance into subproblems defined by *cubes* (propositional formulae of the form $l_1 \wedge \cdots \wedge l_n$ where l_i are literals). Each cube defines a single subproblem—generated by assuming the cube is true—and each subproblem is then solved or "conquered" either in parallel or in sequence. One of the major challenges of this method is to efficiently find a collection of cubes that makes each of the subproblems easy enough to solve quickly.

The primary reason that cube-and-conquer has been so successful is because of the insight that cubing is effectively done by "look-ahead" solvers while conquering is effectively done by "conflict-driven" solvers. Look-ahead solvers make progress by reasoning on a *global* level, i.e., at each stage they attempt to find the next decision that makes as much progress as possible. Conversely, conflict-driven solvers reason on a more *local* level, attempting to find decisions that happen to work well together even if each individual decision is not optimal globally (Heule and van Maaren 2009).

Thus, look-ahead solvers are good at partitioning a problem into subproblems of approximately equal difficulty and conflict-driven solvers are good at uncovering clever ways to solve problems that admit relatively short solutions. Using look-ahead solvers for cubing and conflict-driven solvers for conquering produces a solving method that often outperforms either look-ahead or conflict-driven solvers.

2.2 The SAT+CAS paradigm

The SAT+CAS paradigm is a recently proposed paradigm for solving problems that can be specified by both logical and algebraic constraints. First proposed by (Ábrahám 2015) and (Zulkoski, Ganesh, and Czarnecki 2015), it has since been successfully used to solve an impressive variety of problems (Davenport et al. 2019).

The idea behind the SAT+CAS paradigm is to use a SAT solver on the constraints that can be directly encoded in Boolean logic and to use a CAS on the constraints that are too complex to be directly encoded in SAT. Because SAT solvers are currently the best solution for many combinatorial problems (Heule, Kullmann, and Biere 2018) the SAT+CAS paradigm can be useful for many problems that seemingly have nothing to do with logic.

In short, the paradigm is effective because it combines the search power of modern SAT solvers with the expressiveness and mathematical functionality of modern computer algebra systems. The SAT+CAS paradigm has been already been successful in searching for combinatorial matrices (Bright, Kotsireas, and Ganesh 2018; Bright et al. 2019) making it natural to use the paradigm to search for projective planes as well.

2.3 Projective planes

A projective plane is a collection of points and lines that satisfy certain axioms, for example, in a projective plane any two lines intersect at a unique point. Finite projective planes can be defined in terms of *incidence matrices* that have a 1 in the (i, j)th entry exactly when the jth point is on the ith line. In this framework, a projective plane of order n is a square $\{0, 1\}$-matrix of order $n^2 + n + 1$ such that

(P1) all rows and columns contain exactly $n + 1$ ones, and

(P2) the inner product of any two rows or columns is one.

Two projective planes are said to be *equivalent* if one can be transformed into the other via a series of row or column permutations.

Projective planes are known to exist in all orders that are primes or prime powers but despite extensive study for over 200 years it is unknown if they exist in any other orders. Some orders such as $n = 6$ have been ruled out on theoretical grounds making $n = 10$ the first uncertain case. This stimulated a massive computer search for such a plane (Lam 1991) based on the form such a plane must have assuming certain codewords exist. A *codeword* is a $\{0, 1\}$-vector in the rowspace (mod 2) of a $\{0, 1\}$-matrix and the *weight* of a codeword is the number of 1s that it contains.

It can be shown (Hall 1980) that a projective plane of order ten must generate codewords of weight 15, 16, or 19, thus dramatically shrinking the search space and naturally splitting the search into three cases. As shown by (Carter 1974), there are ten possibilities (up to equivalence) for the first eight rows of the planes that generate weight 16 codewords. These cases are listed in Table 1 along with the symmetries that exist in the first eight rows and initial columns (see below for more details). Five of these possibilities (cases II to VI.a) were eliminated by the searches of (Carter 1974) and the other five were eliminated by the searches of (Lam, Thiel, and Swiercz 1986).

In each search a computer would attempt to extend the first eight rows into a complete projective plane, often working column-by-column. If the search reached a column that

Case	Initial Columns	Symmetries	Group Size
I.a	28	$S_4 \wr S_2$	1152
I.b	23	$S_4 \times S_4$	576
I.c	18	$S_4 \wr S_2$	1152
II	28	$S_4 \times S_2$	48
III	28	D_8	16
IV	28	$D_4 \times S_2$	16
V	28	$S_3 \times S_2$	12
VI.a	28	$S_2 \times S_2$	4
VI.b	26	S_2	2
VI.c	24	$S_2 \times S_2$	4

Table 1: The ten possible cases for the first eight rows of a projective plane of order ten generating a weight 16 codeword and the symmetry groups of the initial columns. Here S_n denotes the symmetric group of order $n!$, D_n denotes the dihedral group of order $2n$, and $\wr$ denotes the wreath product.

could only be completed by violating the definition of a projective plane then the search would *backtrack* to a previous column and try another possibility until all possibilities had been tried.

2.4 Structure of the weight 16 starting cases

(Carter 1974) showed that in each weight 16 starting case the first eight rows and 16 columns can be completely determined up to equivalence. Furthermore, he derived properties that the structure of the rest of the projective plane must satisfy. In particular, the projective plane has the following decomposition into a 3×2 grid of submatrices as follows:

$$\begin{array}{c} \\ 8 \\ 72 \\ 31 \end{array} \begin{array}{c} \begin{array}{cc} 16 & 95 \end{array} \\ \begin{pmatrix} 2 & k \\ 9 & 8-2k \\ 0 & k+3 \end{pmatrix} \end{array}$$

Here the numbers outside the matrix denote the number of rows or columns in that part of the submatrix. The numbers inside the matrix denotes how many 1s there are in each column in that part of the submatrix; certain columns depend on a parameter k that differs between columns. The first 16 columns along with the columns with $k > 1$ are known as the *initial* columns. The columns with $k > 0$ are known as *inside* columns and columns with $k = 0$ are known as *outside* columns. The number of outside columns in the projective plane varies between cases but each case has at least 45 outside columns so we may assume that the upper-right 8×45 submatrix contains zeros.

2.5 Symmetry groups

A projective plane (or partial projective plane) may be symmetric in nontrivial ways, in other words, there may exist row or column permutations that fix the entries of the plane. Such symmetries are important to detect because they can dramatically reduce the search space—and therefore the running time—of any search that makes use of them.

For example, Figure 1 shows the starting partial projective plane (the first eight rows and initial columns) from case I.c. This matrix is symmetric under the permutation that swaps

```
11110000000000010
00001111000000010
00000001111000010
00000000000111110
10001000100100001
01000100010010001
00100010001001001
00010001000100101
```

Figure 1: The upper-left 8×18 submatrix from case I.c.

the first two rows and column k with column $k+4$ for $1 \leq k \leq 4$. The set of all row and column permutations that fix the entries of a matrix forms a group known as the *symmetry group* of the matrix.

In the matrix of Figure 1 any permutation of the first four rows, any permutation of the last four rows, and the permutation that swaps row i and row $i+4$ for $1 \leq i \leq 4$ occur (with appropriate column permutations) in the symmetry group. The size of this permutation group is $4! \cdot 4! \cdot 2 = 1152$ and the group is isomorphic to the group of symmetries of a pair of tetrahedrons. The symmetry groups for each of the ten possible weight 16 starting configurations (the first eight rows and the initial columns) are given in Table 1.

3 SAT encoding

We now describe the SAT encoding that we use to prove the nonexistence of projective planes of order ten containing weight 16 codewords. We use the Boolean variables $p_{i,k}$ to correspond with the (i,k)th entry of the projective plane we are attempting to find, with $p_{i,k}$ assigned to true exactly when the (i,k)th entry is a 1.

3.1 Incidence constraints

First, we describe how we encode the property that the incidence matrix defined by the $p_{i,j}$ satisfies the properties of a projective plane. In particular, we encode property (P2) that any two rows or columns of the projective plane intersect exactly once (we say two rows or two columns *intersect* if they share a 1 in the same location). The constraints are split into two types:

(1) All row and column inner products are *at most* 1.

(2) All row and column inner products are *at least* 1.

Additionally, in the second case, it was only necessary to consider the inner products between the first eight rows and the later rows, and the inner products between the first 16 columns and the later columns. We also only used the first 80 rows and at most 71 columns. Our searches showed that there are no satisfying assignments of even these weaker constraints.

For (1), the constraints

$$\bigwedge_{1 \leq k,l \leq 111} (\neg p_{i,k} \vee \neg p_{i,l} \vee \neg p_{j,k} \vee \neg p_{j,l})$$

say that row i and row j do not intersect twice. In other words, the inner product of row i and j is at most 1. We include these constraints for each distinct pair of indices (i,j)

with $1 \leq i, j \leq 80$ and ignore the constraints where k or l is larger than the last column used (which varied between cases).

For (2), the constraint $\bigvee_{k=1}^{111}(p_{i,k} \wedge p_{j,k})$ says that row i and row j intersect at least once (i.e., have an inner product of at least 1). However, this constraint is not in conjunctive normal form so it can't directly be used with a typical SAT solver.

Instead, we use this constraint in the form $\bigvee_{k \in S(i)} p_{j,k}$ where $S(i)$ is the set of indices k such that $p_{i,k}$ is true. The first eight rows of the projective plane are completely known beforehand in each starting case, so $S(i)$ is well-defined for $1 \leq i \leq 8$. We include these constraints for all $1 \leq i \leq 8$ and $9 \leq j \leq 80$.

Similarly, we include constraints that say that column k and column l intersect at least once. These constraints are of the form $\bigvee_{i \in T(k)} p_{i,l}$ where $T(k)$ is the set of indices i such that $p_{i,k}$ is true. We used these constraints for all k between 1 and 16 and for all $l > 16$ up to the last column used.

We have chosen the number of rows and columns to use based on the known structure of the starting cases and in an attempt to minimize the number of variables and constraints necessary. In order to use the above constraints we require at least 80 rows and all inside columns because the indices in the set $S(i)$ for $1 \leq i \leq 8$ are from the inside columns, and the indices in the set $T(k)$ for $1 \leq k \leq 16$ are from the first 80 rows. By experimentation we found that *only* using inside columns produced satisfiable instances—we found that it was necessary to use an additional five outside columns to make all instances unsatisfiable.

3.2 Breaking column symmetries

Consider the starting matrix shown in Figure 2 (the first eight rows of case VI.c with the inside columns). This matrix has a symmetry group containing $5!^6 \cdot 4!^2 \cdot 2^2$ permutations. The factor of 2^2 arises from symmetries that involve row permutations and we discuss how we handle those in Section 3.3. The larger $5!^6 \cdot 4!^2$ factor arises from column permutations of the last 38 columns. We break these symmetries by enforcing a lexicographic ordering on the columns.

In each starting case the first 16 columns of the projective plane are explicitly known and each column has exactly nine 1s following the first eight rows. Thus, by sorting the rows following the first eight rows we can ensure there are nine consecutive 1s in the first column (see Figure 3).

We describe how we enforce a lexicographic ordering on the columns through the example submatrix in Figure 3. For simplicity we assume the first column does not intersect with any of the columns being ordered in the first eight rows. A similar construction can still be used if this is not true (replacing the first column with another column). Each row of the submatrix contains at most a single 1 or we would have a pair of columns that intersect each other twice. Thus, lexicographically ordering the columns of the submatrix ensures that a 1 cannot be in the upper-right or lower-left corners of the submatrix as displayed by the 0s in Figure 3.

Similarly, each column of the submatrix contains at most a single 1. By assumption, the first column must intersect each of the other columns in question somewhere in the given submatrix, so each column contains exactly a single 1. Consider the entry marked x in Figure 3. If this entry contains a 1 then all entries in the next column that are above it must be 0 for the columns to be correctly ordered.

Thus, considering the variables labeled x, y, and z in Figure 3, we include the clauses $\neg x \vee \neg y$ and $\neg x \vee \neg z$ in our encoding. We include similar clauses for each unknown entry in the submatrix. The remaining satisfying assignments are exactly those whose columns are in lexicographic order, therefore enforcing a unique ordering on columns which are otherwise identical. In the case of the columns shown in Figure 2 this decreases the size of the search space by a factor of $5!^6 \cdot 4!^2 \approx 10^{15}$.

3.3 Breaking initial symmetries

In Section 3.2 we described how we break most of the symmetries in our instances. However, it remains to break the initial symmetries involving *both* row and column permutations (i.e., those described in Table 1). We used two methods of breaking these symmetries. One method was particularly effective when the size of the symmetry group was not too large and was used in the cases II to VI.c. The second method took advantage of the specific form of the symmetry group of the cases I.a–c.

Method 1 In addition to the column symmetries of the last 38 columns of the matrix in Figure 2 there are an additional two generators φ_1 and φ_2 of the symmetry group of this matrix. These generators involve both row and column permutations. In cycle notation the row permutations are $(15)(26)(34)$ and $(12)(34)(56)(78)$ and the column permutations are completely determined by these row permutations. For example, after swapping rows 1 and 5 and rows 2 and 6, the first column (containing 1s in rows 1 and 2) becomes the 14th column (containing 1s in rows 5 and 6), so φ_1 must send column 1 to column 14.

In a similar way, permutations of the first 16 columns uniquely extend to permutations of the first 80 rows. Consider the symmetries of the first 80 rows and the initial columns where the undetermined entries are given some fixed value such as zero. In the example given in Figure 2, the initial columns are the first 24 columns. Our first symmetry breaking method focuses on the unknown entries in the upper-left 80×24 submatrix P.

In particular, we fix an ordering on the variables in the submatrix P whose values are undetermined. For example, one possible ordering of these variables is left-to-right and top-to-bottom, i.e.,

$$L_P \coloneqq [p_{9,17}, p_{9,18}, p_{9,19}, \ldots, p_{80,22}, p_{80,23}, p_{80,24}].$$

Under the symmetry φ_1 this ordering becomes

$$[p_{56,19}, p_{56,22}, p_{56,17}, \ldots, p_{17,18}, p_{17,20}, p_{17,24}]$$

which we denote by $\varphi_1(L_P)$. If P is a 80×24 partial projective plane then $\varphi_1(P)$ is also a 80×24 partial projective plane. It follows that any partial projective plane of this form can be transformed into an equivalent partial projective

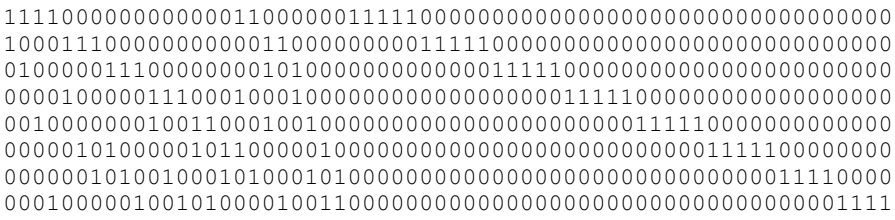

Figure 2: The upper-left 8×62 submatrix in case VI.c.

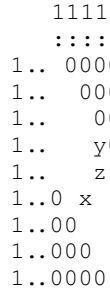

Figure 3: An internal 9×5 submatrix of a projective plane where the columns share a 1 in one of the first eight rows, the rows share a 1 in the first column, and no 1s appear in the hidden rows above the submatrix. The 0s that appear are known under the assumption that the columns are lexicographically ordered. This also implies that the entries marked y and z are 0 if the entry marked x contains a 1.

plane with $L_P \leq_{\text{lex}} \varphi_1(L_P)$. Similarly, up to equivalence we can assume that

$$L_P \leq_{\text{lex}} \varphi(L_P) \qquad (*)$$

where φ is any symmetry of P. In the example of Figure 2 we take φ to be φ_1, φ_2, and $\varphi_1 \circ \varphi_2$ (there is no need to take φ to be the identity symmetry since then $(*)$ is trivial).

It remains to describe how we encode the lexicographic constraint $(*)$ in conjunctive normal form. We express the general lexicographic constraint

$$[x_1, \ldots, x_n] \leq_{\text{lex}} [y_1, \ldots, y_n]$$

using $3n-2$ clauses and $n-1$ new variables (Knuth 2015). Denoting the new variables by $a_1, \ldots, a_{n-1}$, the clauses are

$$\neg x_k \vee y_k \vee \neg a_{k-1}, \;\; \neg x_k \vee a_k \vee \neg a_{k-1}, \;\; y_k \vee a_k \vee \neg a_{k-1}$$

for $k = 1, \ldots, n-1$ (with $\neg a_0$ omitted) along with the clause $\neg x_n \vee y_n \vee \neg a_{n-1}$.

In practice this method breaks almost all of the symmetries of the search space that exist in the initial columns, though it is not guaranteed to break them all. This is because for certain instantiations of the variables of P it may be the case that $(*)$ does not remove any equivalent solutions. This occurs when P is unchanged under all symmetries φ of the symmetry group (i.e., when $L_P = \varphi(L_P)$ for all symmetries φ). However, this did not occur very often in practice.

Because this method uses constraints of the form $(*)$ for every nontrivial symmetry φ, it worked best when the symmetry group was relatively small. We used this symmetry breaking method on the starting cases II–VI.c.

Method 2 Consider the starting matrix given in Figure 1 (case I.c). Since each row contains eleven 1s in total and there are five 1s per row in the initial columns, there are another six 1s in each row. Furthermore, since the rows are already pairwise intersecting in the initial columns, none of the rows intersect following the initial columns. In other words, each row contains six 1s on columns that are not incident with the other rows. We call the columns that contain those six 1s a *block* of 1s.

Without loss of generality we may assume that the first block occurs on columns 19–24, the second block occurs on columns 25–30, the third block on columns 31–36, etc. By exhaustive search, we find that there are 49,472 solutions of the SAT instance that uses the first 80 rows, the columns up to and including the first block, and the lexicographic column ordering clauses described in Section 3.2.

Our symmetry breaking method now uses the following properties of the symmetry group of case I.c:

1. The symmetry group fixing the first row (i.e., the symmetries that do not move the first block) is $S_4 \times S_3$.
2. The action of the symmetry group on the set of blocks is transitive, i.e., for each pair of blocks there exists a symmetry that sends one block to another block.

We use the first property to show that of the 49,472 possibilities for the first block, only 469 possibilities are inequivalent under the symmetries that fix the first row. These numbers agree with those reported by (Lam, Thiel, and Swiercz 1986). This splits the search into 469 distinct cases, one for each nonequivalent possibility of the first block. Additionally, we now use the second property to remove further symmetries within these cases.

We label each of the 49,472 possibilities for the first block with a label between 1 and 469 and let t be the labeling function that when given an instantiation of the first block returns its label. Furthermore, let B_i denote the submatrix consisting of the ith block where $1 \leq i \leq 8$ and let φ_i denote a symmetry that sends block i to block 1 where $2 \leq i \leq 8$. (By the second property such a symmetry exists.) We extend the labeling function t by giving an instantiation B of an arbitrary block B_i the same label as $\varphi_i(B)$.

In case I.c a partial projective plane that includes all initial columns and all blocks is made up of $18+8 \cdot 6 = 66$ columns. Up to equivalence we can assume in this case that any such partial projective plane must satisfy

$$t(B_1) \leq t(B_i) \quad \text{for} \quad 2 \leq i \leq 8. \qquad (**)$$

In other words, we can assume that the first block has the minimum label of all the blocks 1–8. For suppose a par-

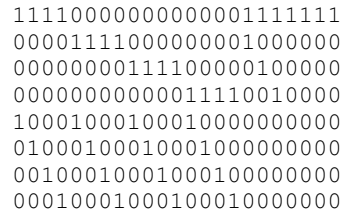

Figure 4: The upper-left 8×23 submatrix from case I.a.

tial projective plane did not satisfy this condition: then there must be an index m such that block m has the minimum label. Applying φ_m to this partial projective plane permutes the block labels and produces an equivalent partial projective plane such that block 1 has the minimum label.

To encode the constraint $(**)$ in our SAT instances we use a series of blocking clauses. In each SAT instance the left-hand side of $(**)$ is known in advance, since each instance contains a fixed instantiation of the first block. In the first SAT instance the label of the first block is 1. In this case $(**)$ is trivial and does not block any solutions.

In the second SAT instance we need to block all solutions where $t(B_i) = 1$ for $2 \leq i \leq 8$. To do this, suppose B is an instantiation of the first block that is labeled 1. We generate $\varphi_i^{-1}(\varphi(B))$ for all $2 \leq i \leq 8$ and all φ in the symmetry group fixing the first line. This gives us an explicit collection of instantiated blocks that we want to ignore. If B' is one of these blocks then the clause

$$\neg\Big(\bigwedge_{p \in B',\, p \text{ true}} p\Big) \equiv \bigvee_{p \in B',\, p \text{ true}} \neg p$$

prevents B' from occurring in the solution of the SAT instance. We include such clauses in the SAT instance for all B' of the form $\varphi_i^{-1}(\varphi(B))$. Similarly, in the kth SAT instance we include clauses of this form for all B' of the form $\varphi_i^{-1}(\varphi(B))$ where B is an instantiation of the first block whose label is strictly less than k.

The above description specifically applies to case I.c, but cases I.a and I.b can be handled in a similar way. In case I.a the main difference is that each block consists of seven columns and some columns are shared between blocks. However, the same method applies because the two properties of the symmetry group that we used in I.c also hold in this case (cases I.a and I.c have the same symmetry group).

Figure 4 shows the submatrix containing the first eight rows in case I.a, along with the first 16 columns and the columns of the first block. In this case we find 21,408 solutions of the SAT instance using the first 80 rows and 23 columns. We also use the lexicographic column ordering clauses (as described in Section 3.2) on the final four columns as these columns are otherwise identical. Using the symmetries that fix the first row we find that only 275 of the 21,408 solutions are inequivalent, thus naturally splitting the problem into 275 SAT instances that we solve in the same way as we solve the instances from case I.c.

We solve case I.b in a similar way, but in this case four of the blocks contain seven columns and the other four blocks contain six columns. We order the blocks such that the first four blocks contain seven columns and the last four blocks contain six columns. The first block is chosen to consist of the inside columns with 1s on the first row so that the upper-left 8×23 submatrix is identical to the matrix in Figure 4. Thus we also have 21,408 solutions of the SAT instance using the first 80 rows and 23 columns. The symmetry group that fixes the first row is the same as in case I.a so we also have 275 inequivalent solutions.

In case I.b the symmetry group does not act transitively on the blocks because there are no symmetries that send blocks with seven columns to blocks with six columns or vice versa. However, the symmetry group does act transitively on the first four blocks. Thus we use the same symmetry breaking condition given in $(**)$ except replacing the condition on i with $2 \leq i \leq 4$.

4 Implementation and results

All symmetry groups and row/column permutations in the symmetry groups were computed using the computer algebra system Maple 2018. Once those had been explicitly computed, a simple Python script (of approximately 200 lines) was written to generate the SAT instances. The script used 80 rows in all cases and accepted the number of columns to use as a parameter. Ten variations of this script were used, one for each starting case described in Table 1.

4.1 Cases II to VI.c

The SAT instances in these cases included the constraints (1) and (2) from Section 3.1, the column symmetry breaking constraints from Section 3.2, and the symmetry breaking constraints from method 1 in Section 3.3. The cube-and-conquer paradigm solved these SAT instances faster than any other method we tried. As described in (Heule, Kullmann, and Marek 2017) a two-level splitting process was used.

The top-level splitting was performed by exhaustively finding all possible satisfying assignments for a small portion of the projective plane—the first k columns where k began at 17 and was incremented by one until over 50 solutions were found. The exhaustive search was performed by a version of the SAT solver MapleSAT (Liang et al. 2016) that also generates proofs that no solutions are missed (i.e., a proof that $C \wedge \neg \bigvee_i S_i$ is unsatisfiable where C are the SAT constraints and S_i is a cube specifying the ith solution).

The "cubing" solver March_cu (Heule et al. 2011) was used to generate a second-level partition of each SAT instance using all initial columns, all inside columns, and five outside columns. For each SAT instance, March_cu generates a collection of cubes and produces a set of incremental SAT instances that contain both the constraints and the cubes. Performance in this step was improved by simplifying the SAT constraints before calling March_cu. We used the preprocessor of the SAT solver Lingeling (Biere 2017) because it generated simplification proof traces without renaming variables. The "conquering" SAT solver iGlucose, a modification of Glucose (Audemard and Simon 2018) by M. Heule, was used to solve the incremental SAT instances and produce proofs of unsatisfiability.

Case	Instances	Columns	Time (h)	Proof size
I.a	275	65	6.79	16G
I.b	275	68	15.51	32G
I.c	469	59	96.04	164G
II	75	65	3.25	16G
III	52	65	1.57	9.9G
IV	54	65	0.14	0.5G
V	196	65	0.66	3.2G
VI.a	81	65	1.26	8.5G
VI.b	160	66	2.40	15G
VI.c	64	67	3.87	24G

Table 2: Summary of the results of our implementation applied to all weight 16 cases.

The proofs from the simplification, cubing, and conquering steps were combined into a single proof by concatenation and the proofs were verified using the proof checker DRAT-trim (Wetzler, Heule, and Hunt 2014). The proof that the top-level partition was exhaustive was also verified by DRAT-trim.

4.2 Cases I.a to I.c

In cases I.a to I.c the SAT instances included the constraints (1) and (2) from Section 3.1 and the column symmetry breaking constraints from Section 3.2. Using MapleSAT we find all satisfying assignments of the columns up to and including the first block. A Python script checks which solutions are equivalent under the symmetries that fix the first row.

For every inequivalent solution a new SAT instance is generated including the symmetry breaking constraints from method 2 in Section 3.3. The cube-and-conquer method did not perform well on these instances, perhaps because the symmetry breaking clauses used in method 2 had length 36, significantly longer than the clauses used in method 1 which had length 3. We used MapleSAT to solve these instances and produce proofs of unsatisfiability which were then verified by DRAT-trim. The instances in cases I.a–b used all initial columns, all inside columns, and five outside columns. The instances in case I.c were the same except the columns in blocks 2 and 3 were ignored because (Lam, Thiel, and Swiercz 1986) found no solutions even in this restricted search area.

Surprisingly, 3 of the 469 SAT instances in case I.c were found to be satisfiable, producing four 80×59 partial projective planes (one instance had two inequivalent solutions and two instances had one solution each). Including the columns from blocks 2 and 3 in these instanced produced unsatisfiable instances, i.e., none of the solutions could be extended into a 80×71 partial projective plane.

4.3 Results

A summary of our results is presented in Table 2. In particular, this table specifies the number of SAT instances used in each case and how many columns were used in each SAT instance. The table also includes the total running time (in hours) of the SAT solvers and the size of the proofs (after trimming with DRAT-trim) in each case. The computations were run on a cluster of Intel Xeon E5-2683 processors running at 2.1 GHz. All cases were solved in approximately 130 hours with the hardest being case I.c. In this case the SAT instances tended to get progressively easier because the symmetry breaking method included an increasing number of blocking clauses.

The four 80×59 partial projective planes that were found in case I.c are included in the supplementary material. These are particularly interesting since (Lam, Thiel, and Swiercz 1986) claim that they should not exist. In this case we followed their initialization of the search exactly, using the same starting configuration matrix (shown in Figure 1), the same 80 rows, and the same 59 columns (only 41 of which contain undetermined entries). Thus we are forced to conclude their search was incomplete but because no certificate of their search exists it is impossible to determine the source of the discrepancy.

5 Conclusion and future work

We have provided for the first time a SAT certification that there exist no projective planes of order ten generating weight 16 codewords. This verifies the searches of (Carter 1974), (Lam, Thiel, and Swiercz 1986), and the verification of (Roy 2010). The previous searches relied on highly optimized computer programs and special-purpose search algorithms. In contrast, our search used widely available and well-tested SAT solvers, computer algebra systems, and proof verifiers. Our search produced the first proof of nonexistence for this problem that can be checked by a third party.

Furthermore, our search is the fastest known verification of this result. The searches of (Carter 1974) that used about 140 hours on supercomputers were verified in under 7 hours, and the searches of (Lam, Thiel, and Swiercz 1986) that used about 2000 hours on a VAX-11 were verified in under 125 hours. This is in large part due to the increase in computation power available today—however, the verification of (Roy 2010) which used modern AMD CPUs running at 2.4 GHz required 16,000 hours.

We do not claim our search is a *formal* verification of this result because our encoding relies on many properties that were derived mathematically and not in a computer-verifiable form. However, we now have a practical method for producing such a formal proof: by formally deriving the SAT encoding that we used from the projective plane axioms. This would require expertise in both projective geometry and a formal proof system and would certainly be a significant undertaking. However, the tools to do this already exist and have been used to formally prove other results derived using SAT certificates (Cruz-Filipe, Marques-Silva, and Schneider-Kamp 2018).

A similar SAT+CAS encoding can also be used to show the nonexistence of weight 15 codewords in under 10 seconds. The next challenge is to show the nonexistence of weight 19 codewords as well. We believe the SAT+CAS approach will be useful in this search but it will likely require alternative encodings and/or symmetry breaking methods that exploit the peculiarities of the weight 19 search.

References

Ábrahám, E. 2015. Building bridges between symbolic computation and satisfiability checking. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, 1–6. New York: ACM.

Audemard, G., and Simon, L. 2018. On the Glucose SAT solver. *International Journal on Artificial Intelligence Tools* 27(1):1–25.

Biere, A. 2017. CaDiCaL, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2017. In Balyo, T.; Heule, M.; and Järvisalo, M., eds., *Proc. of SAT Competition 2017 – Solver and Benchmark Descriptions*, volume B-2017-1 of *Department of Computer Science Series of Publications B*, 14–15. University of Helsinki.

Bright, C.; Đoković, D.; Kotsireas, I.; and Ganesh, V. 2019. A SAT+CAS approach to finding good matrices: New examples and counterexamples. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 1435–1442.

Bright, C.; Kotsireas, I.; and Ganesh, V. 2018. A SAT+CAS method for enumerating Williamson matrices of even order. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 6573–6580.

Carter, J. L. 1974. *On the existence of a projective plane of order ten*. Ph.D. Dissertation, University of California, Berkeley.

Cruz-Filipe, L.; Marques-Silva, J.; and Schneider-Kamp, P. 2018. Formally verifying the solution to the Boolean Pythagorean triples problem. *Journal of Automated Reasoning* 1–28.

Davenport, J. H.; England, M.; Griggio, A.; Sturm, T.; and Tinelli, C. 2019. Symbolic computation and satisfiability checking. *Journal of Symbolic Computation* in press.

Hall, M. 1980. Configurations in a plane of order ten. In *Annals of Discrete Mathematics*, volume 6. Elsevier. 157–174.

Heule, M. J. H., and van Maaren, H. 2009. Look-ahead based SAT solvers. *Handbook of satisfiability* 185:155–184.

Heule, M. J. H.; Kullmann, O.; Wieringa, S.; and Biere, A. 2011. Cube and conquer: Guiding CDCL SAT solvers by lookaheads. In *Haifa Verification Conference*, 50–65. Springer.

Heule, M. J. H.; Kullmann, O.; and Biere, A. 2018. Cube-and-conquer for satisfiability. In *Handbook of Parallel Constraint Reasoning*. Springer. 31–59.

Heule, M. J. H.; Kullmann, O.; and Marek, V. W. 2017. Solving very hard problems: Cube-and-conquer, a hybrid SAT solving method. In *IJCAI*, 4864–4868.

Heule, M. J. H. 2018. Schur number five. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 6598–6606.

Kåhrström, J. 2002. On projective planes. Technical report.

Knuth, D. E. 2015. *The Art of Computer Programming, Volume 4, Fascicle 6: Satisfiability*. Addison-Wesley Professional.

Lam, C. W. H.; Thiel, L.; and Swiercz, S. 1986. The nonexistence of code words of weight 16 in a projective plane of order 10. *Journal of Combinatorial Theory, Series A* 42(2):207–214.

Lam, C. W. H.; Thiel, L.; and Swiercz, S. 1989. The nonexistence of finite projective planes of order 10. *Canadian Journal of Mathematics* 41(6):1117–1123.

Lam, C. W. H. 1991. The search for a finite projective plane of order 10. *The American Mathematical Monthly* 98(4):305–318.

Liang, J. H.; Ganesh, V.; Poupart, P.; and Czarnecki, K. 2016. Exponential Recency Weighted Average Branching Heuristic for SAT Solvers. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 3434–3440.

MacWilliams, F. J.; Sloane, N. J. A.; and Thompson, J. G. 1973. On the existence of a projective plane of order 10. *Journal of Combinatorial Theory, Series A* 14(1):66–78.

Roy, D. J. 2010. Confirmation of the non-existence of a projective plane of order 10. Master's thesis, Carleton University.

Wetzler, N.; Heule, M. J.; and Hunt, W. A. 2014. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *International Conference on Theory and Applications of Satisfiability Testing*, 422–429. Springer.

Zulkoski, E.; Ganesh, V.; and Czarnecki, K. 2015. MathCheck: A math assistant via a combination of computer algebra systems and SAT solvers. In Felty, A. P., and Middeldorp, A., eds., *Automated Deduction - CADE-25*, volume 9195 of *Lecture Notes in Computer Science*. 607–622.

A SAT Certification for the Nonexistence of Weight 16 Codewords in a Projective Plane of Order Ten: Supplementary Material

Partial projective planes

Figures 1–4 contain the four inequivalent 80×59 partial projective planes generated from the satisfying assignments found in case I.c. The examples 3 and 4 are nearly identical and only differ in two columns of block 5.

It is straightforward to verify that these examples are valid partial projective planes: each column and row contains at most eleven 1s, and the inner product of any two rows (or columns) is at most 1. Furthermore, any row (or column) with exactly eleven 1s has an inner product with any other row (or column) of exactly 1. As an additional check, the structural properties of the partial plane with 80 rows derived by (Carter 1974) are satisfied: namely, there are exactly six 1s (plus one 1 in the first eight rows) in each column of each block and there are exactly eight 1s in each outside column.

Our search used the exact same initialization as the search described by (Lam, Thiel, and Swiercz 1986); we used the same upper-left 8×18 submatrix, the same 80 rows, and the same 59 columns. 41 of those columns contain undetermined entries, namely, the 36 columns from blocks 1 and blocks 4–8 (ignoring blocks 2 and 3) and the five outside columns that are incident with the row that is incident to columns 5 and 10.

The search of (Lam, Thiel, and Swiercz 1986) found no partial projective planes of this form:

> The program [...] did not find any completions for these 41 columns.

Since we found legal completions that should have been detected by their search we are forced to conclude that their search was incomplete. Despite this, they were correct in claiming that there are no projective planes that can be generated from the starting matrix of case I.c. Each of the examples we found could not be extended into a 80×71 partial projective plane that also includes the columns from blocks 2 and 3.

References

Carter, J. L. 1974. *On the existence of a projective plane of order ten*. Ph.D. Dissertation, University of California, Berkeley.

Lam, C. W. H.; Thiel, L.; and Swiercz, S. 1986. The nonexistence of code words of weight 16 in a projective plane of order 10. *Journal of Combinatorial Theory, Series A* 42(2):207–214.

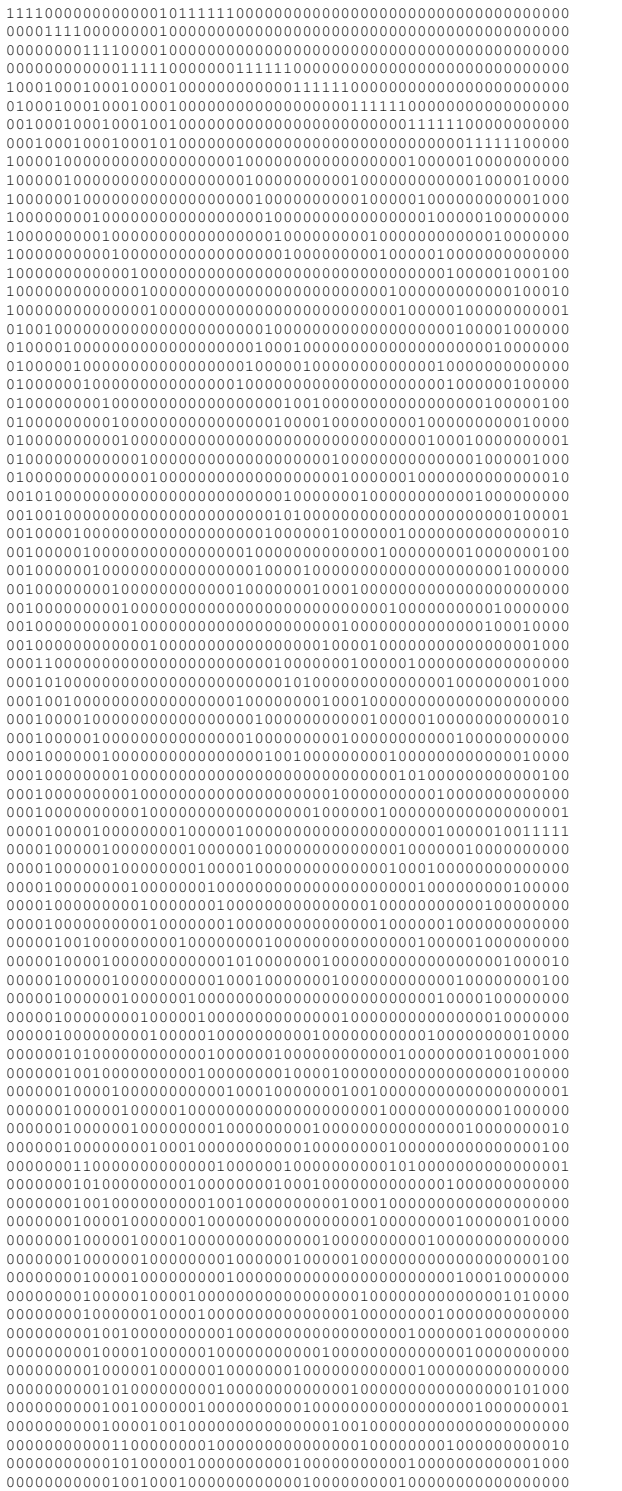

Figure 1: 80×59 partial projective plane example 1.

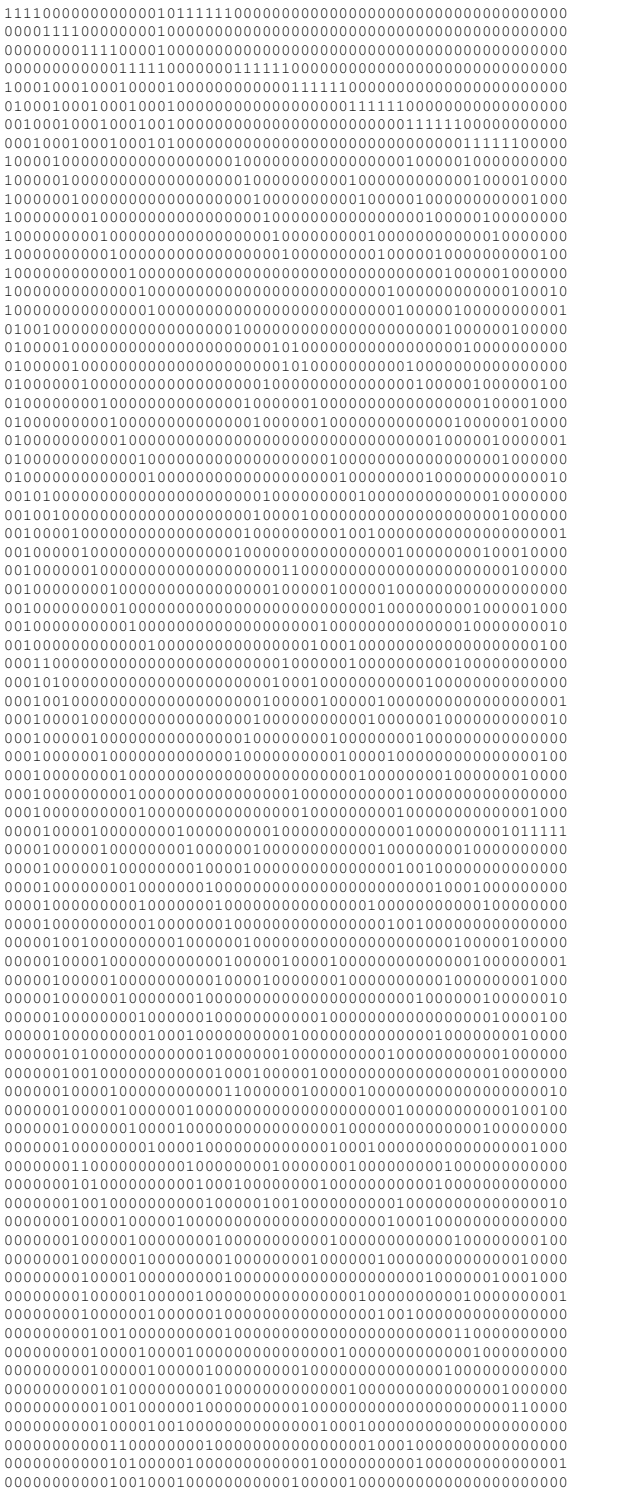

Figure 2: 80×59 partial projective plane example 2.

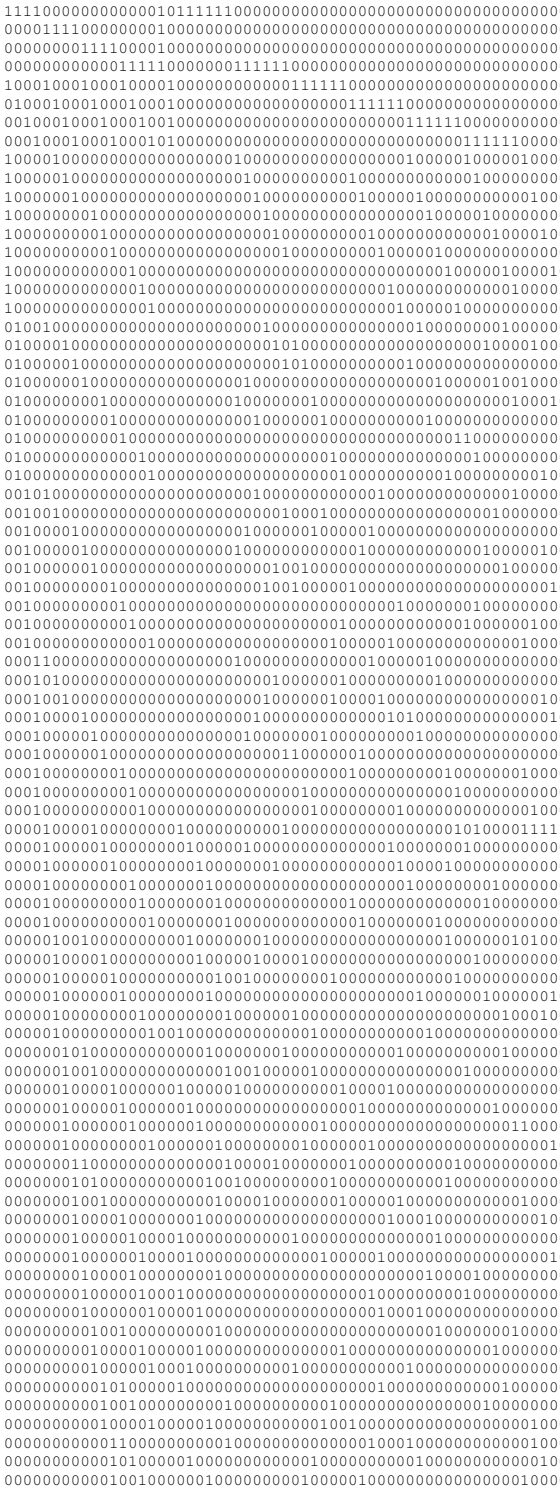

Figure 3: 80×59 partial projective plane example 3.

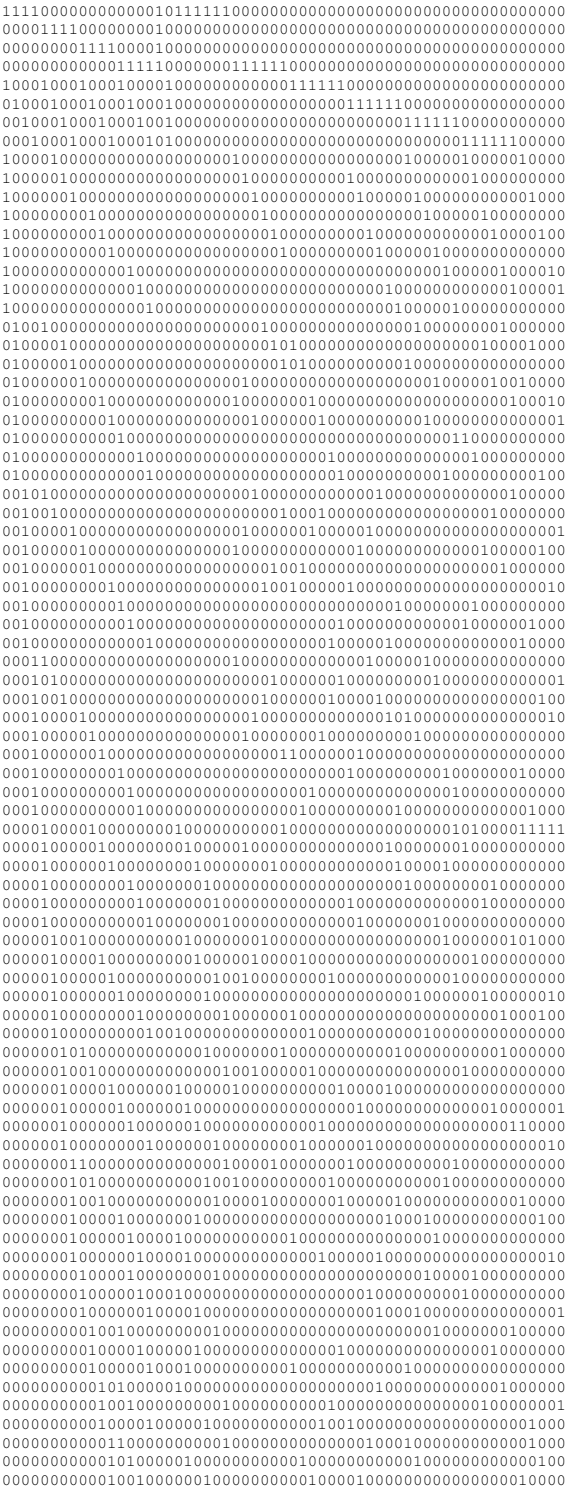

Figure 4: 80×59 partial projective plane example 4.