$2^{\text{To be, or not to be?}}$

A look at boolean satisfiability

Davin Choo

DSO National Laboratories
CFL1

2^{nd} February 2017

Outline

1. Overview
2. Definitions
3. State of the art
4. Going forward

Coffee talk

Friend	What are you doing at work?	
Me	Improving SAT solvers	
Friend	Wah, so can get perfect score on the test?	✗
Friend	Wah, so if you find an efficient method to solve all cases, you can get $1 million and eternal glory?	✓

Friend	How are you doing it?
Me	By improving heuristics in existing solvers via AI/ML techniques, and coming up with new solving methods

Friend	Why are you looking at SAT solving?
Me	If I tell you, I would have to kill you[1]

[1] We all secretly like to say that for fun, right? :p

What is SAT?

- A boolean SATisfiability problem refers to determining whether there exists a *satisfying assignment* to a given boolean formula F
- Example (Pigeonhole problem): 2 pigeons, 1 hole

$$\begin{aligned} x_1, x_2 &\in \{0,1\} = \{False, True\} \\ x_1 &: \text{Pigeon 1 in the hole} \\ x_2 &: \text{Pigeon 2 in the hole} \\ \overline{x_1 \wedge x_2} &: \text{Cannot have 2 pigeons in same hole} \\ F &: (x_1) \wedge (x_2) \wedge (\overline{x_1 \wedge x_2}) \end{aligned}$$

Question: What to assign to x_1 and x_2 such that F holds?

This problem is UNSATisfiable.

Pigeonhole is SAT $\iff$ no. of pigeons $\leq$ no. of holes

A million dollar question: P = NP?

Millennium Prize Problems

https://en.wikipedia.org/wiki/Millennium_Prize_Problems

"The Millennium Prize Problems are seven problems in mathematics that were stated by the Clay Mathematics Institute in 2000. The problems are Birch and Swinnerton-Dyer conjecture, Hodge conjecture, Navier-Stokes existence and smoothness, P versus NP problem, ~~Poincaré conjecture~~, Riemann hypothesis, and Yang-Mills existence and mass gap. A correct solution to any of the problems results in a US $1 million prize being awarded by the institute to the discoverer(s)."

Rally to Restore Sanity and/or Fear (October 30, 2010)

Image source: https://emeryblogger.com/2010/

Why is SAT hard?

Informally,

P Problems that can be *solved efficiently*

NP Problems whose solutions can be *verified efficiently*

Examples of interesting P problems:

- Inverting a matrix
- Linear Programming (LP) problems

Examples of interesting NP problems:

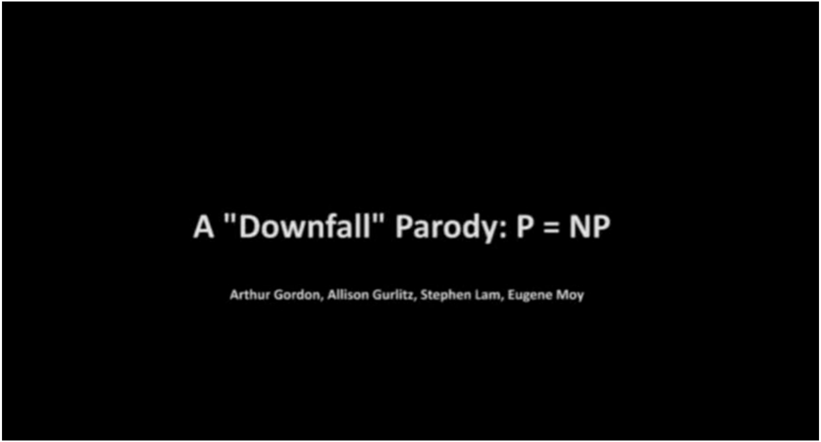
- Integer Linear Programming (ILP) problems
- Travelling salesman problem
- SAT problems

NP-complete

- Integer factorization[2]
  Many cryptographic primitives rely on this problem being hard

[2] As of 2016, not known if it is NP-complete

Implications of "P = NP?" in research

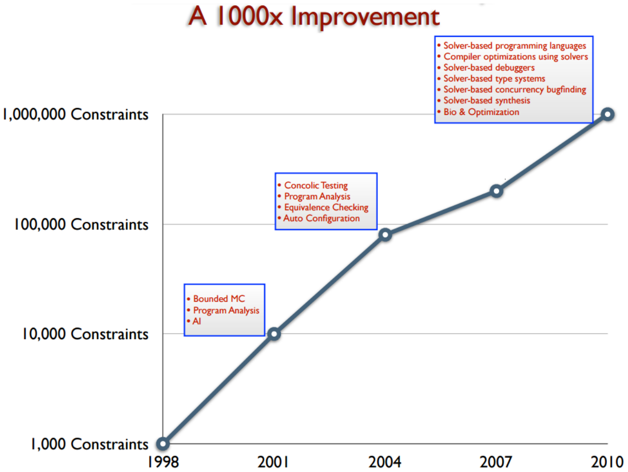


Source: https://www.youtube.com/watch?v=GSIodz9GWxc

Why is SAT interesting?

- Theoretical community: "P = NP?"
- Boolean formulas are *very* expressive.
  - Many interesting problems are NP-hard
    e.g. Travelling Salesman Problem, Bin packing
  - Any NP-hard problem can be mapped to SAT
- Beyond Boolean: Satisfiability Modulo Theories (SMT) solvers
- Industry: Solvers as a practical tool (e.g. Microsoft's Z3)
  - Model checking: Encode program semantics
  - Theorem provers: Prove theorems with set of encoded axioms
  - Efficiently solve game theoretical problems:
    Solving stable matching problem with couples (IJCAI 2015)
  - Using SAT solvers as black-box to perform efficient
    constrained sampling and counting (AAAI 2016)

Why is SAT interesting?



Source: `https://ece.uwaterloo.ca/~vganesh/talks/SATSMT-Dagstuhl-Aug8-12-2011-part1.pdf`

Personal take

- Give up on solving *all* instances efficiently
- Hope: Solve *specific classes* of instances efficiently
  e.g. Can we build an efficient pigeonhole solver?
- Even speeding it up in constant time is good practical progress (10x speed up: 10 years $\rightarrow$ 1 year)

Propositional logic

- Set of variables: $X = \{x_1, x_2, ..., x_n\}$
- Set of literals: $L = \{x_1, \overline{x_1}, x_2, \overline{x_2}, ..., x_n, \overline{x_n}\}$, $l_i = x_i$ or $\overline{x_i}$
- Common boolean operators $\circ$: $\wedge$, $\vee$, $\Rightarrow$, $\oplus$ and $\equiv$
- Let $\mathcal{F}$ be the set of well-formed formulas. $F \in \mathcal{F}$ if (i) $F = x_i$, (ii) $F = \overline{x_i}$, or (iii) $F = F_1 \circ F_2$ (for $F_1, F_2 \in \mathcal{F}$)
- Assignment $\alpha : X \to \{0, 1\}$.

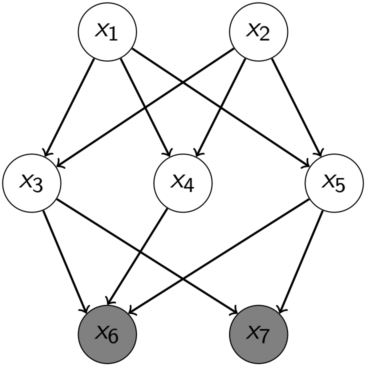
  For $l_i \in L$, $\alpha(l_i) = \begin{cases} \alpha(x_i) & \text{if } l_i = x_i \\ 1 - \alpha(x_i) & \text{if } l_i = \overline{x_i} \end{cases}$
- Valuation function under assignment α, $\nu_\alpha : \mathcal{F} \to \{0, 1\}$
- α is a satisfying assignment for F if $\nu_\alpha(F) = 1$

Conjunctive Normal Form (CNF)

- F is in CNF if:
  - $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$
  - $C_i = l_{i,1} \vee \ldots \vee l_{i,k_i}$
  - Informally, F is in the form of "ANDs of ORs"
  - C_i is also called a clause
  - $\nu_\alpha(C_i) = 1$, if $\exists l_{i,k} \in C_i$ such that $\alpha(l_{i,k}) = 1$
  - $\nu_\alpha(F) = 1$, if $\forall C_i \in F$ such that $\nu_\alpha(C_i) = 1$
- Fact: All boolean formulas in prop. logic can be put into CNF e.g. $(x_1) \wedge (x_2) \wedge (\overline{x_1 \wedge x_2})$ becomes $(x_1) \wedge (x_2) \wedge (\overline{x_1} \vee \overline{x_2})$
- DIMACS: A specification of storing CNF on computers as inputs to SAT solvers

How would *you* solve this?

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \in \{0, 1\}$
Unknown: x_1, x_2, x_3, x_4, x_5
Observed: x_6, x_7



<u>Model</u>

x_1 : Real unknown
x_2 : Real unknown
$x_3 : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$x_5 : x_1 \oplus x_2$
$x_6 : \neg x_3 \vee \neg x_4 \vee \neg x_5$
$x_7 : x_3 \wedge x_5$

Method 1 : Truth table enumeration

x_1 : Real unknown
x_2 : Real unknown
$x_3 : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$x_5 : x_1 \oplus x_2$
$x_6 : \neg x_3 \vee \neg x_4 \vee \neg x_5$
$x_7 : x_3 \wedge x_5$

x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	0	0	0	0	1	0
0	1	1	0	1	1	1
1	0	1	1	1	0	1
1	1	1	0	0	1	0

- Given x_6 and x_7, read off solution in $\mathcal{O}(1)$
  $x_6 = 0, x_7 = 0 \rightarrow \nu(F) = 0$ for every possible α — (UNSAT)
  $x_6 = 0, x_7 = 1 \rightarrow \alpha(x_1) = 1, \alpha(x_2) = 0$
  $x_6 = 1, x_7 = 0 \rightarrow \alpha(x_1) = \alpha(x_2)$ ——— (2 solutions)
  $x_6 = 1, x_7 = 1 \rightarrow \alpha(x_1) = 1, \alpha(x_2) = 0$
- Problem: Memory requirement scales in $\mathcal{O}(2^{nu})$
  n = number of variables
  u = number of real unknowns

Method 2 : Tree search methods

- Breadth first search
  - Worst case leads to explosion in memory requirements
- Depth first search
  - Transfers memory overhead to time overhead
  - If we are lucky, we guess it in 1 shot!
- Improvement: Use tree pruning methods while searching
- Problem: How to pick branching variable?
- Problem: How to pick polarity of variable?
- Problem: If UNSAT, may have to visit every branch

Approaches

Problem definition Given a formula F, find a satisfying assignment α^* (SAT) or declare that F is unsatisfiable (UNSAT).

Incomplete solvers

- Return a correct solution if found
- Declare no solution with some level of confidence
- Usually some form of local search
- Performs well on problems with several solutions

Complete solvers (*)

- Return a correct solution if found
- Declare no solution with absolute certainty
- Exhaustive search techniques that work even when there is exactly 1 solution out of 2^n possibilities

Davis-Putnam-Logemann-Loveland (DPLL)

Depth first search with chronological backtrack

Algorithm 1 $\mathrm{DPLL}(\mathrm{F})$

if $F = \varnothing$ **then return** $\varnothing$
else if $\varnothing \in F$ **then return** UNSAT
else
  Pick an unassigned variable x ▷ e.g. Pick smallest x_i
  if $\mathrm{DPLL}(F[x/1]) \neq$ UNSAT **then** ▷ Try $x = \mathit{True}$
    return $\mathrm{DPLL}(F[x/1]) \cup \{x/1\}$
  else if $\mathrm{DPLL}(F[x/0]) \neq$ UNSAT **then** ▷ Try $x = \mathit{False}$
    return $\mathrm{DPLL}(F[x/0]) \cup \{x/0\}$
  else ▷ Both assignments to x failed. No solution
    return UNSAT
  end if
end if

What goes on in $F[x/1]$?

Substitution x replaced by *True*, $\overline{x}$ by *False*.

Can ignore all clauses in $F[x/1]$ that had x previously

Unit propagation If a clause becomes unit, that remaining literal is immediately assigned and F is further simplified

$$F : \text{Clause } C_j = \overline{x_1} \vee x_2 \vee 0 \vee \ldots \vee 0$$
$$F[x/1] : \text{Clause } C_j = 0 \vee x_2 \vee 0 \ldots \vee 0$$
Have to assign $x_2 = \textit{True}$ to satisfy $F[x/1]$

Pure literal elimination Assign any pure literals

A variable x_i is pure if only x_i or $\overline{x_i}$ appears in F.
If F contains both x_i and $\overline{x_i}$, x_i is not pure.

DPLL Example

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \in \{0, 1\}$
Unknown: x_1, x_2, x_3, x_4, x_5
Observed: x_6, x_7

<u>Model</u>

x_1 : Real unknown
x_2 : Real unknown
$x_3 : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$x_5 : x_1 \oplus x_2$
$x_6 : \neg x_3 \vee \neg x_4 \vee \neg x_5$
$x_7 : x_3 \wedge x_5$

Suppose:
$x_6 = \mathit{True}$
$x_7 = \mathit{True}$

<u>Model</u>

x_1 : Real unknown
x_2 : Real unknown
$x_3 : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$x_5 : (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$
$T : \neg x_3 \vee \neg x_4 \vee \neg x_5$
$T : x_3 \wedge x_5$

$x_3 = ?$
$x_4 = ?$
$x_5 = ?$
$x_6 = \mathit{True}$
$x_7 = \mathit{True}$

DPLL Example

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \in \{0, 1\}$
Unknown: x_1, x_2, x_3, x_4, x_5
Observed: x_6, x_7

Model
x_1 : Real unknown
x_2 : Real unknown
$x_3 : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$x_5 : x_1 \oplus x_2$
$x_6 : \neg x_3 \vee \neg x_4 \vee \neg x_5$
$x_7 : x_3 \wedge x_5$

Suppose:
$x_6 = True$
$x_7 = True$

Model
x_1 : Real unknown
x_2 : Real unknown
$T : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$T : (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$
$T : F \vee \neg x_4 \vee F$
$T : T \wedge T$

$x_3 = True$
$x_4 = ?$
$x_5 = True$
$x_6 = True$
$x_7 = True$

DPLL Example

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \in \{0, 1\}$
Unknown: x_1, x_2, x_3, x_4, x_5
Observed: x_6, x_7

Model

x_1 : Real unknown
x_2 : Real unknown
$x_3 : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$x_5 : x_1 \oplus x_2$
$x_6 : \neg x_3 \vee \neg x_4 \vee \neg x_5$
$x_7 : x_3 \wedge x_5$

Suppose:
$x_6 = True$
$x_7 = True$

Model

x_1 : Real unknown
x_2 : Real unknown
$T : x_1 \vee x_2$
$F : x_1 \wedge \neg x_2$
$T : (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$
$T : F \vee T \vee F$
$T : T \wedge T$

$x_3 = True$
$x_4 = False$
$x_5 = True$
$x_6 = True$
$x_7 = True$

DPLL Example

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \in \{0, 1\}$
Unknown: x_1, x_2, x_3, x_4, x_5
Observed: x_6, x_7

<u>Model</u>

x_1 : Real unknown
x_2 : Real unknown
$x_3 : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$x_5 : x_1 \oplus x_2$
$x_6 : \neg x_3 \vee \neg x_4 \vee \neg x_5$
$x_7 : x_3 \wedge x_5$

Suppose:
$x_6 = \mathit{True}$
$x_7 = \mathit{True}$

After repeatedly using unit propagation and simplification:
$\alpha(x_3) = 1$
$\alpha(x_4) = 0$
$\alpha(x_5) = 1$
$\alpha(x_6) = 1$
$\alpha(x_7) = 1$

CNF reduces to:
$(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

Perform branching to solve:
$\alpha(x_1) = 0$
$\alpha(x_2) = 1$

But wait, there's more!

Implementations: Pick literals → Pick variable, then polarity

Variable selection Heuristics (e.g. VSIDS, CHB, LRB)

Polarity selection Heuristics (e.g. BOHM, MOM, Jeroslow-Wang), Polarity preservation

Efficient data structures SATO's Head/Tail, Watched literals

Learning while searching Conflict Driven Clause Learning (CDCL)

Simplifying the problem Clause subsumption, Bounded variable addition (BVA), Bounded variable elimination (BVE)

Native XOR Simplex, Gaussian elimination
e.g. *CryptoMiniSAT*, Tero's PhD, NTU's *SimpSat*

Random restarts To avoid getting stuck locally

Incremental SAT Allows injection of new constraints on the fly

Some details on the 4 most important add-ons

1. Conflict Driven Clause Learning (CDCL)
    - Perform resolution on conflict clause in implication graph
2. Variable selection heuristics
    - Branch on $\text{argmax}_{\text{free } v} h(v)$, $h(v) =$ heuristic of variable v.
    - VSIDS[3]: Increase $h(v)$ when we see variable v
    - Conflict History Based: Skew $h(v)$ to encourage conflicts
    - Learning Rate Based: Skew $h(v)$ to encourage learning
3. Polarity preservation
    - Avoid repeating subtree search when backtracking
    - Assign variable with the polarity it was previously, if possible
    - Can be implemented using a simple size n boolean array
4. Watched literals data structure
    - For every clause, track only 2 of the literals
    - Allows fast checking when propagating and $\mathcal{O}(1)$ backtracking

---

[3] Variable State Independent Decaying Sum

Notable implementations and papers

GRASP Conflict Driven Clause Learning

Chaff VSIDS heuristic, Watched literals data structure

Paper on polarity preservation
Knot Pipatsrisawat and Adnan Darwiche.
A Lightweight Component Caching Scheme for Satisfiability Solvers

MiniSAT Developed by Niklas En, Niklas Srensson.
Relatively small code base.
Used by various people to try ideas and extend upon

CryptoMiniSAT Personal project of Mate Soos.
Incorporates XOR natively via Gaussian elimination.
Actively tries to incorporate new research ideas

Stalmarck's method

- Background
  - Invented and patented (expired in 2011) by Gunnar Stalmarck
  - Key technology behind Prover Technology
  - Few records in literature as no one worked on it
- Tautology checker
  - Given a propositional formula F, able to declare if it is UNSAT
  - Can be modified to return a counter-model if F is SAT
  - Usage: Given F, check if $\neg F$ is UNSAT

Triples representation

- Instead of CNF, represent problem as *triples* $\langle$ parent ID : left $\circ$ right $\rangle$
- The 2(n+i) indexing holds only if every clause has 1 binary operator
- Otherwise, chunk it up (Beware of bracket ordering)
- For example: $\overline{x_3} \vee \overline{x_4} \vee \overline{x_5} = (\overline{x_3} \vee \overline{x_4}) \vee \overline{x_5}$
- Initially, every triple represents an eq class of its own
- Merge things that must hold into the *True* eq class

	ID	Semantic meaning
Not *real* triples	0	False
	1	True
	$\vdots$	$\vdots$
	2i	$\overline{x_i}$
	2i+1	x_i
	$\vdots$	$\vdots$
	2(n+i)	$\overline{\text{Clause i}}$
	2(n+i)+1	Clause i
	$\vdots$	$\vdots$
	The rest	Link up variable definitions

Triples representation

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \in \{0, 1\}$
Unknown: x_1, x_2, x_3, x_4, x_5
Observed: x_6, x_7

Model

x_1 : Real unknown
x_2 : Real unknown
$x_3 : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$x_5 : x_1 \oplus x_2$
$x_6 : \neg x_3 \vee \neg x_4 \vee \neg x_5$
$x_7 : x_3 \wedge x_5$

ID	Semantic meaning
0	False
1	True
2i	$\overline{x_i}$
2i+1	x_i
16, **17**	$x_1 \vee x_2$
18, **19**	$x_1 \wedge \neg x_2$
20, **21**	$x_1 \oplus x_2$
22, **23**	$\overline{x_3} \vee \overline{x_4}$
24, **25**	$23 \vee \overline{x_5}$
26, **27**	$x_3 \wedge x_5$
28, **29**	$7 \equiv 17$
30, **31**	$9 \equiv 19$
32, **33**	$11 \equiv 21$
34, **35**	$13 \equiv 25$
36, **37**	$15 \equiv 27$

Merge $\{29, 31, 33, 35, 37\}$ with 1
(Shadow): Merge $\{28, 30, 32, 34, 36\}$ with 0

0-Saturation

- Repeated application of *simple deductive rules* merge equivalence classes till we reach a fixed point
- F is UNSAT if $\mathit{True}(\top, 1)$ and $\mathit{False}(\bot, 0)$ are in same eq class

Here are some simple rules:

$$\langle P : L \wedge R \rangle \qquad \langle P : L \vee R \rangle \qquad \langle P : L \oplus R \rangle \qquad \langle P : L \equiv R \rangle$$

$$\frac{P \equiv \top}{L \equiv \top, R \equiv \top} \qquad \frac{P \equiv \bot}{L \equiv \bot, R \equiv \bot} \qquad \frac{P \equiv \top}{L \equiv \overline{R}} \qquad \frac{P \equiv L}{R \equiv \top}$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$\frac{L \equiv \overline{R}}{P \equiv \bot} \qquad \frac{L \equiv \overline{R}}{P \equiv \top} \qquad \frac{P \equiv \bot}{L \equiv R} \qquad \frac{R \equiv \bot}{P \equiv \overline{L}}$$

Example with 0-Saturation

ID	Semantic meaning
0	False
1	True
2i	$\overline{x_i}$
2i+1	x_i
16, **17**	$x_1 \vee x_2$
18, **19**	$x_1 \wedge \neg x_2$
20, **21**	$x_1 \oplus x_2$
22, **23**	$\overline{x_3} \vee \overline{x_4}$
24, **25**	$23 \vee \overline{x_5}$
26, **27**	$x_3 \wedge x_5$
28, **29**	$7 \equiv 17$
30, **31**	$9 \equiv 19$
32, **33**	$11 \equiv 21$
34, **35**	$13 \equiv 25$
36, **37**	$15 \equiv 27$

Merge $\{29, 31, 33, 35, 37\}$ with 1
Merge $\{28, 30, 32, 34, 36\}$ with 0 (Shadow)

[0, 28, 30, 32, 34, 36]
[1, 29, 31, 33, 35, 37]
[2][3][4][5][6][7]
[8][9][10][11][12][13]
[14][15][16][17][18][19]
[20][21][22][23][24][25]
[26][27]

Example with 0-Saturation

ID	Semantic meaning
0	False
1	True
2i	$\overline{x_i}$
2i+1	x_i
16, **17**	$x_1 \vee x_2$
18, **19**	$x_1 \wedge \neg x_2$
20, **21**	$x_1 \oplus x_2$
22, **23**	$\overline{x_3} \vee \overline{x_4}$
24, **25**	$23 \vee \overline{x_5}$
26, **27**	$x_3 \wedge x_5$
28, **29**	$7 \equiv 17$
30, **31**	$9 \equiv 19$
32, **33**	$11 \equiv 21$
34, **35**	$13 \equiv 25$
36, **37**	$15 \equiv 27$

Rules fired:

1. $36 \equiv 0 \models 15 \equiv 27$
2. $4 \equiv 0 \models 13 \equiv 25$
3. $32 \equiv 0 \models 11 \equiv 21$
4. $30 \equiv 0 \models 9 \equiv 19$
5. $28 \equiv 0 \models 7 \equiv 17$

$$
\begin{gathered}
[0, 28, 30, 32, 34, 36] \\
[1, 29, 31, 33, 35, 37] \\
[6, 16][7, 17] \\
[8, 18][9, 19] \\
[10, 20][11, 21] \\
[12, 24][13, 25] \\
[14, 26][15, 27] \\
[2][3][4][5][22][23]
\end{gathered}
$$

Personally, I call this the *model prior*.

Example with 0-Saturation

ID	Semantic meaning
0	False
1	True
2i	$\overline{x_i}$
2i+1	x_i
16, **17**	$x_1 \vee x_2$
18, **19**	$x_1 \wedge \neg x_2$
20, **21**	$x_1 \oplus x_2$
22, **23**	$\overline{x_3} \vee \overline{x_4}$
24, **25**	$23 \vee \overline{x_5}$
26, **27**	$x_3 \wedge x_5$
28, **29**	$7 \equiv 17$
30, **31**	$9 \equiv 19$
32, **33**	$11 \equiv 21$
34, **35**	$13 \equiv 25$
36, **37**	$15 \equiv 27$

[0, 28, 30, 32, 34, 36]
[1, 29, 31, 33, 35, 37]
[6, 16][7, 17]
[8, 18][9, 19]
[10, 20][11, 21]
[12, 24][13, 25]
[14, 26][15, 27]
[2][3][4][5][22][23]

Suppose we set $x_6 = 0, x_7 = 0$,
Merge $\{12, 14\}$ with 1
Merge $\{13, 15\}$ with 0 (Shadow)

[0, 13, 15, 25, 27, 28, 30, 32, 34, 36]
[1, 12, 14, 24, 26, 29, 31, 33, 35, 37]
[6, 16][7, 17]
[8, 18][9, 19]
[10, 20][11, 21]
[2][3][4][5][22][23]

Example with 0-Saturation

ID	Semantic meaning
0	False
1	True
2i	$\overline{x_i}$
2i+1	x_i
16, **17**	$x_1 \vee x_2$
18, **19**	$x_1 \wedge \neg x_2$
20, **21**	$x_1 \oplus x_2$
22, **23**	$\overline{x_3} \vee \overline{x_4}$
24, **25**	$23 \vee \overline{x_5}$
26, **27**	$x_3 \wedge x_5$
28, **29**	$7 \equiv 17$
30, **31**	$9 \equiv 19$
32, **33**	$11 \equiv 21$
34, **35**	$13 \equiv 25$
36, **37**	$15 \equiv 27$

[0, 13, 15, 25, 27, 28, 30, 32, 34, 36]
[1, 12, 14, 24, 26, 29, 31, 33, 35, 37]
[6, 16][7, 17]
[8, 18][9, 19]
[10, 20][11, 21]
[2][3][4][5][22][23]

Rule fired:

- $(23 \vee \overline{x_5}) \equiv F$,

  so $23 \equiv False$ and $\overline{x_5} \equiv False$
- $25 \equiv 0 \models 23 \equiv 0, 10 \equiv 0$
- Merge $22 \equiv 1, 11 \equiv 1$ (Shadow)

[0, 10, 13, 15, 20, 23, 25, 27, 28, 30, 32, 34, 36]
[1, 11, 12, 14, 21, 22, 24, 26, 29, 31, 33, 35, 37]
[6, 16][7, 17]
[8, 18][9, 19]
[10, 20][11, 21]
[2][3][4][5][22][23]

Example with 0-Saturation

ID	Semantic meaning
0	False
1	True
2i	$\overline{x_i}$
2i+1	x_i
16, **17**	$x_1 \vee x_2$
18, **19**	$x_1 \wedge \neg x_2$
20, **21**	$x_1 \oplus x_2$
22, **23**	$\overline{x_3} \vee \overline{x_4}$
24, **25**	$23 \vee \overline{x_5}$
26, **27**	$x_3 \wedge x_5$
28, **29**	$7 \equiv 17$
30, **31**	$9 \equiv 19$
32, **33**	$11 \equiv 21$
34, **35**	$13 \equiv 25$
36, **37**	$15 \equiv 27$

[0, 10, 13, 15, 20, 23, 25, 27, 28, 30, 32, 34, 36]
[1, 11, 12, 14, 21, 22, 24, 26, 29, 31, 33, 35, 37]
[6, 16][7, 17]
[8, 18][9, 19]
[10, 20][11, 21]
[2][3][4][5][22][23]

Rule fired:

- $(x_3 \wedge x_5) \equiv F$ but $x_5 \equiv True$, so $x_3 \equiv False$
- $27 \equiv 0, 11 \equiv 1 \models 7 \equiv 0$
- Merge $6 \equiv 1$ (Shadow)

[0, 7, 10, 13, 15, 17, 20, 23, 25, 27, 28, 30, 32, 34, 36]
[1, 6, 11, 12, 14, 16, 21, 22, 24, 26, 29, 31, 33, 35, 37]
[8, 18][9, 19]
[10, 20][11, 21]
[2][3][4][5][22][23]

Example with 0-Saturation

ID	Semantic meaning
0	False
1	True
2i	$\overline{x_i}$
2i+1	x_i
16, **17**	$x_1 \vee x_2$
18, **19**	$x_1 \wedge \neg x_2$
20, **21**	$x_1 \oplus x_2$
22, **23**	$\overline{x_3} \vee \overline{x_4}$
24, **25**	$23 \vee \overline{x_5}$
26, **27**	$x_3 \wedge x_5$
28, **29**	$7 \equiv 17$
30, **31**	$9 \equiv 19$
32, **33**	$11 \equiv 21$
34, **35**	$13 \equiv 25$
36, **37**	$15 \equiv 27$

[0, 7, 10, 13, 15, 17, 20, 23, 25, 27, 28, 30, 32, 34, 36]
[1, 6, 11, 12, 14, 16, 21, 22, 24, 26, 29, 31, 33, 35, 37]
[8, 18][9, 19]
[10, 20][11, 21]
[2][3][4][5][22][23]

Rule fired:

- $(\overline{x_3} \vee \overline{x_5}) \equiv F$,

  so $\overline{x_3} \equiv \mathit{False}$ and $\overline{x_5} \equiv \mathit{False}$
- $23 \equiv 0 \models 6 \equiv 0, 8 \equiv 0$
- Merge $7 \equiv 1, 9 \equiv 1$ (Shadow)

[0, 1, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]
[10, 20][11, 21]
[2][3][4][5][22][23]

Contradiction since $\mathit{True}(1) \equiv \mathit{False}(0)$

k-Saturation

- For interesting problems, 0-Saturation is insufficient
- Make assumptions and branch in a Breadth First manner
- k-Saturation
  - Defined recursively in terms of $(k-1)$-Saturation
  - Base case: 0-Saturation = Apply simple rules till fixed point
- Dilemma rule
  - Intersect results from opposing branches in k-Saturation
  - Example:
    Suppose branching $x_1 \equiv \mathit{True}$ yields $x_2 \equiv \mathit{True}$.
    Suppose branching $x_1 \equiv \mathit{False}$ also yields $x_2 \equiv \mathit{True}$.
    Taking the intersection, we yield $x_2 \equiv \mathit{True}$ from here on.

k-Saturation

Note: $\textsc{Saturate}(0, E) = \text{0-Saturate}(E)$

Algorithm 2 $\textsc{Saturate}$(k, E)

$C \leftarrow$ equivalence classes of E ▷ 1 representative per class
for $x, y \in C, x \neq y$ **do**
  $E_{x \equiv y} \leftarrow \textsc{Saturate}(k-1,\ E \text{ with } x \equiv y)$
  $E_{x \equiv \overline{y}} \leftarrow \textsc{Saturate}(k-1,\ E \text{ with } x \equiv \overline{y})$
  if $E_{x \equiv y}$ has Contradiction **then** $E \leftarrow E_{x \equiv \overline{y}}$
  else if $E_{x \equiv \overline{y}}$ has Contradiction **then** $E \leftarrow E_{x \equiv y}$
  else $E \leftarrow E_{x \equiv y} \cap E_{x \equiv \overline{y}}$ ▷ Dilemma rule
  end if
end for
return E

Stalmarck's method

Breadth first search using k-Saturation and dilemma rule
E is a set of equivalent classes of triples converted from $\neg F$

Algorithm 3 STALMARCK(F)

```
k ← 0
loop
    E' ← SATURATE(k, E)
    if E' is contradictory then
        return F is a tautology
    else if E' has only 2 eq classes (True and False) then
        return Assignment based on E'
    end if
    k ← k + 1                                  ▷ Increment k
end loop
```

Example with 0-Saturation

ID	Semantic meaning
0	False
1	True
2i	$\overline{x_i}$
2i+1	x_i
16, **17**	$x_1 \vee x_2$
18, **19**	$x_1 \wedge \neg x_2$
20, **21**	$x_1 \oplus x_2$
22, **23**	$\overline{x_3} \vee \overline{x_4}$
24, **25**	$23 \vee \overline{x_5}$
26, **27**	$x_3 \wedge x_5$
28, **29**	$7 \equiv 17$
30, **31**	$9 \equiv 19$
32, **33**	$11 \equiv 21$
34, **35**	$13 \equiv 25$
36, **37**	$15 \equiv 27$

[0, 28, 30, 32, 34, 36]
[1, 29, 31, 33, 35, 37]
[6, 16][7, 17]
[8, 18][9, 19]
[10, 20][11, 21]
[12, 24][13, 25]
[14, 26][15, 27]
[2][3][4][5][22][23]

Suppose we set $x_6 = 1, x_7 = 0$,
Merge $\{13, 14\}$ with 1
Merge $\{12, 15\}$ with 0 (Shadow)

[0, 12, 15, 24, 27, 28, 30, 32, 34, 36]
[1, 13, 14, 25, 26, 29, 31, 33, 35, 37]
[6, 16][7, 17]
[8, 18][9, 19]
[10, 20][11, 21]
[2][3][4][5][22][23]

No applicable simple rule!

Example with 0-Saturation

ID	Semantic meaning
0	False
1	True
2i	$\overline{x_i}$
2i+1	x_i
16, **17**	$x_1 \vee x_2$
18, **19**	$x_1 \wedge \neg x_2$
20, **21**	$x_1 \oplus x_2$
22, **23**	$\overline{x_3} \vee \overline{x_4}$
24, **25**	$23 \vee \overline{x_5}$
26, **27**	$x_3 \wedge x_5$
28, **29**	$7 \equiv 17$
30, **31**	$9 \equiv 19$
32, **33**	$11 \equiv 21$
34, **35**	$13 \equiv 25$
36, **37**	$15 \equiv 27$

Guess $x_3 \equiv True$,
Merge 7 with 1
Merge 6 with 0 (Shadow)

[0, 6, 28, 30, 32, 34, 36]
[1, 7, 29, 31, 33, 35, 37]
[8, 18][9, 19]
[10, 20][11, 21]
[12, 24][13, 25]
[14, 26][15, 27]
[2][3][4][5][22][23]

After 0-saturation:

[0, 2, 4, 6, 9, 11, 12, 15, 16, 19, 21, 22, 24, 27, 28, 30, 32, 34, 36]
[1, 3, 5, 7, 8, 10, 13, 14, 17, 18, 20, 23, 25, 26, 29, 31, 33, 35, 37]

Satisfying assignment:
$x_1 = True, x_2 = True$

Notable implementations and papers

Stalmarck implementation 404 Not Found

Tutorial Mary Sheeran, Gunnar Stalmarck. *A Tutorial on Stalmarck's Proof Procedure for Propositional Logic*

Thesis Jakob Nordström. *Stalmarcks Method versus Resolution: A Comparative Theoretical Study*

Generalization Aditya Thakur, Thomas Reps. *A Generalization of Ståalmarcks Method*

Summary

Today:

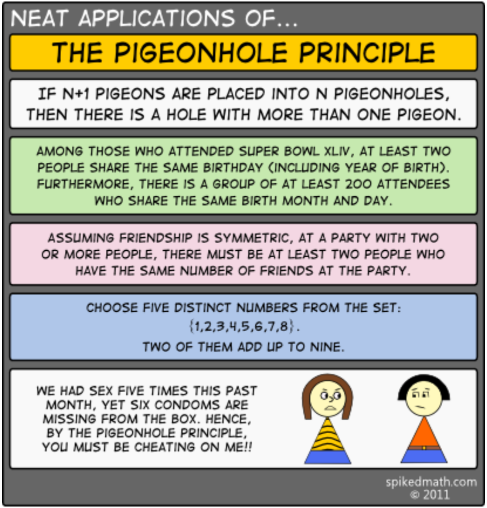
- Introduction to SAT
  - What is SAT?
  - Why is SAT hard?
  - Why is SAT interesting?
- State-of-the-art SAT solvers
  - DPLL + CDCL (Depth first)
  - Stalmarck (Breadth first)
- Sneak peek at our work in this area

What's next?

- Specialised solvers for each problem class
- Combination of depth-first and breadth-first
- Parallelism

Let me know if you have any ideas/questions! ☺

For fun



Moral of the story: Don't cheat on someone who knows counting!

Source: `http://spikedmath.com/390.html`

DIMACS

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \in \{0, 1\}$
Unknown: x_1, x_2, x_3, x_4, x_5
Observed: x_6, x_7

<u>Model</u>

x_1 : Real unknown
x_2 : Real unknown
$x_3 : x_1 \vee x_2$
$x_4 : x_1 \wedge \neg x_2$
$x_5 : x_1 \oplus x_2$
$x_6 : \neg x_3 \vee \neg x_4 \vee \neg x_5$
$x_7 : x_3 \wedge x_5$

```
c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
-6 -3 -4 -5 0
6 3 0
6 4 0
6 5 0
c x7 := x3 and x5
7 -3 -5 0
-7 3 0
-7 5 0
c Observations
6 0
7 0
```

$x_3 : x_1 \vee x_2$

$x_4 : x_1 \wedge \overline{x_2}$

$x_5 : x_1 \oplus x_2$

$x_6 : \overline{x_3} \vee \overline{x_4} \vee \overline{x_5}$

$x_7 : x_3 \wedge x_5$

Observations

DPLL on DIMACS

```
c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
-6 -3 -4 -5 0
6 3 0
6 4 0
6 5 0
c x7 := x3 and x5
7 -3 -5 0
-7 3 0
-7 5 0
c Observations
6 0
7 0
```

```
c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
F -3 -4 -5 0
T 3 0
T 4 0
T 5 0
c x7 := x3 and x5
T -3 -5 0
F 3 0
F 5 0
c Observations
6 0
7 0
```

DPLL on DIMACS

c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
-6 -3 -4 -5 0
6 3 0
6 4 0
6 5 0
c x7 := x3 and x5
7 -3 -5 0
-7 3 0
-7 5 0
c Observations
6 0
7 0

c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
F -3 -4 -5 0
~~T 3 0~~
~~T 4 0~~
~~T 5 0~~
c x7 := x3 and x5
~~T -3 -5 0~~
F 3 0
F 5 0
c Observations
6 0
7 0

DPLL on DIMACS

c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
-6 -3 -4 -5 0
6 3 0
6 4 0
6 5 0
c x7 := x3 and x5
7 -3 -5 0
-7 3 0
-7 5 0
c Observations
6 0
7 0

c DIMACS
p cnf 7 19
c x3 := x1 or x2
F 1 2 0
T -1 0
T -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
F 1 2 0
F -1 -2 0
T -1 2 0
T 1 -2 0
c x6 := -x3 or -x4 or -x5
F -4 F 0
~~T 3 0~~
~~T 4 0~~
~~T 5 0~~
c x7 := x3 and x5
~~T -3 -5 0~~
F 3 0
F 5 0
c Observations
6 0
7 0

DPLL on DIMACS

c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
-6 -3 -4 -5 0
6 3 0
6 4 0
6 5 0
c x7 := x3 and x5
7 -3 -5 0
-7 3 0
-7 5 0
c Observations
6 0
7 0

c DIMACS
p cnf 7 19
c x3 := x1 or x2
F 1 2 0
~~T -1 0~~
~~T -2 0~~
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
F 1 2 0
F -1 -2 0
~~T -1 2 0~~
~~T 1 -2 0~~
c x6 := -x3 or -x4 or -x5
F -4 F 0
~~T 3 0~~
~~T 4 0~~
~~T 5 0~~
c x7 := x3 and x5
~~T -3 -5 0~~
F 3 0
F 5 0
c Observations
6 0
7 0

DPLL on DIMACS

c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
-6 -3 -4 -5 0
6 3 0
6 4 0
6 5 0
c x7 := x3 and x5
7 -3 -5 0
-7 3 0
-7 5 0
c Observations
6 0
7 0

c DIMACS
p cnf 7 19
c x3 := x1 or x2
F 1 2 0
~~T -1 0~~
~~T -2 0~~
c x4 := x1 and -x2
F -1 2 0
T 1 0
T -2 0
c x5 := x1 xor x2
F 1 2 0
F -1 -2 0
~~T -1 2 0~~
~~T 1 -2 0~~
c x6 := -x3 or -x4 or -x5
F -4 F 0
~~T 3 0~~
~~T 4 0~~
~~T 5 0~~
c x7 := x3 and x5
~~T -3 -5 0~~
F 3 0
F 5 0
c Observations
6 0
7 0

DPLL on DIMACS

c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
-6 -3 -4 -5 0
6 3 0
6 4 0
6 5 0
c x7 := x3 and x5
7 -3 -5 0
-7 3 0
-7 5 0
c Observations
6 0
7 0

c DIMACS
p cnf 7 19
c x3 := x1 or x2
F 1 2 0
~~T -1 0~~
~~T -2 0~~
c x4 := x1 and -x2
F -1 2 0
~~T 1 0~~
~~T -2 0~~
c x5 := x1 xor x2
F 1 2 0
F -1 -2 0
~~T -1 2 0~~
~~T 1 -2 0~~
c x6 := -x3 or -x4 or -x5
F -4 F 0
~~T 3 0~~
~~T 4 0~~
~~T 5 0~~
c x7 := x3 and x5
~~T -3 -5 0~~
F 3 0
F 5 0
c Observations
6 0
7 0

DPLL on DIMACS

```
c DIMACS
p cnf 7 19
c x3 := x1 or x2
-3 1 2 0
3 -1 0
3 -2 0
c x4 := x1 and -x2
4 -1 2 0
-4 1 0
-4 -2 0
c x5 := x1 xor x2
-5 1 2 0
-5 -1 -2 0
5 -1 2 0
5 1 -2 0
c x6 := -x3 or -x4 or -x5
-6 -3 -4 -5 0
6 3 0
6 4 0
6 5 0
c x7 := x3 and x5
7 -3 -5 0
-7 3 0
-7 5 0
c Observations
6 0
7 0
```

After repeatedly using unit propagation and simplification:

$\alpha(x_4) = 0$

$\alpha(x_5) = 1$

$\alpha(x_6) = 1$

$\alpha(x_7) = 1$

CNF reduces to:

$(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

Perform branching to solve:

$\alpha(x_1) = 0$

$\alpha(x_2) = 1$