Faster SAT solving with applications to Sudoku

Background

- A SAT solver is a program that takes as input a formula in Boolean logic and returns an assignment to the variables that makes the formula true (if one exists).

```
# SAT Solving Example
with(Logic):
F := (a &or b) &and c;
Satisfy(F);
```

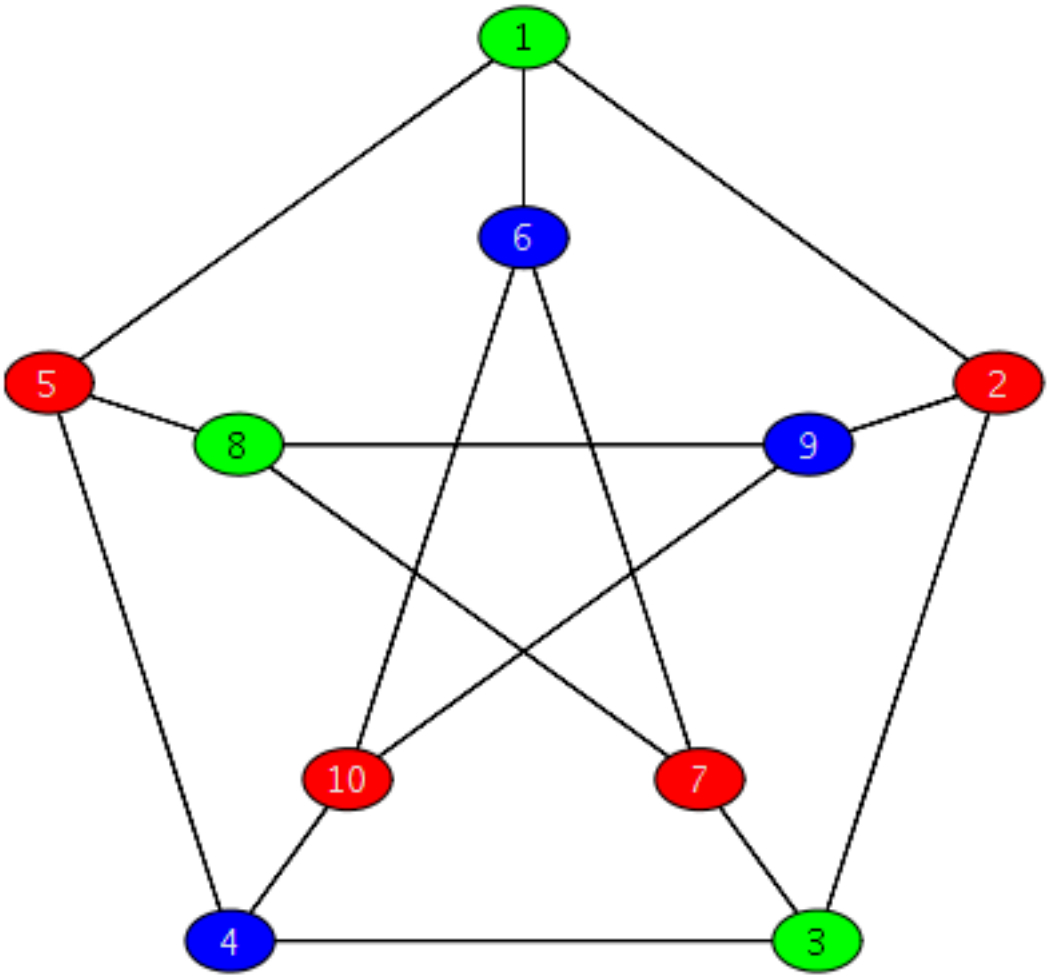
$$F := (a \vee b) \wedge c$$
$$\{a = false, b = true, c = true\}$$

- Maple has had a Boolean logic package since at least Maple V but the satisfiability-checking command was slow and only intended to be used on small examples.
- In the meantime there has been a lot of progress in SAT solving techniques—to the point that in many domains it is commonplace for people to translate their problem into SAT just to be able to use a SAT solver on it.
- In Maple 2016, Stephen Forrest updated the Satisfy command to use MiniSat, a free SAT solver that hasn't been updated since 2010 but forms the basis for many current state-of-the-art SAT solvers.
- In Maple 2018, I updated the Satisfy command to use MapleSAT, an improved version of MiniSat developed by Vijay Ganesh's research group at Waterloo that won awards in 2016 and 2017.

Graph colouring

- The ChromaticNumber command of the GraphTheory package returns

the minimal number of colours necessary to colour a graph such that no two adjacent vertices share the same colour.



- In Maple 2018, I added a *sat* method which solves the problem by translating it into Boolean logic and calling MapleSAT.
- This method solves some benchmarks in seconds that Maple 2017 couldn't solve in an hour.

```
G := :-Import("example/DSJC125.1.s6", base = datadir);
CodeTools:-Usage(GraphTheory:-ChromaticNumber(G));
```

G := Graph 1: an undirected unweighted graph with 125 vertices and 736 edge(s)

memory used=283.87KiB, alloc change=8.75MiB, cpu time=233.00ms, real time=6.37s, gc time=151.28ms

5

Problem: Overhead

- Profiling revealed that MapleSAT was much faster than Maple's legacy solver but there was a significant amount of overhead in the Maple ↔ MapleSAT link.
- The link was not written for performance: it was based on passing a string to MiniSat.
- In one example with ~37,000 clauses, translating the formula to a string took **1.85s** and finding a satisfying assignment took **0.007s**.
- Not acceptable especially if you want to call the solver many times.

Improved MapleSAT link

- Instead of passing a string we pass a Maple object and have MapleSAT internally convert it into the format it uses.
- With the new link the formula from above is passed to MapleSAT and solved in **0.03s**.

Improved Tseitin

- MapleSAT requires formulas to be in conjunctive normal form. This conversion is done with Maple's Tseitin function.
- Previously Tseitin would always first convert its input into negation normal form, even if it was unnecessary. A check has been added so that Tseitin will check if a formula is in CNF before converting to negation normal form.
- Calling Tseitin on the formula from above previously took about **0.7s**, it now takes about **0.1s**.

Maximum clique

- I added a *sat* method to the MaximumClique command of the GraphTheory package that finds a maximum clique using a SAT solver.
- I also added a *hybrid* method that uses the Grid package to run both the default method and the SAT method in parallel and returns the first result.

Timings

- Timings comparing the hybrid, SAT, and default methods with a timeout of 200 seconds:

BENCHMARK	HYBRID	SAT	DEFAULT

c-fat500-1	2.519	6.202	**0.121**
keller4	9.059	**7.643**	TIMEOUT
p_hat500-1	33.122	TIMEOUT	**31.566**
MANN_a9	1.539	**0.157**	186.845
brock200_2	21.13	23.35	**19.388**
c-fat200-2	1.573	2.077	**0.143**
c-fat200-5	9.559	**7.785**	17.913
johnson8-2-4	1.076	0.06	**0.011**
c-fat500-10	122.164	**114.125**	TIMEOUT
johnson8-4-4	1.556	**0.241**	6.029
c-fat200-1	1.536	0.725	**0.053**
hamming6-4	1.13	0.068	**0.047**
hamming8-2	54.097	**49.863**	TIMEOUT
johnson16-2-4	7.092	**5.575**	TIMEOUT
hamming8-4	9.584	**7.81**	TIMEOUT
p_hat300-1	3.568	11.891	**2.351**
c-fat500-5	38.588	**35.405**	117.546
p_hat700-1	163.249	TIMEOUT	**151.535**
c-fat500-2	2.077	8.936	**0.777**
hamming6-2	2.054	**0.613**	40.838

- The SAT method was the fastest for half of the solved benchmarks and solved 18/80 benchmarks compared to the default method's 15/80 benchmarks.
- The hybrid method solved 20/80 benchmarks but was usually the second fastest method.

Example worksheets

- In addition to the clique-finding example worksheet, I also wrote worksheets for

* The n-queens problem
* Solving the world's hardest Sudoku
* Solving the Einstein riddle
* Solving the 8-puzzle
* An interactive Sudoku game