SAT Solvers and Combinatorics Problems

Curtis Bright
University of Windsor

Computational Proof Techniques
for Combinatorics on Words
CanaDAM 2021
May 25, 2021

SAT:

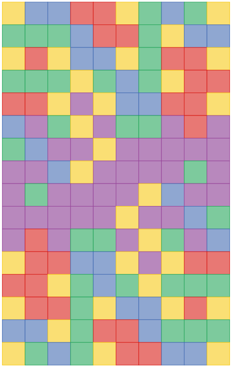
Boolean satisfiability problem

SAT solvers use clever trial-and-error to find solutions

Effectiveness of SAT solvers

SAT solvers are incredibly effective at solving some kinds of search problems that have nothing to do with logic.[1]

- Discrete optimization
- Hardware and software verification
- Combinatorial problems like colouring the positive integers as far as possible so that a, b, and $a+b$ are never all the same colour[2]



Additionally, SAT solvers produce unsatisfiability certificates when no solutions exist.

[1] C. Bright, J. Gerhard, I. Kotsireas, V. Ganesh. Effective Problem Solving Using SAT Solvers. *Maple in Mathematics Education and Research*, 2019.

[2] M. Heule. Schur Number Five. *AAAI 2018.*

Combinatorial problems solved by SAT

2008 Kouril and Paul determined the sixth van der Waerden number on two colours.

2013 Bundala and Zavodny computed optimal sorting networks for up to sixteen inputs.

2014 Konev and Lisitsa solved a special case of the Erdős discrepancy conjecture.

2016 Heule, Kullmann, and Marek solved the Boolean Pythagorean triples problem.

2016 Bright et al. determined the smallest counterexample of the Williamson conjecture.

2018 Heule computed the fifth Schur number.

2019 Bright, Kotsireas, and Ganesh found the first Williamson matrices of order 70.

2020 Heule et al. resolved Keller's conjecture.

2021 Bright et al. gave the first certifiable solution of Lam's Problem.

A colouring challenge

Can you colour the positive integers using a fixed number of colours while ensuring that any k integers in arithmetic progression

$$a,\ a+b,\ a+2b,\ \ldots,\ a+(k-1)b$$

are not all coloured the same way?

A colouring challenge

Can you colour the positive integers using a fixed number of colours while ensuring that any k integers in arithmetic progression

$$a,\ a+b,\ a+2b,\ \ldots,\ a+(k-1)b$$

are not all coloured the same way?

In 1927, van der Waerden proved no!

Finite van der Waerden colouring

With two colours and $k = 3$ you can colour the positive integers up to eight with the colouring

$$01100110 \qquad \text{(represented as a binary word).}$$

This cannot be extended farther; either way produces k integers of the same colour in arithmetic progression:

$$\underset{\uparrow}{\mathbf{0}}110\underset{\uparrow}{\mathbf{0}}110\underset{\uparrow}{\mathbf{0}} \qquad\qquad 01\underset{\uparrow}{\mathbf{1}}00\underset{\uparrow}{\mathbf{1}}10\underset{\uparrow}{\mathbf{1}}$$

Van der Waerden numbers

The *var der Waerden number* $W_{r,k}$ is the largest number such that there exists an r-colouring of $\{1, \ldots, W_{r,k} - 1\}$ without colouring k integers in arithmetic progression in the same way.

For example, $W_{2,3} > 8$ since 01100110 is a 2-colouring avoiding arithmetic progressions of length 3 of the same colour.

In fact, this is the longest possible such colouring, so $W_{2,3} = 9$. A SAT solver can be used to provide a certificate showing $W_{2,3} \leq 9$.

Specifying a problem in SAT

Most modern SAT solvers require specifying a problem as logical *clauses* (e.g., $x \vee y \vee \neg z$) written

$$\ell_1 \vee \cdots \vee \ell_n$$

which is true when at least one ℓ_i is true. Each ℓ_i must be a single variable or negated variable.

Given a set of clauses a SAT solver will search for an assignment to the variables making **all** of the clauses true.

Van der Waerden in SAT

Let r_i be a variable that is true exactly when i is coloured red.

We want to avoid colouring k integers in arithmetic progression (e.g., 1, 2, 3 with $k = 3$) all the same colour.

1, 2, 3 not all coloured blue: $r_1 \vee r_2 \vee r_3$

1, 2, 3 not all coloured red: $\neg r_1 \vee \neg r_2 \vee \neg r_3$

Similarly, we include clauses of this form for all triples of integers in arithmetic progression.

Complete encoding

Suppose we want to use a SAT solver to test if $W_{2,k} > n$.

Then we use the SAT instance defined by the clauses

$$r_a \lor r_{a+b} \lor \cdots \lor r_{a+(k-1)b}$$

$$\neg r_a \lor \neg r_{a+b} \lor \cdots \lor \neg r_{a+(k-1)b}$$

for all $1 \leq a, b \leq n$ with $a + (k-1)b \leq n$.

If the instance is satisfiable then $W_{2,k} > n$ and an explicit colouring of $\{1, \ldots, n\}$ is found; otherwise $W_{2,k} \leq n$.

PySAT implementation

PySAT is a Python package that can be used to generate and solve SAT instances. The following code tests if $W_{2,k} > n$:

```python
from pysat.solvers import Solver
with Solver() as s:

  # Run over all arithmetic progressions
  for b in range(1, n+1):
    for a in range(1, n-(k-1)*b+1):
      # Form indices in arithmetic progression
      indices = range(a, a+(k-1)*b+1, b)
      s.add_clause(indices)
      s.add_clause([-x for x in indices])

  result = s.solve()
```

Finding $W_{2,k}$ exactly

To determine the value of $W_{2,k}$, start n at k and increase n by 1 until you are able to show $W_{2,k} \leq n$.

Clauses generated from previous values of n are still valid and can be reused—only need to add clauses that include the integer n.

```
# Avoid monochromatic arithmetic progressions
# that include the integer n
for b in range(1, n):
  indices = range(n, 0, -b)[:k]
  if len(indices) == k:
    # Only add clauses of length k
    s.add_clause(indices)
    s.add_clause([-x for x in indices])
```

Results

Results of using PySAT to find small van der Waerden numbers:

$W_{2,2} = 3$	0.00 seconds
$W_{2,3} = 9$	0.00 seconds
$W_{2,4} = 35$	0.00 seconds
$W_{2,5} = 178$	56.34 seconds
$W_{2,6} > 214$	37.99 seconds
$W_{2,6} > 215$	9749.85 seconds

In 2008, Kouril and Paul used a custom SAT solver and a cluster of over 200 machines for about 250 days to show that $W_{2,6} = 1132$.

Finite projective planes

A *quad-free* matrix contains no rectangle with 1s in the corners.

A *finite projective plane* is equivalent to a quad-free $(0, 1)$-matrix with the same number of 1s in each row and column.

1	1	0
1	0	1
0	1	1

order 1

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

order 2

1	0	0	0	1	0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	1	0	1	0	0	0
0	1	0	0	0	1	1	0	0	1	0	0	0
1	0	0	0	0	1	0	1	0	0	1	0	0
0	1	0	1	0	0	0	0	1	0	1	0	0
0	0	1	0	1	0	1	0	0	0	1	0	0
1	0	0	1	0	0	1	0	0	0	0	1	0
0	1	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1	0
0	0	0	1	1	1	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1	1	0	0	0	1
1	1	1	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	1	1	1

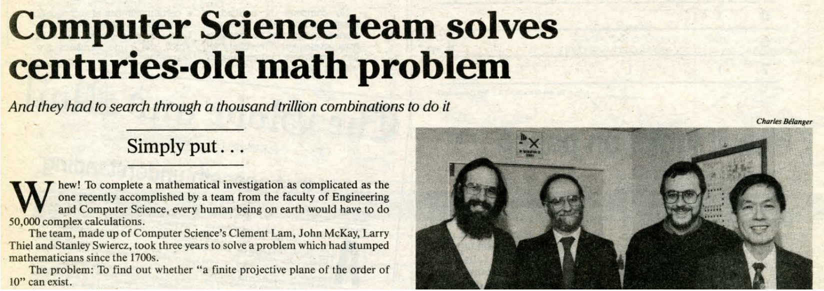
order 3

In order n, each row and column contains $n + 1$ ones.

Projective planes of small orders

1	2	3	4	5	6	7	8	9	10
✓	✓	✓	✓	✓	✗	✓	✓	✓	✗

Lam's problem



Computer Science team solves centuries-old math problem

And they had to search through a thousand trillion combinations to do it

Charles Bélanger

Simply put . . .

Whew! To complete a mathematical investigation as complicated as the one recently accomplished by a team from the faculty of Engineering and Computer Science, every human being on earth would have to do 50,000 complex calculations.

The team, made up of Computer Science's Clement Lam, John McKay, Larry Thiel and Stanley Swiercz, took three years to solve a problem which had stumped mathematicians since the 1700s.

The problem: To find out whether "a finite projective plane of the order of 10" can exist.

Resolution of Lam's problem

Lam et al.[3] used custom-written software to show that a projective plane of order ten does not exist.

We must trust the searches ran to completion—the authors were upfront that mistakes were a real possibility.

Using a SAT solver, we generated the first certifiable resolution of Lam's problem.[4]

[3] C. Lam, L. Thiel, S. Swiercz. The Nonexistence of Finite Projective Planes of Order 10. *Canadian Journal of Mathematics*, 1989.

[4] C. Bright, K. Cheung, B. Stevens, I. Kotsireas, V. Ganesh. A SAT-based Resolution of Lam's Problem. *AAAI 2021*.

Lam's problem encoding

If $x_{i,j}$ represents that entry (i,j) of the projective plane contains a 1 then specifying a matrix is quad-free can be done using the clauses

$$\neg x_{i,j} \lor \neg x_{i,j'} \lor \neg x_{i',j} \lor \neg x_{i',j'}$$

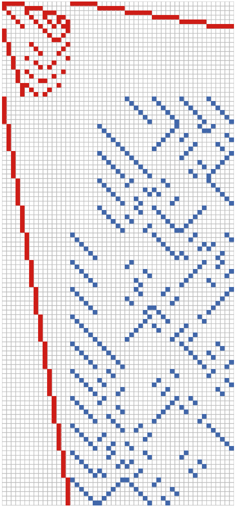
for all distinct pairs of indices (i,j) and (i',j').

The constraints that there are exactly eleven 1s in each row and column are reformulated and expressed in a convenient way for a SAT solver.

Discrepancies

The lack of verifiable certificates has real consequences. We found discrepancies with the intermediate results of both Lam's search and an independent verification from 2011.

On the right is a 51-column partial projective plane of order ten said to not exist in 2011—but we found with a SAT solver.



Conclusion

Many problems in combinatorics stand to benefit from fast and verifiable search tools.

Don't reinvent the wheel! It's hard to beat a SAT solver at search.

A major issue with SAT solvers is that not all mathematical constraints can effectively be expressed in Boolean logic—this can be overcome by combining SAT with computer algebra.[5]

Thank You!

`curtisbright.com`

[5] C. Bright, I. Kotsireas, V. Ganesh. SAT Solvers and Computer Algebra Systems: A Powerful Combination for Mathematics. *CASCON 2019*.