Satisfiability Problems

Patrick Moore

Australian Mathematics Trust

April 9, 2022

A refresher on boolean logic

- Literals
  - Symbols (e.g. x_1) which are either *true* or *false*
- Disjunction
  - Equivalent to the OR operator, denoted by the symbol $\vee$
- Conjunction
  - Equivalent to the AND operator, denoted by the symbol $\wedge$
- Exclusive-Or
  - Equivalent to the XOR operator, denoted by the symbol $\oplus$
- Negation
  - Equivalent to the NOT operator denoted by the symbol $\neg$
- Boolean Formula
  - A combination of literals, disjunction, conjunction and negation to create a formula which can evaluate to either *true* or *false* (e.g. $(x_1 \vee x_2) \wedge (x_3)$).

Conjunctive Normal Form

A boolean formula is in Conjunctive Normal Form (CNF) if it is the conjunction (AND, $\wedge$) of a list of clauses, each of which is a disjunction (OR, $\vee$) of literals (symbols, either *true* or *false*).

e.g. $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3) \wedge (\neg x2)$
$(x_1 \text{ OR } x_2 \text{ OR } \neg x_3) \text{ AND } (\neg x_1 \text{ OR } x_3) \text{ AND } (\neg x2)$

SAT

Definition

There are N literals, $x_1 \dots x_N$.

We are given a boolean formula in CNF, with the literals $x_1 \dots x_N$.

The formula is **satisfiable** if there is some assignment of $x_1 \dots x_N$ to *true* or *false* such that the boolean formula evaluates to *true*.

The problem is to decide whether or not such an assignment exists, and to output the assignment if it exists.

SAT Example

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3) \wedge (\neg x2)$$
$$(x_1 \text{ OR } x_2 \text{ OR } \neg x_3) \text{ AND } (\neg x_1 \text{ OR } x_3) \text{ AND } (\neg x2)$$

Solving SAT

There is no known algorithm to solve SAT in polynomial time.

Solving SAT in polynomial time has been shown to be equivalent to to solving P = NP.

However, there are many heuristic algorithms to solve SAT fairly quickly and consistently with reasonable amounts of clauses and symbols.

We will be focussing on a couple variations and sub-problems of SAT.

Solving SAT

SAT can be solved in $O(2^N \times M)$, where N is the number of literals and M is the length of the formula in CNF.

Definition of the 3-SAT Problem

- This is a slight variation on the general SAT problem.
  - Setup is the same, but the number of literals in each clause is at most 3.

Reducing 3-SAT to SAT

- Given M clauses with at most 3 literals per clause, give a single boolean formula in CNF that is equisatisfiable to the original clauses.

Reducing 3-SAT to SAT

Reducing SAT to 3-SAT

Given a boolean formula in CNF, provide a set of clauses, each with at most 3 literals, that is equisatisfiable (not necessarily logically equivalent) to the original formula.

Reducing SAT to 3-SAT

2-Satisfiability

- This is very similar to 3-SAT, but now each clause is limited to at most 2 literals (e.g. $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$).

Reducing 2-SAT to an implication graph

$(x_1 \vee x_2)$

- $\neg x_1 \implies x_2$
- $\neg x_2 \implies x_1$

$(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee \neg x_1)$

Solving 2-SAT with Strongly Connected Components (SCC)

- Create the implication graph from the provided clauses.
- Create the SCCs of the implication graph.
- The formula is satisfiable iff all x_i and $\neg x_i$ are in different SCCs.

Solving 2-SAT with Strongly Connected Components (SCC)

Generating an assignment

- Get a topological ordering of the nodes in the condensation graph (Use Kosaraju's or Tarjan's or Josh's).
- Traverse the SCC graph in reverse topological order
  - if any of the elements in the SCC are already set, set them to the same state.
  - otherwise, greedily set the SCC to true.

Generating an assignment

Variations on 2-SAT

- Exercises - Create clauses that perform the following things
  - Force x_1 to be true
  - Force exactly one of x_1 or x_2 to be true
  - Force x_1 and x_2 to have the same value

XOR-SAT

In XOR-SAT, literals are combined with XORs instead of ORs. (e.g. $(x_1 \oplus x_2) \wedge (x_2 \oplus x_3) \wedge (\neg x_3 \oplus x_4)$). We will be focusing on XOR-2-SAT, where clauses have at most 2 literals.

XOR-2-SAT implication graphs

Since the state of one literal in XOR-2-SAT uniquely determines the state of the other, the implication graph of each clause in XOR-2-SAT is much stronger than in regular 2-SAT.
e.g. $x_1 \oplus x_2$

XOR-2-SAT: King Arthur II

The king is holding a gathering of N knights in two rooms. You are given a pairwise list of enemies between the knights. Determine if it is possible to allocate the knights to rooms so that no two enemies are in the same room.

XOR-2-SAT variations

- Exercises - Create clauses/graphs that perform the following things
  - Force two literals to be different
  - Force two literals to be the same
    - Using 1 clause
    - Using 2 clauses without additional negation
  - Force a literal to be true/false

Problems

- XOR-SAT
  - King Arthur II (Easy)
  - Detective (Hard)
  - ICPC Brazil 2018 - Modifying SAT (OOS)
- 2-SAT
  - Black Mountain (Medium)
  - Table Colouring (Hard)
- R-SAT
  - If you run out of problems and want a very interesting variation of SAT

Additional Space

Additional Space

Additional Space