SAT solvers and Applications to Combinatorics

Introduction - Boolean formulas

- A Boolean formula is an expression involving Boolean variables and the three operations $\wedge$ 'and', $\vee$ 'or' and $\neg$ for negation.

$$\phi = (x \vee y) \wedge (z \vee \neg y)$$

- **Definition:** A Boolean formula is *satisfiable* if and only if there exist at least one variable assignment to the variables to the formula which evaluates it to True.
- ϕ is satisfiable with assignment x is true, y is false z is true.

SAT and CNF Encodings

- The Boolean Satisfiability problem(SAT) is to test whether a given Boolean formula is satisfiable.
- In order to uniform the formatting of SAT problems and for the easy of storing, usually the Boolean formula is transformed into conjunctive normal form(CNF).
- CNF is a conjunction of clauses $\bigwedge_i c_i$, each clause c_i being a disjunction of literals $\bigvee_i l_i$ and each literal l_i being either a Boolean variable v or its negation $\neg v$.

SAT solver

- There are many SAT solvers out there, with varying efficiency. The state-of-the-art solvers include Satch and MiniSat and others.
- Most of the SAT solvers is based in the Davis-Putnam-Logemann-Loveland(DPLL) algorithm.
- Satch in action
- DPLL algorithm high-level example

Combinatorics Example

- **Definition:** MinColoring problem states that, given a graph G, find the minimum number of colors which G's vertices can be colored.
- Given a graph $G = (V, E)$ and an integer k, is the graph k-colorable?
- Encode the vertex coloring:

$$\phi_v = \bigwedge_{v_i \in V} (v_i^1 \vee v_i^2 \vee \ldots v_i^k) \wedge \neg(v_i^1 \wedge v_i^2 \wedge \ldots v_i^k)$$

- Encode the edges:

$$\phi_E = \bigwedge_{(v_i, v_j) \in E} (\neg(v_i^1 \wedge v_j^1) \wedge \neg(v_i^2 \wedge v_j^2) \wedge \ldots \neg(v_i^k \wedge v_j^k))$$

- The resulting formula for SAT solver is

$$\psi = \phi_v \wedge \phi_E$$

CNF file format

- In order for a SAT solver to take in the input, the CNF form also must be formatted in a particular way.
- The CNF file needs to have a parameter line starting with 'p', followed with 'cnf' indicating its cnf form. Next, followed with 2 numbers representing the number of literals and clauses respectively.
- Following the parameter line are the clauses.
- Each clause(line) must end with a '0'.
- Variables must be non-zero numbers, negated literals will have a minus sign.
- Comments start with a 'c' in the beginning.
- File must end with a newline character.

CNF file example

```
p cnf 8 16
1 2 0
3 4 0
5 6 0
7 8 0
-1 -2 0
-3 -4 0
-5 -6 0
-7 -8 0
-1 -3 0
-2 -4 0
-3 -5 0
-4 -6 0
-5 -7 0
-6 -8 0
-1 -7 0
-2 -8 0
```

DPLL

- Within the SAT solver, what is the algorithm?
- DPLL consists of 3 main actions.
  - Decision - randomly assign a literal true or false
  - Unit propagation - must assign a unit literal with a value
  - Backtracking - backtracking to a previous decision at conflicts(contradiction)
- A unit clause is a clause where a single literal in it is unassigned and the rest are false. That unassigned literal is called unit literal.
- The algorithm executes in the order of: check for backtracking $\rightarrow$ check the need for unit propagation $\rightarrow$ make a decision.
- Let's see a quick example to solidify understanding.

DPLL example

$(x_1 \vee x_2)$
$(x_1 \vee x_2 \vee x_8)$
$(\neg x_2 \vee \neg x_3 \vee x_4)$
$(\neg x_4 \vee x_5 \vee x_7)$
$(\neg x_4 \vee x_6 \vee x_8)$
$(\neg x_5 \vee \neg x_6)$
$(x_7 \vee \neg x_8)$
$(x_7 \vee x_9 \vee x_{10})$

- Begin with unit propagation.
- Randomly make a decision, $x_7 = 0$?

DPLL Optimization

- The number of clauses grows very fast. For K_6, there are $\binom{6}{2} = 15$ edges. Thus, $2^{15} = 32,768$ different way of coloring the graph.
- Optimization for DPLL, **clause learning**.
- Quicker unit propagation with Two-Watch-Literal invariant.
- Weighted decision instead of random.
- Parallelize the problem with clever encoding.
- Parallel Multithreaded SAT solver.

Application to Ramsey Number

- The area that Jack and I will be focusing on is Ramsey Numbers.
- Using efficient implementation of SAT solvers to potentially improve the bounds on some Ramsey numbers.