Enumerating Projective Planes of Order 9 with Proof Verification

BENJAMIN CHITTLE and CURTIS BRIGHT, University of Windsor, Canada

This report outlines a method of enumerating projective planes of order nine. The enumeration was previously completed by Lam, Kolesova, and Thiel using highly optimized and customized search code. Despite the importance of this result in the classification of projective geometries, the previous search relied on unverified code and has never been independently verified until now. Our enumeration procedure uses a hybrid satisfiability (SAT) solving and symbolic computation approach. SAT solving performs an enumerative search, while symbolic computation removes symmetries from the search space. Certificates are produced which demonstrate the enumeration completed successfully.

ACM Reference Format:
Benjamin Chittle and Curtis Bright. 2024. Enumerating Projective Planes of Order 9 with Proof Verification. *Manuscript* 1, 1, Article 1 (January 2024), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

In this report, we aim to verify the results of a previous search by Lam et al. [1991] for all finite projective planes of order 9. We do this by reducing the problem to that of Boolean satisfiability (SAT), solving it using a computer program known as a SAT solver, and verifying outputted proofs using a proof verifier. We first give the relevant background and definitions on projective planes, latin squares (which are related to projective planes), and SAT. We then explain how to simplify the problem of searching for projective planes of order 9. Finally, we detail the steps of our search and give a summary of search times and results for each step, showing that our results agree with those of Lam et al.

2 Background

2.1 Projective Planes

Projective planes can be thought of as an extension of the more familiar Euclidean plane with the concept of points at infinity. The result is a space in which every pair of lines intersect at least once, and hence no parallel lines are possible. One can imagine a depiction of railroad tracks on a canvas appearing to converge at the horizon, despite the fact that they are parallel in 3D space. There are several representations for projective planes, and we will provide two which will be useful to us based on the definition below.

Definition 2.1. A projective plane of order n is a collection of $n^2 + n + 1$ points and $n^2 + n + 1$ lines such that

(1) every line contains $n + 1$ points,

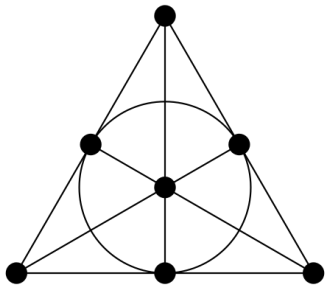

Fig. 1. A projective plane of order 2 (the Fano Plane)

(2) every point is on $n + 1$ lines,
(3) any two distinct lines intersect at exactly one point, and
(4) any two distinct points lie on exactly one line.

Interestingly, it is not possible to generate any projective planes for some orders n (e.g., $n = 6$ [Bruck and Ryser 1949] and $n = 10$ [Lam et al. 1989]). Furthermore, multiple distinct projective planes exist for some orders n (e.g., $n = 9$ [Hall et al. 1959]).

The first alternate representation will be in the form of a bipartite graph. We define two sets of vertices: the first contains a vertex for each point, and the second contains a vertex for each line. We connect point-vertex p to line-vertex l with an edge if and only if the point corresponding to p lies on the line corresponding to l.

We can subsequently derive the second alternate representation by creating an n^2+n+1 by n^2+n+1 incidence matrix from the graph in which each row represents a point and each column represents a line. A 1 in row i and column j indicates that point i lies on line j. A pair of columns or a pair of rows *intersect* if their inner product is exactly 1. A valid projective plane of order n in this form must satisfy the following properties:

(1) each column sum of the matrix is $n + 1$,
(2) each row sum of the matrix is $n + 1$,
(3) every pair of distinct columns of the matrix intersect exactly once, and
(4) every pair of two distinct rows of the matrix intersect exactly once.

We will primarily use this representation of projective planes for the remainder of this report.

2.2 Latin Squares

Latin squares will play an important role in our search, so we define them here.

Definition 2.2. A latin square of order k is a $k \times k$ array consisting of the integers $1, 2, \ldots, k$ such that

(1) each row contains each of the numbers $1, 2, \ldots, k$ exactly once, and
(2) each column contains each of the numbers $1, 2, \ldots, k$ exactly once.

Authors' Contact Information: Benjamin Chittle, chittle5@uwindsor.ca; Curtis Bright, cbright@uwindsor.ca, University of Windsor, Canada.

ACM 1557-7368/2024/1-ART1
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

Fig. 2. A latin square of order 4

As with projective planes, a latin square can be represented as a graph. This is done by creating a vertex for each entry in the latin square and creating an edge between any pair of vertices whose entries lie in the same row, same column, or have the same numeric value.

2.3 SAT

Given a list of clauses, where each clause can be made up of the disjunction variables and their negations, the Boolean satisfiability problem (SAT) asks whether it is possible to assign values of true and false to each variable such that each clause evaluates to true. Despite being a well-known NP-complete problem, heuristic algorithms exist which can efficiently solve SAT instances in many cases. Programs which perform this task, such as MapleSAT, are called SAT solvers.

A useful feature of SAT solvers is their ability to produce proofs or certificates which can be verified by a second program, a proof verifier, to verify the results produced by the solver. By verifying the SAT solver's output, we do not have to rely on the correctness of its programming; instead, if a proof is verified successfully by a proof verifier, we only have to assume the correctness of the proof verifier itself. Importantly, proof verifiers are typically simpler than SAT solvers.

2.4 Structure of Planes of Order 9

A naive search for the projective planes of order 9 would be infeasible due to the large search space of the problem. In order to reduce this search space, it will prove useful to take advantage of the symmetry of projective planes and fix certain values in our incidence matrix. This will be done based on Kolesova's standard form of planes of order 9. The proof of correctness for this form is not provided here, but it shows that any plane of order 9 can be rearranged into this form using only column and row permutations. In other words, any plane of order 9 will be *isomorphic* to a plane in this form.

We will first focus on the leading 27 columns of this standard form. There are eight 8×8 blocks of missing values $B^1, B^2, \ldots, B^8$. However, these blocks have a nice structure: it can be shown that each of these blocks must be a distinct permutation matrix. This is where we connect projective planes of order 9 to latin squares of order 8: we can generate a distinct permutation matrix from each row of a given latin square by treating the numbers in each row as a reordering of the row indices of the identity matrix. In other words, the set of all latin squares of order 8 can be reduced to the set of all partial (up to column 27) projective planes of order 9. Due to symmetry, isomorphic latin squares will reduce to isomorphic projective planes, so we will want to identify and remove such "duplicate" squares before performing the reduction. Our search for the planes of order 9 will begin here.

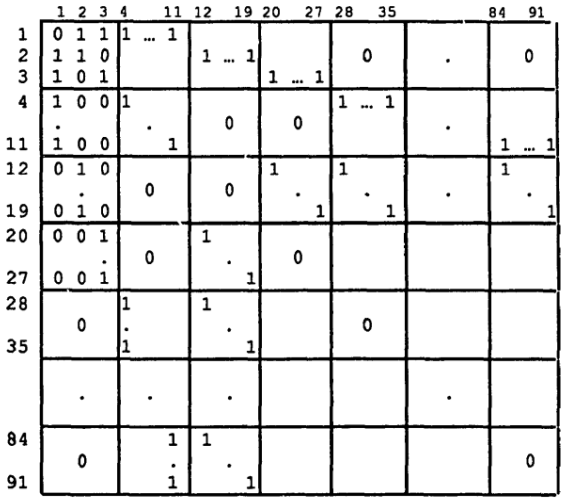

Fig. 3. The standard form for projective planes of order 9 as shown by Kolesova 1989.

3 Search Procedure

We perform our search for all projective planes of order 9 in 3 steps. First we enumerate all latin squares of order 8. Then, for each latin square, we generate a corresponding partial plane up to column 27 and attempt to extend it column 40. Finally, we attempt to extend each remaining partial plane from column 40 to a full projective plane (column 91).

The search component of each step is accomplished by reducing the problem to SAT and performing an exhaustive search for solutions using MapleSAT. Before moving to the next step, we also perform isomorphism removal by reducing every solution to its graph form and checking for isomorphism using nauty. Finally, we verify all MapleSAT proofs using DRAT-trim and all isomorphism removal using a custom Python script.

3.1 Latin Squares of Order 8

We begin by enumerating the latin squares of order 8. In order to reduce the problem to SAT, we need to define an encoding in Boolean logic for latin squares and for the conditions that determine a valid latin square.

First we define an encoding for latin squares. Let $(A_{i,j})$ be a latin square of order 8. We encode A into 8^3 Boolean variables $l_{i,j,k}$ for $1 \leq i, j, k \leq 8$ where $l_{i,j,k}$ is true if $A_{i,j} = k$ and false otherwise. Since MapleSAT uses non-zero integers to represent variables, in practice, we represent the variable $l_{i,j,k}$ as the result of the expression $(i \times 8^2 + j \times 8 + k)$ if $l_{i,j,k}$ is true and $-(i \times 8^2 + j \times 8 + k)$ otherwise.

Then, we define an encoding for the requirements of a latin square. Namely, we must encode the fact the the numbers $1, 2, \ldots, 8$ must each appear exactly once per row and once per column. This is accomplished using two sets of clauses.

The first set of clauses guarantees that each number appears at least once per row and column. For $1 \leq i, j, k \leq 8$ we define

$$l_{1,j,k} \vee l_{2,j,k} \vee \ldots \vee l_{8,j,k}$$

and

$$l_{i,1,k} \vee l_{i,2,k} \vee \ldots \vee l_{i,8,k}$$

The second set of clauses guarantees that each number appears at most once per row and column. For $1 \leq i, i', j, j', k \leq 8$ where $i \neq i'$ and $j \neq j'$ we have

$$\neg l_{i,j,k} \vee \neg l_{i',j,k}$$

and

$$\neg l_{i,j,k} \vee \neg l_{i,j',k}$$

Combining the above two sets of clauses allows us to generate the set of all latin squares of order 8. However, as with projective planes, we will exploit the symmetry of latin squares and perform the search in several steps to reduce the search space. Bash and Python scripts are used to generate clauses and manage the execution of MapleSAT, DRAT-trim, nauty, and our custom isomorphism verification.

We start by fixing the entries of the first row of the latin square as the numbers $1, 2, \ldots, 8$ in their natural order. We include these as unit clauses with the rest of our SAT clauses. We then use MapleSAT to extend from row one to row two and generate all partial 2×8 latin squares that can follow from the first row. After performing isomorphism removal and proof verification, we repeat this process to extend to rows three, four, and finally eight. After isomorphism removal in the final step, we are left with 283,657 latin squares of order 8.

3.2 Column 40

After enumerating the latin squares of order 8, we have a starting point in our search for projective planes of order 9. In this step, for each of the 283,657 latin squares, we generate a corresponding partial projective plane up to column 27 and attempt to extend it to column 40.

In order to generate a partial plane up to column 27, we embed each latin square into the first 27 columns of the standard form incidence matrix. We can generate a permutation matrix from each row of a given latin square by treating the numbers in each row of the square as a reoordering of the row indices of the identity matrix. These permutation matrices are placed in the missing blocks of the standard form in the same order as the rows. We are now almost ready to extend to column 40; however, we must first define another encoding into SAT, this time for projective planes of order 9 and their properties.

An incidence matrix $(A_{i,j})$ of a plane of order 9 can be encoded as $9^2 + 9 + 1 = 91$ Boolean variables by simply defining a variable $a_{i,j}$ for each entry in A where $a_{i,j}$ is true if and only if $A_{i,j} = 1$.

Encoding the definition of a projective plane is trickier. For our approach, it is sufficient to encode only requirements (3) and (4): that every pair of distinct columns (rows, respectively) intersect exactly once. We accomplish this using two sets of clauses.

The first set of clauses, called the *quad free* clauses, guarantees that no intersection happens more than once.

$$\neg a_{i,j} \vee \neg a_{i',j} \vee \neg a_{i,j'} \vee \neg a_{i',j'}$$

for $i < i'$ and $j < j'$.

If one of these clauses were false, we would have a "quad," or a rectangle in the incidence matrix whose corners are 1's. This would indicate multiple intersections of the same columns or rows and violate conditions (3) and (4).

The second set of clauses, called the *innerproduct* clauses, guarantees that each of the missing columns intersects each of the known columns exactly once. If column $j \leq 27$ has 1's in the entries $(i_1, j), (i_2, j), \ldots, (i_{10}, j)$, then for $j' > 27$, we include the clause

$$a_{i_1,j'} \vee a_{i_2,j'} \vee \ldots \vee a_{i_{10},j'}$$

which ensures that column j' intersects column j at least once. It should be noted that a different set of innerproduct clauses need to be generated for each plane because each has different entries in the blocks determined by the latin squares.

Combining these clauses with the unit clauses of a given 27-column partial plane allows us to use a SAT solver to find all possible extensions of the give partial plane to column 40, if any exist. We use a similar workflow to perform this extension as we did with latin squares; however, in practice, the size of this part of the search required the use of 568 CPU's on a Compute Canada cluster, each extending 500 of the 283,657 partial planes, in order to perform the full search in a reasonable amount of time. After extending the 283,657 27-column partial planes, we were left with 7,869 40-column partial planes. Only 3,057 remained after isomorphism removal.

3.3 Column 91

The same process used to extend from column 27 to column 40 is used to extend from column 40 to column 91. Of the 3,057 partial planes up to column 40, 107 full planes were found. Isomorphism removal yielded 4 distinct projective planes of order 9. This agrees with the result found by Lam et al.

4 Results

This section provides a brief summary of our results and timings for each stage of the search.

	Time	Solutions	Verification
Latin Squares	20 minutes	42,878,753	45 minutes
Column 40	0.93 years	7,869	3.87 years
Column 91	6 hours	107	18 hours

Table 1. Summary of solving SAT instances using MapleSAT and proof verification using DRAT-trim.

	Time	Remaining Solutions
Latin Squares	80 hours	283,657
Column 40	1 minute	3,057
Column 91	5 minutes	4

Table 2. Summary of isomorphism removal using nauty.

References

R. H. Bruck and H. J. Ryser. 1949. The Nonexistence of Certain Finite Projective Planes. *Canadian Journal of Mathematics* 1, 1 (Feb. 1949), 88–93. https://doi.org/10.4153/cjm-1949-009-2

Marshall Hall, J. Dean Swift, and Raymond Killgrove. 1959. On projective planes of order nine. *Math. Comp.* 13, 68 (1959), 233–246. https://doi.org/10.1090/s0025-5718-1959-0107208-8

Galina I. Kolesova. 1989. *Enumeration of the Finite Projective Planes of Order Nine.* Master's thesis. Concordia University.

C. W. H. Lam, G. Kolesova, and L. Thiel. 1991. A computer search for finite projective planes of order 9. *Discrete Mathematics* 92, 1-3 (Nov. 1991), 187–195. https://doi.org/10.1016/0012-365x(91)90280-f

C. W. H. Lam, L. Thiel, and S. Swiercz. 1989. The Non-Existence of Finite Projective Planes of Order 10. *Canadian Journal of Mathematics* 41, 6 (Dec. 1989), 1117–1123. https://doi.org/10.4153/cjm-1989-049-4