A SAT-based Resolution of Lam's Problem

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

Abstract

In 1989, computer searches by Lam, Thiel, and Swiercz experimentally resolved Lam's problem from projective geometry—the long-standing problem of determining if a projective plane of order ten exists. Both the original search and an independent verification in 2011 discovered no such projective plane. However, these searches were each performed using highly specialized custom-written code and did not produce nonexistence certificates. In this paper, we resolve Lam's problem by translating the problem into Boolean logic and use satisfiability (SAT) solvers to produce nonexistence certificates that can be verified by a third party. Our work uncovered consistency issues in both previous searches—highlighting the difficulty of relying on special-purpose search code for nonexistence results.

1 Introduction

Projective geometry was developed in the 1600s by renaissance artists and mathematicians in order to describe how to project a three dimensional scene onto a two dimensional canvas. Projective geometry has the counter-intuitive property that *any two lines must meet*. For example, a pair of train tracks (parallel lines in three dimensions) when projected onto two dimensions will meet on the horizon.

Despite an intensive amount of study for over 200 years some basic questions about projective geometry remain unsettled. For example—how many points can a projective geometry have? A geometry is said to be *finite* if it contains a finite number of points and a finite geometry is said to have *order* n if every line contains $n+1$ points (Dembowski 1968).

All finite projective geometries have been classified with the exception of those having exactly two dimensions—the *projective planes*. The first order for which it is theoretically uncertain if a projective plane exists is $n = 10$. Determining if such a projective plane exists has become known as *Lam's problem* after the work of (Lam, Thiel, and Swiercz 1989) experimentally showed that such a plane does not exist—work that was later independently verified by (Roy 2011).

As pointed out by (Heule, Kullmann, and Marek 2017) there are currently three kinds of solvers that are used to solve large but finite decision problems like Lam's problem: special purpose solvers, constraint satisfaction solvers, and satisfiability (SAT) solvers. They note that recently SAT solvers have become so strong that they are "the best solution in most cases". Even still, they note that some problems such as Lam's problem have only been solved by special-purpose means:

> An example where only a solution by [special purpose solvers] is known is the determination that there is no projective plane of order 10...

We remedy this situation by using a SAT solver to resolve the most challenging subcase of Lam's problem. Together with the recent SAT-based results of (Bright et al. 2020a,b) this provides a complete resolution of Lam's problem with all of the exhaustive search work completed by SAT solvers.

The previous searches done in Lam's problem remain fantastic achivements, but a SAT-based resolution has two primary advantages. First, it is more verifiable: a third party can check the nonexistence certificates for themselves and (once they believe in the encoding) be convinced in the nonexistence of a projective plane of order ten without having to trust a search procedure. Second, using well-tested SAT solvers to perform the search is less error-prone than writing special-purpose search code—a reality of developing software for computer-assisted proofs is that it is extremely difficult to make custom-written code both correct and efficient (Lam 1990).

Indeed, our results uncover discrepancies with both the original 1989 search and its 2011 independent verification. As we detail in section 2, the first two steps of Lam et al.'s search are to enumerate what are known as $A1$s and $A2$s. Our work agrees with the previous searches that up to isomorphism there are 66 possiblilities for the $A1$s. However, our count for the $A2$ possibilities disagrees with both of the counts of (Lam, Thiel, and Swiercz 1989) and (Roy 2011)—works which differ between themselves as well (see Section 5). We generate certificates that demonstrate our $A2$ search is complete (see Section 4) and verify the certificates with a proof verifier. These certificates were generated with the assistance of the symbolic computation library Traces (McKay and Piperno 2014)—but we describe how one can verify the certificates without needing to trust the output of the library.

Our work does not provide a completely *formal proof* of

the nonexistence of a projective plane of order ten because we rely on results that currently have no formal computer-verifiable proofs. In particular, we rely on a result of (Carter 1974) that the error-correcting code associated with a hypothetical projective plane of order ten must contain words that are referred to as weight 15, weight 16, or 'primitive' weight 19 words. The former two cases were first ruled out by the searches of (MacWilliams, Sloane, and Thompson 1973) and (Lam, Thiel, and Swiercz 1986) and were recently settled via SAT-based nonexistence certificates (Bright et al. 2020a,b). The primitive weight 19 search is by far the most challenging—it was ruled out by (Lam, Thiel, and Swiercz 1989) and it is the case that we consider in our work.

Our work does provide a possible avenue for constructing a formal proof: by deriving our SAT encoding and the mathematical results that we rely on inside a formal proof system. This would be a significant undertaking but is in principle possible with current tools. In fact, results that were recently proven using SAT solvers such as the resolution of the Boolean Pythagorean triples problem (Heule, Kullmann, and Marek 2016) and a case of the Erdős discrepancy conjecture (Konev and Lisitsa 2014) have since had formal proofs generated based on the SAT encoding (Cruz-Filipe, Marques-Silva, and Schneider-Kamp 2018; Keller 2019).

2 Background

We now provide the necessary background in order to understand our results. In particular, we outline the cube-and-conquer satisfiability solving paradigm, describe Lam's problem from projective geometry, describe the subcase for which we provide nonexistence certificates, and outline the symmetry breaking methods exploited by the nonexistence certificates.

2.1 Cube-and-conquer

The *cube-and-conquer* satisfiability solving paradigm was developed by (Heule et al. 2011) for solving hard combinatorial problems. The method uses two kinds of SAT solvers in two stages: First, a "cubing solver" splits a satisfiability instance into a large number distinct subproblems specified by *cubes*—formulas of the form $l_1 \wedge \cdots \wedge l_n$ where l_i are variables or negated variables. Second, for each cube a "conquering solver" solves the original instance under the assumption that the cube is true. The cube-and-conquer method tends to be effective at quickly solving large satisfiability instances when the cubing solver can generate many cubes encoding subproblems of approximately equal difficulty. It has since been applied to solve huge combinatorial problems such as the Boolean Pythagorean triples problem (Heule, Kullmann, and Marek 2017) and the computation of the fifth Schur number (Heule 2018).

2.2 Lam's problem

Projective geometry was formalized by mathematicians in the 1600s though its roots date back to the work of Pappus of Alexandria in the 4th century. In the 1800s projective geometry became extensively studied—including projective geometries that contain only a finite number of points (von Staudt 1856). The only finite projective geometries that remain to be classified are those in two dimensions and such objects are known as *projective planes*.

Projective planes consist of an incidence relationship between points and lines such that any two distinct lines intersect in a unique point and any two distinct points are on a unique line. To avoid trivial cases we also require that not all (or all but one) of the points lie on the same line. These axioms imply that all lines contain the same number of points and the plane contains the same number of points and lines. If each line has $n+1$ points then the projective plane said to be of *order* n and it will contain exactly n^2+n+1 points. If A is the $\{0,1\}$ incidence matrix encoding which points lie on which lines then the projective plane axioms imply that the off-diagonal entries of the matrices AA^T and A^TA are exactly 1. Two vectors are said to *intersect* if they share a 1 in the same position and therefore the projective plane axioms imply that any two rows or columns of A must intersect.

Projective planes are known to exist in all orders that are prime powers but despite extensive study no projective planes in any other orders are known and it has been conjectured that the order of a projective plane must be a prime power (Weisstein 2002). Certain orders including six have been theoretically ruled out (Bruck and Ryser 1949) leaving $n = 10$ as the first uncertain order—until the computational search of (Lam, Thiel, and Swiercz 1989) did not find a plane of order ten.

Lam et al.'s search was based on properties of the incidence matrix A of a hypothetical projective plane of order ten. In particular, the results of (Carter 1974) and (MacWilliams, Sloane, and Thompson 1973) imply that the words of Hamming weight 19 in the rowspace of A (mod 2) are of three possible forms (called oval, 16-type, and primitive) and there must exist some 16-type or primitive words. However, the searches of (Lam, Thiel, and Swiercz 1986) ruled out the existence of 16-type words and (Lam, Thiel, and Swiercz 1989) ruled out the existence of primitive words.

2.3 Primitive weight 19 words

The most challenging case in the resolution of Lam's problem is to show the nonexistence of primitive weight 19 words. The existence of such words greatly constrains the structure of the incidence matrix A of a projective plane of order ten—in particular, A can be decomposed into a 3×2 collection of submatrices as shown in Figure 1. The row sums of the submatrices were derived in (Carter 1974) and the column sums (that depend on a parameter k counting how many 1s appear in the first six rows of a column) were derived in (Lam, Crossfield, and Thiel 1985). We follow the labelling scheme that appears in the latter work.

Once $A1$ has been fixed this uniquely determines $A3$ and $A4$ without loss of generality (Lam, Crossfield, and Thiel 1985). There are typically a large number of possibilities for $A2$, though this number can be reduced by only considering nonisomorphic $A2$s under the symmetry group of $A1$ (see Section 2.4). Once all the possible $A2$s have been determined the search of Lam et al. continued by attempting to extend each $A2$ into the $A5$ submatrix. In no case was a

$$\begin{array}{r c} & \begin{array}{cc} 19 & 92 \end{array} \\ \begin{array}{r} 6 \\ 37 \\ 68 \end{array} & \left[\begin{array}{c|c} A1 & A4 \\ \hline A2 & A5 \\ \hline A3 & \end{array}\right] \end{array} \qquad \begin{array}{c} \begin{array}{cc} 19 & 92 \end{array} \\ \left[\begin{array}{cc} 5 & 6 \\ 3 & 8 \\ 1 & 10 \end{array}\right] \end{array} \qquad \begin{array}{c} \begin{array}{cc} 19 & 92 \end{array} \\ \left[\begin{array}{cc} k & k \\ 9-2k & 4-2k \\ k+2 & k+7 \end{array}\right] \end{array}$$

labels row sums column sums

Figure 1: Structure of the incidence matrix of a projective plane of order ten containing a primitive weight 19 word. The numbers outside the matrices count the number of rows or columns in each submatrix.

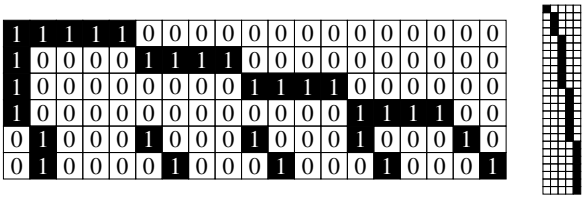


Figure 2: On the left the $A1$ of the first case is shown. On the right the upper-left 25×5 submatrix of its associated $A2$ is shown—this submatrix is completely determined under the assumption that its rows are lexicographically ordered.

completion of $A5$ found, thereby disproving the existence of the complete matrix A.

2.4 Symmetry groups

Two incidence matrices are said to be *isomorphic* if one can be transformed into the other through row or column permutations. The *symmetry group* of a matrix is the set of row and column permutations that fix all entries of the matrix. When the symmetry group of a submatrix of A (such as $A1$ in Figure 1) is large the search starting from that submatrix tends to include a lot of isomorphic matrices. To avoid searching through a space containing many isomorphic matrices an important optimization is to detect and remove as many symmetries as early in the search as possible.

(Lam, Crossfield, and Thiel 1985) found that up to isomorphism there are exactly 66 possible ways of completing the submatrix $A1$; an explicit list containing each possible case is available in (Kaski and Östergård 2006). Lam et al. show how to remove 21 of those possibilies by various argumentation. Computational searches were performed for each of the remaining 45 possibilities and (Lam, Thiel, and Swiercz 1989) found 639,624 nonisomorphic ways of extending these $A1$s to $A2$s. Our count differs from that of Lam et al.—see Section 5 for details.

3 SAT encoding

We now describe the SAT encoding used in our instances in three parts—corresponding to the three steps of Lam et al.'s search. First, a single SAT instance determines the possibilities for the $A1$s (i.e., the upper left 6×19 matrix in Figure 1). Second, for each possible $A1$ a new instance determines the possibilities for the $A2$s. Third, for each possible $A2$ a new instance determines that the matrix cannot be completed to a full incidence matrix A. For efficiency reasons many constraints are dropped from the third set of instances. This results in a small number of solutions to these instances which are then shown to not complete by adding back in some of the constraints (see Section 4.3).

In each of these instances we use the Boolean variable $a_{i,j}$ to represent that the (i,j)th entry of A contains a 1. Note that since A defines a projective plane none of its lines intersect twice; in other words, for every pair of distinct indices (i,j) there do not exist another pair of distinct indices (k,l) such that all the variables $\{a_{i,k}, a_{i,l}, a_{j,k}, a_{j,l}\}$ are true. Thus, each of our SAT instances include clauses of the form $\neg a_{i,k} \vee \neg a_{i,l} \vee \neg a_{j,k} \vee \neg a_{j,l}$ where i and j are indices of the rows under consideration and j and l are indices of the columns under consideration.

3.1 $A1$ encoding

An $A1$ is a 6×19 matrix with $\{0,1\}$ entries containing exactly five 1s in each row and no two rows having more than a single 1 in the same location. Furthermore, the rows and columns of an $A1$ may be lexicographically ordered without loss of generality (Knuth 2015, cf. exercise 495). Moreover, lexicographic constraints can simply be encoded into Boolean logic (Knuth 2015, cf. equation 169). To enforce the fact that the row sum of each row is 5 we use the sequential counter cardinality encoding (Sinz 2005)—cf. exercise 30 (Knuth 2015).

In order to find all possible $A1$s we exhaustively search for solutions of the above SAT instance. Whenever a solution S is found the *blocking clause* $\bigvee_{S \models p} \neg p$ (where $S \models p$ means the variable p is true in S) is added to the SAT instance and the solver is restarted; this produces 3366 solutions. To check the $A1$s for equivalence we construct the *incidence graph* of the matrix (Kaski and Östergård 2006) and use the library Traces (McKay and Piperno 2014) to discard the $A1$s whose incidence graphs are isomorphic to those of previously found $A1$s.

3.2 $A2$ encoding

An $A2$ is a 37×19 matrix with $\{0,1\}$ entries containing exactly three 1s in each row and no two rows having more than a single 1 in the same location. Furthermore, in the complete incidence matrix A the $A2$ appears directly below an $A1$ and this completely determines the number of 1s that appear in each column (see Figure 1). Because the rows of $A2$ may be permuted freely without loss of generality we assume that the rows of $A2$ are lexicographically sorted. The row sum, column sum, and lexicographic constraints are each encoded in the same way as in the $A1$ encoding.

The $A2$ SAT instances may now be exhaustively solved similar to how the $A1$ SAT instance is solved and this produces over 56 million solutions. Due to the large number of solutions it is much more efficient to employ symmetry removal during the solving process. To this end we adapt the "recorded objects" method of isomorph-free exhaustive generation (Kaski and Östergård 2006) to the SAT context. In order to do this we first split the $A2$ search space into a

number of successive levels, where each level successively fills in more of the $A2$ matrix.

Note that the first row of $A1$ is $1^5 0^{14}$ (as a binary vector) because its columns are lexicographically ordered. Thus the first five columns pairwise intersect in $A1$ and so will not intersect at all in $A2$. It follows that the column sum and lexicographic constraints completely fix the first five columns of $A2$. For example, Figure 2 contains one possible $A1$ and the first five columns of the $A2$ generated by this $A1$. The set of rows of $A2$ can be split into *levels* based on which rows are identical in the first five columns. For example, the levels of the $A2$ in Figure 2 consist of the first row of $A2$ and then the rows 2–4, 5–11, 12–18, 19–25, and the remaining rows 26–37 (consisting of zeros in the first five columns).

Our exhaustive search proceeds by finding completions of the entries in each successive level. At each level isomorphism removal is performed on the completions found at that level. When multiple completions are isomorphic to each other only a single one of those completions is used for the purposes of completing the next level.

When a completion of a level is found, the incidence graph of the $A1$ and $A2$ (up to the given level) is formed and passed to the Traces library. Traces generates a certificate of the incidence graph and if the certificate is new the completion is *recorded* as a new solution of the current level. If the certificate has been previously seen, the incidence graph is compared to the incidence graphs of the previously found completions to verify that it is indeed isomorphic to one of them. Once it has been verified that the completion is isomorphic to a previously found completion the new completion is discarded. Formally, a blocking clause $\bigvee_{S \models p} \neg p$ is added to the SAT instance, where S consists of the assignment formed by the completion to be discarded (up to the given level). The solver is then resumed—the blocking clause being added to the SAT instance on-the-fly, i.e., *programmatically* (Ganesh et al. 2012).

This procedure requires keeping track of the nonisomorphic completions (and their certificates) that are found at each level as the search is progressing but its advantage is that symmetries are detected and removed much earlier than they would otherwise be if isomorphism removal was only performed at the final level. Moreover, the search is still exhaustive in the sense that all nonisomorphic $A2$s will be found. Formally, we state the following theorem whose proof appears in the supplementary material.

Theorem 1. *If the $A2$ SAT instances are solved with isomorphism removal performed after the completion of each level then the solver will record exactly one representative from each equivalence class of $A2$ completions.*

3.3 Main encoding

We now describe our main encoding of determining the nonexistence of A containing a given $A1$ and $A2$. First, there is a unique way of completing $A3$ (assuming a lexicographic ordering of its rows) because its row sums are 1. Similarly, there is a unique way of completing $A4$ because its columns have at most two 1s (having $k \geq 3$ in the rightmost column of Figure 1 is impossible). We complete the columns of $A4$ with column sum 2 first (adding a single column of this form for each pair of lines that do not already intersect in the first 19 columns) and assume a lexicographic ordering of the remaining columns. Figure 3 shows one possible $A1$ and its associated $A4$.

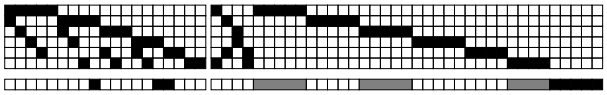


Figure 3: The $A1$ of case 66 (upper left) and its associated $A4$ (upper right). One particular row of an $A2$ is shown (lower left) and the form of its completion in $A5$ is shown (lower right). Each of the gray rectangles contain a single 1 and there are another five 1s on the row—ordered here to appear as soon as possible.

The columns of an $A4$ can be split into *blocks* where the ith block consists of the columns containing a 1 in the ith row. The columns not incident to any of the first six blocks are called *outside* columns. For example, in Figure 3 the first block consists of column 20 and columns 24–28 and the columns 52–56 are outside columns.

The columns of $A5$ that are identical in $A4$ can be taken to appear in lexicographic order without loss of generality—in other words, we assume that the columns within each block (except for those with column sum 2 in $A4$) are lexicographically sorted. Similarly, the rows in $A5$ that are identical in $A3$ are assumed to appear in lexicographic order.

At this point we could simply encode the row sums, column sums, and lexicographic constraints using the same encoding that was used in the $A1$ and $A2$ encodings. The work of Lam et al. imply these instances are unsatisfiable, but we optimized the encoding in order to reduce the amount of computational resources required to prove unsatisfiability.

In particular, we did not use all 92 columns of $A5$ because this resulted in an excessive number of constraints. Instead, we selected between 4 and 6 blocks (see Section 3.4) from each instance and only used the constraints appearing in those blocks. In this context the row sum constraints no longer apply and we ignore the column sum constraints in the bottom (last 68 rows) of $A5$. Instead, we use an alternative encoding that directly enforces incidences that the first 19 columns have in $A5$. We also use an improved lexicographic encoding taking into account the form of the rows or columns being ordered.

Incidence constraints The first 19 columns of A are known in each SAT instance; say C_j denotes the set of row indices of the 1s in column j with $1 \leq j \leq 19$. The axioms of a projective plane imply that any two columns intersect—thus for each pair of column indices (j, k) with $1 \leq j \leq 19$ and k in the blocks we are completing we include the clause $\bigvee_{i \in C_j} a_{i,k}$. Similarly, if R_i denotes the set of column indices of the 1s in row i we include clauses of the form $\bigvee_{j \in R_i} a_{k,j}$ where $7 \leq k \leq 111$ and block i is being completed.

Lexicographic constraints Consider the columns of a block that have a single 1 in $A4$ (e.g., columns 24–28 in

Figure 3). By Figure 1 we know there are exactly three 1s in the first 43 rows of these columns (two of which appear in $A5$). If i and i' are the row indices of the 1s in column j that is not the final column in a block then the lexicographic ordering of the columns implies that $A_{i^*,j+1} = 0$ for all $7 \leq i^* < \min(i, i')$. Translating this into conjunctive normal form, we include the clauses $\neg a_{i,j} \vee \neg a_{i',j} \vee \neg a_{i^*,j+1}$ for all $7 \leq i^* < i' < i \leq 43$ and all columns j that have a single 1 in $A4$ (except for the final column in a block). This generalizes the lexicographic encoding of (Bright et al. 2020b) and is also the encoding used for the row lexicographic constraints.

These constraints uniquely fix certain entries. For example, if the bottom row of Figure 3 was the first row in $A2$ then the leftmost entry in each gray rectangle would have to be 1. In our main instances we reorder the rows of $A2$ to maximize the number of mutually intersecting rows at the top in order to fix as many entries as possible.

Partial isomorphism removal constraints These constraints are optional and do not apply in most cases—including all of the hardest cases—though when they do apply they effectively constrain the search. They encode the "extra partial isomorphism testing" condition of (Lam, Thiel, and Swiercz 1989). Briefly, if two of the first six rows intersect after the first 19 columns then the vector formed by adding (mod 2) these rows to $1^{19}0^{92}$ (as a binary vector) forms a new vector that intersects six rows of A five times each. The specific six rows vary between cases but may quickly be determined in each case by examining $A1$ and $A2$. There are only a small number of ways of completing those six rows and each way must be isomorphic to one of the 66 possibilities for $A1$. If the completion corresponds to a case that has already been solved then a clause blocking that completion can be included in the SAT instance as it cannot lead to a solution. If the $A1$ cases are ordered so that those with the fewest number of $A2$s are solved first then about 22% of $A2$s have all of their completions isomorphic to previously solved cases.

3.4 Block selection

Each of our SAT instances used the columns from 4 to 6 blocks. We used two different methods of selecting the blocks: a simple method which was used in the easiest cases and a more involved method which was used in the hardest cases and which we experimentally found was about 4 to 5 times faster in those cases.

Inside block method This method was used on the cases with two or more columns with column sum 2 in $A4$—these tend to be the easiest cases. This method simply selects five blocks (i.e., ignores a single block) chosen to maximize the number of intersections that occur in the first six rows of the selected blocks. For example, in Figure 3 blocks 5 and 6 have the most intersections in $A4$ so they are chosen. Additionally, blocks 1 and 2 are chosen since they intersect blocks 5 and 6. If possible, the block that is ignored is a block that has no intersections in $A4$; otherwise, one of the remaining blocks with a single intersection in $A4$ is ignored (e.g., block 3 in Figure 3).

Outside block method This method was used on the cases with at most a single column with column sum 2 in $A4$—these tend to be the hardest cases because of the lack of intersections between blocks in $A4$ (which tend to produce conflicts leading to quick proofs of unsatisfiability). Thus, in these cases we used a seventh "outside" block that shares columns with as many of the first six (or *inside*) blocks as possible.

First, we choose a "special" line in the first 37 rows of $A5$ which includes points in as many inside blocks as possible. For example, in Figure 3 the line at the bottom includes points from the blocks 1, 3, and 6, though in some cases a line can be found that is incident to all six inside blocks. The remaining points on the special line become the outside block. For example, in Figure 3 the outside block would include the last five columns of the diagram if the bottom line was selected as the special line.

The SAT instance only uses blocks incident to the special line and the blocks (if any) that intersect in the first six rows. Additionally, in order to reduce the number of constraints one inside block is dropped from the instance (so long as at least four blocks in total are selected). The dropped block is selected to have the fewest number of rows with unassigned entries in common with the other blocks. Such a block is the least likely to produce conflicts—since conflicts between two blocks occur in rows where both blocks have unassigned entries.

4 Certificates

In this section we describe the certificates from our resolution of Lam's problem and how a third party can verify them. We use several kinds of certificates based on the different parts of the search. First, we provide a certificate which specifies the possible cases for $A1$. For each $A1$ we provide another certificate specifying each of the ways (if any) of extending that $A1$ to $A2$, and similarly for each $A2$ we provide a certificate specifying each of the ways (if any) of extending that $A2$ to the selected blocks (as described in Section 3.4) of $A5$. Finally, we provide certificates showing all completions of the selected blocks do not extend to a complete A and thereby resolve Lam's problem. Our certificates are based on the DRAT (deletion resolution asymmetric tautology) format (Wetzler, Heule, and Hunt Jr. 2014) which is a standard format used for verifying the unsatisfiability of SAT instances.

4.1 $A1$ certificate

The $A1$ certificate consists of a DRAT proof that the $A1$ SAT instance adjoined with 3366 blocking clauses (one for each solution of the original instance) is unsatisfiable, thus providing all solutions of the original instance. Furthermore, a separate certificate (produced by the library Traces) contains an explicit permutation of the rows and columns of each solution that produces a unique canonical form in each case. Using these permutations one can verify each solution is isomorphic to one of 66 $A1$s, providing an upper bound on the number of $A1$s that need to be considered—all that is strictly necessary to verify the resolution of Lam's problem.

Traces also provides the elements of the symmetry group of each $A1$. One can verify this output without needing to trust Traces' implementation; it is straightforward to verify that the permutations produced by Traces do in fact fix the entries of $A1$ and one can independently check (e.g., by hand) that the size of the symmetry group is correct.

4.2 $A2$ certificates

The $A2$ certificates consist of DRAT proofs of unsatisfiability for each $A2$ SAT instance adjoined with blocking clauses for every solution found and blocking clauses for every (possibly partial) completion that is isomorphic to a recorded completion (as described in Section 3.2). Accompanying each solution and partial completion is a set of row and column permutations (provided by Traces) for translating the solution or partial completion into a canonical form.

In order to verify one of the certificates, one needs to verify that (1) the DRAT proof does indeed show the unsatisfiability of the augmented SAT instance, and (2) the canonical form of each (partial or complete) solution is equal to the canonical form of one of the recorded nonisomorphic completions.

By Theorem 1 this shows that the set of solutions consists of exactly one solution from each equivalence class of solutions to the original SAT instance. Moreover, each blocked clause is independently justified (without trusting Traces) to block one of the recorded solutions or to block a partial completion isomorphic to a recorded partial completion.

The fact that each of the recorded solutions are nonisomorphic to each other *does* rely on trusting Traces' canonical form. In order to verify the resolution of Lam's problem it is not strictly necessary to verify the recorded $A2$s are nonisomorphic—however, in order to verify (without trusting Traces) that our $A2$ counts did not include extraneous cases we also performed a verification that all recorded solutions were nonisomorphic of each other.

Note that all $A2$s that are isomorphic to a given $A2$ can be generated through the symmetry group of its associated $A1$; up to row permutations any two isomorphic $A2$s must be isomorphic to each other via a permutation in the symmetry group of $A1$. Additionally, as mentioned in Section 4.1, the symmetry group of each $A1$ can be verified without trusting Traces. No two recorded $A2$s are isomorphic to each other under the symmetries of their associated $A1$, thereby showing each of the recorded $A2$s are indeed mutually nonisomorphic. This can be exhaustively verified by applying every $A1$ symmetry group permutation to every recorded $A2$.

4.3 Main certificates

The main certificates consist of DRAT proofs of unsatisfiability for each of the SAT instances described in Section 4.3 adjoined with blocking clauses for the completions (if any) of the blocks selected to appear in the SAT instance. The majority of SAT instances had no completions but in about 3% of them completions were found—see the supplementary material for one explicit such completion.

An individual certificate for each completion can be generated showing that the completion does not extend to a complete incidence matrix A. Alternatively, it is more efficient to generate one certificate for each case $A1$ showing that none of the completions found in that case can be extended. In order to do this, we generate a SAT instance for each $A1$ that includes the variables and constraints from all six inside blocks and those from the outside block columns appearing in each completion. Each completion C found for that $A1$ is specified as a set of incremental assumption unit clauses (Audemard, Lagniez, and Simon 2013). Once the solver finds no solutions while assuming $\bigwedge_{C \models p} p$ it moves on to the next set of assumptions. This "incremental" solve produces a single DRAT certificate which proves the clause $\bigvee_{C \models p} \neg p$ for each completion C under consideration—thereby demonstrating that these completions C cannot be extended to all six inside blocks. This verifies that a full completion of A cannot in fact exist.

5 Results

Our SAT instances are generated by Python scripts that will be published as free software and from which the certificates can be generated; the $A1$ and $A2$ certificates will also be made available on a public repository. Unless otherwise specified we solve the SAT instances using MapleSAT (Liang et al. 2016) and verify the certificates using GRATgen (Lammich 2017) run on Linux using at most 4GB of memory. We solve the main instances on a cluster of Intel E5 cores at 2.1 GHz and solve the other instances on an Intel i7 core at 2.7 GHz.

5.1 $A1$ and $A2$ results

The $A1$ instance can be generated and solved in a few seconds. The resulting certificate (about 1MB) shows that there are 3366 total solutions of the SAT instance and 3300 of them are isomorphic to one of the remaining 66 $A1$s. We label these $A1$s using the case numbers given in (Kaski and Östergård 2006).

The $A2$ instances are generated and solved in about 25 minutes and produce a total of 650,370 nonisomorphic $A2$s. The resulting certificates along with the canonical form labelings provided by Traces total about 7GB. Fifteen cases (4, 10, 14, 19, 28–31, 35, 40, 44, 45, 59, 61, and 62) are found to have no $A2$s. Five cases (32, 38, 54, 57, and 64) imply the existence of a word of weight 16 in A's rowspace (Lam, Crossfield, and Thiel 1985) and previous searches (Carter 1974; Lam, Thiel, and Swiercz 1986) and certificates (Bright et al. 2020b) demonstrates the nonexistence of such a word. Case 52 is eliminated by a theoretical argument (Lam, Crossfield, and Thiel 1985), leaving 45 remaining cases to solve.

Our counts for the number of nonisomorphic $A2$s in the remaining 45 cases match those of (Lam, Thiel, and Swiercz 1989) in all but the eight cases shown in Table 1. The independent verification (Roy 2011) does not provide the number of $A2$s found in each case but their total $A2$ count is inconsistent with both the counts of Lam et al. and the counts provided by our certificates. It is unclear what caused the discrepancy between these searches—the previous searches did not produce certificates and by personal communication

Case	Lam	Our Work	Case	Lam	Our Work
11	7397	7059	39	1010	505
12	10,966	10,635	56	1554	794
16	5958	8040	58	4329	4188
23	8033	7971	66	662	168

Table 1: The $A2$ counts given by (Lam, Thiel, and Swiercz 1989) that differ with the counts given by our work.

we have been informed that the code and data from the previous searches are no longer available.

The method described in Section 4.2 of checking that the generated $A2$s are mutually nonisomorphic generates 56,171,748 total $A2$s. Adding these $A2$s as blocking clauses to the original $A2$ instances produces unsatisfiable instances that provide a second verification that no $A2$s were missed (and without relying on Theorem 1). However, the instances generated in this way took about 150 times longer to solve.

5.2 Main results

The 45 remaining cases have a total of 639,075 nonisomorphic $A2$s between them. SAT instances are generated for each of these and are simplified using the solver CaDiCaL (Biere 2019) run for 20,000 conflicts. In total this simplification uses about 400 hours and determines that 166,408 of the instances are unsatisfiable.

The remaining simplified instances are processed using the "cubing" solver March_cu (Heule et al. 2011). We disabled the default cubing cutoff of this solver in favor of a cutoff based on the number of free variables in the subproblems specified by each cube. More precisely, when the number of free variables in a subproblem drops below a provided bound no more cubing occurs in that subproblem. The cubing bound is controlled by March_cu's `-n` parameter, but we modified March_cu so that the auxillary variables from the cardinality constraints are not considered free. The cutoff bound was experimentally chosen by randomly selecting up to several hundred instances from each case and determining a bound that minimizes the sum of the cubing and conquering times. Ultimately, the cubing solver produces over 312 million cubes and uses about 1200 hours.

The simplified SAT instances are solved using the conquering solver MapleSAT with the cubes produced by March_cu. This requires about 15,000 total core hours or about 16 hours of real time when simultaneously distributed across 30 machines with 32 cores each. In each instance the DRAT proof produced by MapleSAT is concatenated with the simplification proof produced by CaDiCaL. The combined proofs total about 110TB in a binary format and are used to verify that each of the original SAT instances (after adding blocking clauses for each solution found by MapleSAT) are unsatisfiable. It is possible that each solution found by MapleSAT leads to more than one solution of the original SAT instance because variables are eliminated during the simplification process. Regardless, the DRAT proofs show that the solutions found by MapleSAT are exhaustive in the sense that any satisfying assignment of the original SAT instance must extend a solution found by MapleSAT.

MapleSAT finds 24,882 partial solutions and these are all shown to not extend to a full incidence matrix in a total of about 6 minutes. These certificates are about 4GB and are checked with DRAT-trim (Wetzler, Heule, and Hunt Jr. 2014) in forward checking mode. The default backward checking mode can not be used as these certificates are generated using incremental assumptions and therefore prove the negation of each set of assumptions rather than an empty clause.

6 Conclusion

In this paper we have completed a resolution of Lam's problem from finite geometry—the problem of showing the nonexistence of a projective plane of order ten. Extensive searches solved this problem in a landmark result in the 1980s and this result remains one of the most significant results in computational combinatorial classification. Our work improves on these searches by producing certificates that can be verified by a third party using a proof verifier.

In contrast to the previous resolutions of Lam's problem our work is less error-prone in the sense that it does not require writing custom-purpose search algorithms. Instead, we reduce the problem to SAT and use SAT solvers to perform all of the exhaustive searches. Our work demonstrates the benefits of this approach, as we uncover inconsistencies with both the original search and an independent confirmation.

Moreover, our search provides the fastest known demonstration of the nonexistence of primitive weight 19 words in a projective plane of order ten (in part due to increases in computational capacity). Our search shows this nonexistence result using about 24 months on desktop CPUs (at 2.1 GHz), while (Roy 2011) used about 27 months on desktop CPUs (at 2.4 GHz) and (Lam, Thiel, and Swiercz 1989) used about 27 months on a VAX 11/780 and about 3 months on a CRAY-1A.

As previously mentioned our work does not provide a *formal* proof resolving Lam's problem because it relies on some theoretical results that currently have no computer verifiable proofs as well as some unverified scripts that generate and solve the SAT instances. As future work we would like to see a completely formal verification based on a SAT encoding. This will likely be a significant challenge due to the amount of theoretical results that the encoding relies on as well as the extensive parallelization that is seemingly necessary in order to check the proof in a reasonable amount of time.

References

Audemard, G.; Lagniez, J.-M.; and Simon, L. 2013. Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *International conference on theory and applications of satisfiability testing*, 309–317. Springer.

Biere, A. 2019. CaDiCaL at the SAT Race 2019. In Heule, M. J. H.; Järvisalo, M.; and Suda, M., eds., *Proceedings of SAT Race 2019: Solver and Benchmark Descriptions*, vol-

ume B-2019-1. Department of Computer Science, University of Helsinki.

Bright, C.; Cheung, K.; Stevens, B.; Roy, D.; Kotsireas, I.; and Ganesh, V. 2020a. A Nonexistence Certificate for Projective Planes of Order Ten with Weight 15 Codewords. *Applicable Algebra in Engineering, Communication and Computing* 31: 195–213.

Bright, C.; Cheung, K. K. H.; Stevens, B.; Kotsireas, I.; and Ganesh, V. 2020b. Unsatisfiability Proofs for Weight 16 Codewords in Lam's Problem. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 1460–1466. IJCAI Press.

Bruck, R. H.; and Ryser, H. J. 1949. The nonexistence of certain finite projective planes. *Canadian Journal of Mathematics* 1(1): 88–93.

Carter, J. L. 1974. *On the existence of a projective plane of order ten*. Ph.D. thesis, 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.

Dembowski, P. 1968. *Finite geometries*. Springer-Verlag.

Ganesh, V.; O'Donnell, C. W.; Soos, M.; Devadas, S.; Rinard, M. C.; and Solar-Lezama, A. 2012. Lynx: A programmatic SAT solver for the RNA-folding problem. In *International Conference on Theory and Applications of Satisfiability Testing*, 143–156. Springer.

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

Heule, M. J. H.; Kullmann, O.; and Marek, V. W. 2016. Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer. In *International Conference on Theory and Applications of Satisfiability Testing*, 228–245.

Heule, M. J. H.; Kullmann, O.; and Marek, V. W. 2017. Solving Very Hard Problems: Cube-and-Conquer, a Hybrid SAT Solving Method. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 4864–4868. IJCAI Press.

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

Kaski, P.; and Östergård, P. R. J. 2006. *Classification Algorithms for Codes and Designs*. Springer.

Keller, C. 2019. SMTCoq: Mixing Automatic and Interactive Proof Technologies. In Hanna, G.; Reid, D. A.; and de Villiers, M., eds., *Proof Technology in Mathematics Research and Teaching*, 73–90. Cham: Springer International Publishing.

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

Konev, B.; and Lisitsa, A. 2014. A SAT Attack on the Erdős Discrepancy Conjecture. In Sinz, C.; and Egly, U., eds., *Theory and Applications of Satisfiability Testing – SAT 2014*, volume 8561 of *LNCS*, 219–226. Springer International Publishing, Cham. ISBN 978-3-319-09284-3.

Lam, C. W. H. 1990. How reliable is a computer-based proof? *Mathematical Intelligencer* 12(1): 8–12.

Lam, C. W. H.; Crossfield, S.; and Thiel, L. 1985. Estimates of a computer search for a projective plane of order 10. *Congressus Numerantium* 48: 253–263.

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.

Lammich, P. 2017. Efficient verified (UN)SAT certificate checking. In *International Conference on Automated Deduction*, 237–254. Springer.

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.

McKay, B. D.; and Piperno, A. 2014. Practical graph isomorphism, II. *Journal of Symbolic Computation* 60(0): 94–112.

Roy, D. J. 2011. *Confirmation of the Non-existence of a Projective Plane of Order 10*. Master's thesis, Carleton University.

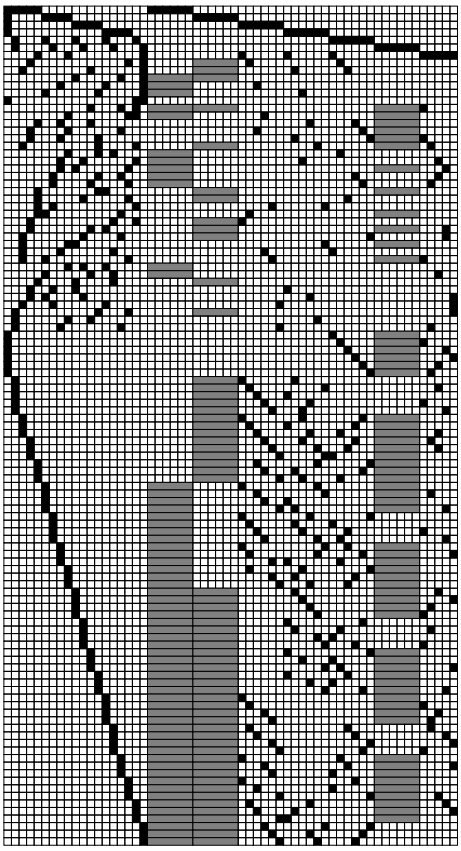
Sinz, C. 2005. Towards an optimal CNF encoding of Boolean cardinality constraints. In *International conference on principles and practice of constraint programming*, 827–831. Springer.

von Staudt, K. G. C. 1856. *Beiträge zur Geometrie der Lage, Erstes Heft*. Nürnberg: Verlag der Fr. Korn'schen Buchhandlung.

Weisstein, E. W. 2002. *CRC Concise Encyclopedia of Mathematics*. CRC Press.

Wetzler, N.; Heule, M. J. H.; and Hunt Jr., 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.

A SAT-based Resolution of Lam's Problem: Supplementary Material



One partial completion of A found by our main search. Black entries represent 1, white entries represent 0, and each gray rectangle contains a single 1. This particular completion consists of an outside block along with the inside blocks 3, 4, and 5. As mentioned in the main text, the rows of the $A2$ have been reordered from lexicographic order—instead the first row is incident to as many inside blocks as possible and the number of mutually intersecting rows in $A2$ at the top is maximized.

Theorem 1. *If the $A2$ SAT instances are solved with isomorphism removal performed after the completion of each level then the solver will record exactly one representative from each equivalence class of $A2$ completions.*

Proof. Because isomorphism removal will be performed after the final level (i.e., after finding a complete $A2$) at most one solution from each equivalence class will be recorded.

Next, we must show that at least one solution from each equivalence class will be recorded. Suppose for the sake of contradiction that there is a solution S that is not recorded (and no solutions isomorphic to S are recorded). Let S_l denote the submatrix of S up to and including level l. Since S_0 (the empty matrix) will be recorded at the beginning of the search and $S = S_6$ is not recorded there must be a level l such that S_{l-1} is recorded but S_l is not recorded.

Since S_{l-1} is recorded the solver will examine all possible ways of completing S_{l-1} to level l and thus will encounter S_l in the search. Since S_l will be found but not recorded, there must be some isomorphic S'_l that was previously recorded. Let φ be the permuation of rows and columns that sends S_l to S'_l and let $\varphi(S)$ denote applying the permutations of φ to S followed by resorting its rows (as the lexicographic ordering of the rows after the first l levels may be disturbed). This resorting does not affect the first l levels of $\varphi(S)$ as these rows are already sorted and lexicographically greater than the rows after the first l levels. Thus $\varphi(S)$ is an $A2$ isomorphic to S whose first l levels consist of S'_l and are therefore recorded.

Now there must be a new level l' with $l < l'$ such that the first $l' - 1$ levels of $\varphi(S)$ are recorded but the first l' levels are not recorded. Repeating the above reasoning we find a recorded matrix isomorphic to the first l' levels of $\varphi(S)$ and proceeding inductively we arrive at a recorded matrix containing all levels and that is isomorphic to S, a contradiction. □

Note that one must be careful when isomorphism removal is performed in order to ensure completeness—the proof of Theorem 1 requires that isomorphism removal is performed following the completion of each level. The theorem will not necessarily hold if isomorphism removal is performed following the completion of each row, for example.