Searching for projective planes with computer algebra and SAT solvers

Curtis Bright$^{1,2,3}$  
Brett Stevens$^1$  
Ilias Kotsireas$^3$  

Kevin Cheung$^1$  
Dominique Roy$^1$  
Vijay Ganesh$^2$

$^1$Discrete Mathematics Group, Carleton University  
$^2$Computer Aided Reasoning Group, University of Waterloo  
$^3$Computer Algebra Research Group, Wilfrid Laurier University

July 19, 2019
SAT: Boolean satisfiability problem

SAT solvers: Clever brute force
Effectiveness of SAT solvers

Many problems that have nothing to do with logic can be effectively solved by reducing them to Boolean logic and using a SAT solver.
Effectiveness of SAT solvers

Many problems that have nothing to do with logic can be effectively solved by reducing them to Boolean logic and using a SAT solver.

Examples

- Hardware and software verification
- Scheduling subject to constraints
- Finding or disproving the existence of combinatorial objects
Effectiveness of SAT solvers

Many problems that have nothing to do with logic can be effectively solved by reducing them to Boolean logic and using a SAT solver.

Examples

- Hardware and software verification
- Scheduling subject to constraints
- Finding or disproving the existence of combinatorial objects

Limitations

Lack of expressiveness, and SAT solvers perform poorly on highly symmetric problems.
CAS: Computer algebra system
Symbolic mathematical computing
Example

What is the automorphism group of this graph?

Maple returns $\langle (2, 5), (3, 8)(4, 7), (1, 2)(3, 4)(5, 6)(7, 8) \rangle$.

Limitations
CASs are not optimized to do large (i.e., exponential) searches.
Example

What is the automorphism group of this graph?

MAPLE returns

\[\langle (2, 5), (3, 8)(4, 7), (1, 2)(3, 4)(5, 6)(7, 8) \rangle.\]
Example

What is the automorphism group of this graph?

Maple returns

\[ \langle (2, 5), (3, 8)(4, 7), (1, 2)(3, 4)(5, 6)(7, 8) \rangle. \]

Limitations

CASs are not optimized to do large (i.e., exponential) searches.
SAT + CAS

Brute force + Knowledge
MathCheck

Our SAT+CAS system MathCheck has constructed over 100,000 various combinatorial objects. For example, this \(\{\pm 1\}\)-matrix with pairwise orthogonal rows:
Results first shown by MathCheck

- Found the smallest counterexample of the Williamson conjecture.
- Verified the even Williamson conjecture up to order 70.
- Found three new counterexamples to the good matrix conjecture.
- Verified the best matrix conjecture up to order seven.
- Verified the Ruskey–Savage conjecture up to order five.
- Verified the Norine conjecture up to order six.

Details available at:

uwaterloo.ca/mathcheck
A projective plane is a set of points and lines and a relation between points and lines such that:

- There is a unique line between any two points.
- Any two lines meet at a unique point.
Projective planes of order $n$

A finite projective plane is a collection of $n^2 + n + 1$ lines and $n^2 + n + 1$ points such that:

- There are $n + 1$ points on each line.
- There are $n + 1$ lines through each point.
Incidence matrix representation

Projective plane of order 2:

\[
\begin{bmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

- \(\{0, 1\}\)-matrix of size 7 \(\times\) 7.
- Each row (representing lines) contains exactly three 1s.
- Each column (representing points) contains exactly three 1s.
For what orders do projective planes exist?
...every known plane has prime power order ... [and] has been constructed in one way or another from a finite field...
The Bruck–Ryser theorem

If $n$ is the order of a projective plane and $n \equiv 1, 2 \pmod{4}$ then $n$ is the sum of two squares.
Projective planes of small orders

\[\begin{array}{cccccccccccccc}
2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
\checkmark & \checkmark & \checkmark & \checkmark & \text{X} & \checkmark & \checkmark & \checkmark & ? & ? & \checkmark & \text{X} & ? \\
\end{array}\]

✓ Finite field construction
X Bruck–Ryser theorem
The first critical value of \( n \) is \( n = 10 \). A thorough investigation of this case is currently beyond the facilities of computing machines.

Marshall Hall Jr.

*Finite Projective Planes*

1955
Enter coding theory

“We’ve devised a new security encryption code. Each digit is printed upside down.”
Codewords

A *codeword* generated by a projective plane is a vector in the row space of its incidence matrix (over $F_2 = \{0, 1\}$).

The *weight* of a codeword is the number of 1s it contains.
A search for weight 15 codewords

In 1970, MacWilliams, Sloane, and Thompson showed that a projective plane of order ten must generate weight 15, 16, or 19 codewords.

Furthermore, they used three hours of computing on a mainframe computer to show that codewords of weight 15 do not exist.
Other searches

We know of three other searches with code we could run:

▶ [Dominique Roy, 2005] Implementation in C, runs in 78 minutes.
▶ [Casiello, Indaco, and Nagy, 2010] Implementation in GAP, runs in 7 minutes.
▶ [Xander Perrott, 2016] Implementation in Mathematica, runs in 55 minutes.
Other searches

We know of three other searches with code we could run:

▶ [Dominique Roy, 2005] Implementation in C, runs in 78 minutes.
▶ [Casiello, Indaco, and Nagy, 2010] Implementation in GAP, runs in 7 minutes.
▶ [Xander Perrott, 2016] Implementation in Mathematica, runs in 55 minutes.

Our result

We verified the search using a SAT+CAS method in seconds.
Searches for weight 16 codewords

In 1974, Carter performed a partial search for weight 16 codewords using approximately 140 hours on a mainframe computer.
Searches for weight 16 codewords

In 1986, Lam, Thiel, and Swiercz completed the weight 16 search using about 1,900 hours of computing on a VAX-11/780.
Searches for codewords of weight 19

In 1989, Lam, Thiel, and Swiercz used about 19,200 hours on a VAX-11/780 and 2,000 hours on a CRAY-1A supercomputer run by the Institute for Defense Analyses to show that no weight 19 codewords exist.
I’m sorry, but that’s the way it goes. The order 12 case is open, by the way, but a computer attack along the same lines would take ten thousand million times as long.

Ian Stewart

Another Fine Math
You’ve Got Me Into…

1992
Using SAT solvers for combinatorial search
Surprisingly, SAT solving is getting so strong that indeed SAT solvers seem today the best solution in most cases.

Marijn Heule, Oliver Kullmann, Victor Marek
Solving Very Hard Problems: Cube-and-Conquer, a Hybrid SAT Solving Method
2017
If a weight 15 codeword exists, MacWilliams, Sloane, and Thompson showed that the first 21 rows (up to equivalence) of the incidence matrix of a projective plane of order ten are exactly:

```
1111100000000000000000000000000000000000000000000000000000000000000000000000000
1000111100000000000000000000000000000000000000000000000000000000000000000000000
0100110011100000000000000000000000000000000000000000000000000000000000000000000
0010001001001100000000000000000000000000000000000000000000000000000000000000000
0001000100101010000000000000000000000000000000000000000000000000000000000000000
0000100010010110000000000000000000000000000000000000000000000000000000000000000
1000000010000100000000000000000000000000000000000000000000000000000000000000000
0100001000100000000000000000000000000000000000000000000000000000000000000000000
0010000100100010000000000000000000000000000000000000000000000000000000000000000
0001000011000001000000000000000000000000000000000000000000000000000000000000000
00001000000100100000000000000000000000000000000000000000000000000000000000000000
0000010001001010100000000000000000000000000000000000000000000000000000000000000
```
The next 24 rows are of this form, where blanks are unknown entries:
Consider lines 1 and 28:

11111000000000000000...00000000000000000000000011111000...
00000000001000000 00...00 0000 00

There must be some point that is on both of these lines.

A 1 must appear here.

In Boolean logic:

\[ a \lor b \lor c \lor d \lor e \lor f \]
Consider lines 1 and 28:

\begin{verbatim}
11111000000000000000...000000000000000011111000...
0000000001000000 00...00 0000 00
\end{verbatim}

There must be some point that is on both of these lines.
SAT encoding

Consider lines 1 and 28:

```
11111000000000000000...00000000000000000011111000...
0000000001000000 00...00 0000 00 ****** ...
```

There must be some point that is on both of these lines.

A 1 must appear here.
Consider lines 1 and 28:

11111000000000000000...0000000000000000000011111000...
0000000001000000 00...00 0000 00 abcdef ... 

There must be some point that is on both of these lines.

A 1 must appear here.

In Boolean logic:

\[ a \lor b \lor c \lor d \lor e \lor f \]
Consider lines 22 and 40:

\[
\begin{align*}
1000000000000000 & \ldots \\
0000000000000010000 & 0000 \ldots \\
\end{align*}
\]
Consider lines 22 and 40:

\[
\begin{align*}
1000000000000000 & \ldots \\
00000000010000000 & \ldots \\
\end{align*}
\]

Exactly one point must appear on both of these two lines.

Exactly
Consider lines 22 and 40:

\[
\begin{array}{c}
100000000000000000000* \quad * \\
000000000000000000000100000*00000* \\
\end{array}
\]

*Exactly* one point must appear on both of these two lines.

These cannot all simultaneously be 1.
Consider lines 22 and 40:

\[ \begin{array}{cccc}
100000000000000 & a & b & \ldots \\
000000000000001 & c & 00000 & d & \ldots 
\end{array} \]

*Exactly* one point must appear on both of these two lines.

These cannot all simultaneously be 1.

In Boolean logic:

\[ \neg a \lor \neg b \lor \neg c \lor \neg d \]
Solving the SAT instance

Up to 27 rows, the SAT instance has about 150 unknown variables, 1000 clauses, and over $10^{18}$ solutions.

However, many columns are rows are identical and permuting them produces other equivalent solutions.
Symmetry breaking

Using appropriate row/column permutations, we can assume the first 27 rows are:
Symmetry breaking

Using appropriate row/column permutations, we can assume the first 27 rows are:

These rows can be sorted using row permutations.
Using appropriate row/column permutations, we can assume the first 27 rows are:

These columns can be sorted using column permutations.
Symmetry breaking

Using appropriate row/column permutations, we can assume the first 27 rows are:

Now 42,496 solutions.
Solving the SAT instances

The instances with up to 42 rows can now be solved in seconds. These instances are all satisfiable.
Solving the SAT instances

The instances with up to 42 rows can now be solved in seconds. These instances are all satisfiable.

The instance with 43 rows is unsatisfiable and requires about 7 minutes to solve.
Solving the SAT instances

The instances with up to 42 rows can now be solved in seconds. These instances are all satisfiable.

The instance with 43 rows is unsatisfiable and requires about 7 minutes to solve.

This verifies the search of MacWilliams–Sloane–Thompson, but we can do better...
Using CAS to speed up the search
CAS symmetry breaking

A CAS can be used to find symmetries of the partially filled incidence matrix.

There are 48 symmetries that fix the already assigned entries in the first 27 rows.
Isomorphism blocking

When the SAT solver finds a solution of the first 27 rows, we use the 48 symmetries to block all isomorphic solutions.
Isomorphism blocking

When the SAT solver finds a solution of the first 27 rows, we use the 48 symmetries to block all isomorphic solutions.

The SAT solver finds 1,021 inequivalent solutions in 2.5 seconds.
Isomorphism blocking

When the SAT solver finds a solution of the first 27 rows, we use the 48 symmetries to block all isomorphic solutions.

The SAT solver finds 1,021 inequivalent solutions in 2.5 seconds.

It takes just 6.5 seconds to show that the SAT instances up to 43 rows generated by these 1,021 solutions are unsatisfiable.
Isomorphism blocking

When the SAT solver finds a solution of the first 27 rows, we use the 48 symmetries to block all isomorphic solutions.

The SAT solver finds 1,021 inequivalent solutions in 2.5 seconds.

It takes just 6.5 seconds to show that the SAT instances up to 43 rows generated by these 1,021 solutions are unsatisfiable.

This verifies MacWilliams–Sloane–Thompson’s search in 9 seconds.
Weight 16 searches

In 1974, Carter spent \( \sim 140 \) hours on a mainframe. We verified his searches in 7 hours.

In 1986, Lam, Thiel, and Swiercz spent \( \sim 1,900 \) hours of computing on a VAX-11/780. We verified their searches in 124 hours.
In 1974, Carter spent $\sim 140$ hours on a mainframe. We verified his searches in 7 hours.

In 1986, Lam, Thiel, and Swiercz spent $\sim 1,900$ hours of computing on a VAX-11/780. We verified their searches in 124 hours.

This verifies the weight 16 search (also verified by Roy using $\sim 16,000$ hours on a desktop in 2010) in 131 hours.
Weight 16 searches

In 1974, Carter spent $\sim 140$ hours on a mainframe. We verified his searches in 7 hours.

In 1986, Lam, Thiel, and Swiercz spent $\sim 1,900$ hours of computing on a VAX-11/780. We verified their searches in 124 hours.

This verifies the weight 16 search (also verified by Roy using $\sim 16,000$ hours on a desktop in 2010) in 131 hours.

Next steps

We are currently working on verifying the weight 19 searches to produce a fully independent verification of the order ten search.