State-of-the-art SAT Solving

Marijn J.H. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

SC^2 Summer School, July 31, 2017

Satisfiability (SAT) Solving Has Many Applications

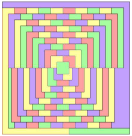
formal verification

security

bioinformatics

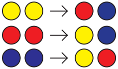
train safety

planning and scheduling



automated theorem proving

exploit generation



term rewriting termination

encode → **SAT solver** → **decode**

Dress Code as Satisfiability Problem

The SAT problem: Can a formula in propositional logic be satisfied?

Propositional logic

- Boolean variables : **tie** and **shirt** (for the example below)
- Logic symbols : $\neg$ (not), $\vee$ (or), $\wedge$ (and)
- Literals : **tie**, $\neg$**tie**, **shirt**, and $\neg$**shirt**

Three conditions / clauses :

- not wearing a **tie** nor a **shirt** is impolite $(\textbf{tie} \vee \textbf{shirt})$
- clearly one should not wear a **tie** without a **shirt** $(\neg\textbf{tie} \vee \textbf{shirt})$
- wearing a **tie** and a **shirt** is overkill $\neg(\textbf{tie} \wedge \textbf{shirt}) \equiv (\neg\textbf{tie} \vee \neg\textbf{shirt})$

Is the formula $(\textbf{tie} \vee \textbf{shirt}) \wedge (\neg\textbf{tie} \vee \textbf{shirt}) \wedge (\neg\textbf{tie} \vee \neg\textbf{shirt})$ satisfiable?

A Larger, but Still Small Satisfiability Problem

Is the formula below satisfiable?

$$
\begin{aligned}
&(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4) \wedge \\
&(x_1 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee \neg x_4 \vee \neg x_5) \wedge (x_2 \vee x_3 \vee x_5) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_5) \wedge \\
&(x_1 \vee x_5 \vee x_6) \wedge (\neg x_1 \vee \neg x_5 \vee \neg x_6) \wedge (x_2 \vee x_4 \vee x_6) \wedge (\neg x_2 \vee \neg x_4 \vee \neg x_6) \wedge \\
&(x_1 \vee x_6 \vee x_7) \wedge (\neg x_1 \vee \neg x_6 \vee \neg x_7) \wedge (x_2 \vee x_5 \vee x_7) \wedge (\neg x_2 \vee \neg x_5 \vee \neg x_7) \wedge \\
&(x_3 \vee x_4 \vee x_7) \wedge (\neg x_3 \vee \neg x_4 \vee \neg x_7) \wedge (x_1 \vee x_7 \vee x_8) \wedge (\neg x_1 \vee \neg x_7 \vee \neg x_8) \wedge \\
&(x_2 \vee x_6 \vee x_8) \wedge (\neg x_2 \vee \neg x_6 \vee \neg x_8) \wedge (x_3 \vee x_5 \vee x_8) \wedge (\neg x_3 \vee \neg x_5 \vee \neg x_8) \wedge \\
&(x_1 \vee x_8 \vee x_9) \wedge (\neg x_1 \vee \neg x_8 \vee \neg x_9) \wedge (x_2 \vee x_7 \vee x_9) \wedge (\neg x_2 \vee \neg x_7 \vee \neg x_9) \wedge \\
&\qquad (\neg x_3 \vee \neg x_6 \vee \neg x_9) \wedge (x_4 \vee x_5 \vee x_9) \wedge (\neg x_4 \vee \neg x_5 \vee \neg x_9)
\end{aligned}
$$

A Larger, but Still Small Satisfiability Problem

Is the formula below satisfiable?

$$
\begin{aligned}
&(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4) \wedge \\
&(x_1 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee \neg x_4 \vee \neg x_5) \wedge (x_2 \vee x_3 \vee x_5) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_5) \wedge \\
&(x_1 \vee x_5 \vee x_6) \wedge (\neg x_1 \vee \neg x_5 \vee \neg x_6) \wedge (x_2 \vee x_4 \vee x_6) \wedge (\neg x_2 \vee \neg x_4 \vee \neg x_6) \wedge \\
&(x_1 \vee x_6 \vee x_7) \wedge (\neg x_1 \vee \neg x_6 \vee \neg x_7) \wedge (x_2 \vee x_5 \vee x_7) \wedge (\neg x_2 \vee \neg x_5 \vee \neg x_7) \wedge \\
&(x_3 \vee x_4 \vee x_7) \wedge (\neg x_3 \vee \neg x_4 \vee \neg x_7) \wedge (x_1 \vee x_7 \vee x_8) \wedge (\neg x_1 \vee \neg x_7 \vee \neg x_8) \wedge \\
&(x_2 \vee x_6 \vee x_8) \wedge (\neg x_2 \vee \neg x_6 \vee \neg x_8) \wedge (x_3 \vee x_5 \vee x_8) \wedge (\neg x_3 \vee \neg x_5 \vee \neg x_8) \wedge \\
&(x_1 \vee x_8 \vee x_9) \wedge (\neg x_1 \vee \neg x_8 \vee \neg x_9) \wedge (x_2 \vee x_7 \vee x_9) \wedge (\neg x_2 \vee \neg x_7 \vee \neg x_9) \wedge \\
&\qquad (\neg x_3 \vee \neg x_6 \vee \neg x_9) \wedge (x_4 \vee x_5 \vee x_9) \wedge (\neg x_4 \vee \neg x_5 \vee \neg x_9)
\end{aligned}
$$

Yes. The correctness of the solution is easy to check.

A Larger, but Still Small Satisfiability Problem

Is the formula below still satisfiable?

$$
\begin{aligned}
&(x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \neg x_4) \land \\
&(x_1 \lor x_4 \lor x_5) \land (\neg x_1 \lor \neg x_4 \lor \neg x_5) \land (x_2 \lor x_3 \lor x_5) \land (\neg x_2 \lor \neg x_3 \lor \neg x_5) \land \\
&(x_1 \lor x_5 \lor x_6) \land (\neg x_1 \lor \neg x_5 \lor \neg x_6) \land (x_2 \lor x_4 \lor x_6) \land (\neg x_2 \lor \neg x_4 \lor \neg x_6) \land \\
&(x_1 \lor x_6 \lor x_7) \land (\neg x_1 \lor \neg x_6 \lor \neg x_7) \land (x_2 \lor x_5 \lor x_7) \land (\neg x_2 \lor \neg x_5 \lor \neg x_7) \land \\
&(x_3 \lor x_4 \lor x_7) \land (\neg x_3 \lor \neg x_4 \lor \neg x_7) \land (x_1 \lor x_7 \lor x_8) \land (\neg x_1 \lor \neg x_7 \lor \neg x_8) \land \\
&(x_2 \lor x_6 \lor x_8) \land (\neg x_2 \lor \neg x_6 \lor \neg x_8) \land (x_3 \lor x_5 \lor x_8) \land (\neg x_3 \lor \neg x_5 \lor \neg x_8) \land \\
&(x_1 \lor x_8 \lor x_9) \land (\neg x_1 \lor \neg x_8 \lor \neg x_9) \land (x_2 \lor x_7 \lor x_9) \land (\neg x_2 \lor \neg x_7 \lor \neg x_9) \land \\
&(x_3 \lor x_6 \lor x_9) \land (\neg x_3 \lor \neg x_6 \lor \neg x_9) \land (x_4 \lor x_5 \lor x_9) \land (\neg x_4 \lor \neg x_5 \lor \neg x_9)
\end{aligned}
$$

No. Adding a single clause eliminates all solutions.

Checking a No answer can be expensive.

Satisfiability as the Cornerstone of the P = NP Question

A fundamental question in computer science asks whether searching for a solution is harder than verifying a given solution.

For example, consider the Sudoku on the right: Is searching for the solution harder than verifying a given candidate solution?

	4		3					
						7	9	
			6					
			1	4		5		
9							1	
2								6
				7	2			
	5					8		
				9				

Satisfiability as the Cornerstone of the P = NP Question

A fundamental question in computer science asks whether searching for a solution is harder than verifying a given solution.

For example, consider the Sudoku on the right: Is searching for the solution harder than verifying a given candidate solution?

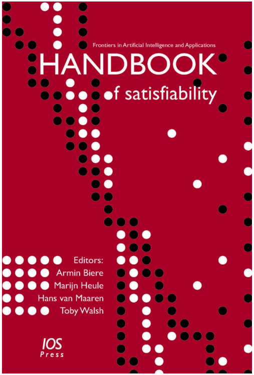
1	4	7	3	8	9	2	6	5
5	8	6	2	1	4	7	9	3
3	9	2	6	5	7	1	8	4
8	7	3	1	4	6	5	2	9
9	6	4	7	2	5	3	1	8
2	1	5	9	3	8	4	7	6
6	3	8	5	7	2	9	4	1
7	5	9	4	6	1	8	3	2
4	2	1	8	9	3	6	5	7

This is the P = NP question. Solving it is worth $1,000,000.

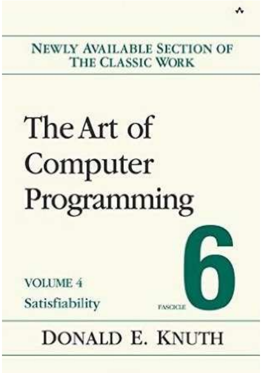
Cook-Levin Theorem [1971]: SAT is NP-complete.
Searching is as easy as verifying if and only if this holds for SAT.

Enormous Progress in the Last Two Decades

mid '90s: formulas solvable with thousands of variables and clauses
now: formulas solvable with millions of variables and clauses



Edmund Clarke: *"a key technology of the 21st century"*



Donald Knuth: *"evidently a killer app, because it is key to the solution of so many other problems"*

Overview

Search for Lemmas (now)

- Learning Lemmas
- Data-structures
- Heuristics

Search for Simplification (after the break)

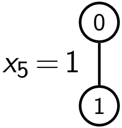
- Variable elimination
- Blocked clause elimination
- Unhiding redundancy

Conflict-driven SAT solvers: Search and Analysis

$$(x_1 \lor x_4) \land$$
$$(x_3 \lor \neg x_4 \lor \neg x_5) \land$$
$$(\neg x_3 \lor \neg x_2 \lor \neg x_4) \land$$
$$\mathcal{F}_{\text{extra}}$$

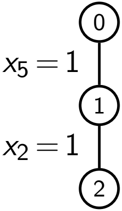
Conflict-driven SAT solvers: Search and Analysis

$$
\begin{aligned}
&(x_1 \lor x_4) \land \\
&(x_3 \lor \neg x_4 \lor \neg x_5) \land \\
&(\neg x_3 \lor \neg x_2 \lor \neg x_4) \land \\
&\mathcal{F}_{\text{extra}}
\end{aligned}
$$



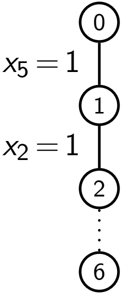
Conflict-driven SAT solvers: Search and Analysis

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \neg x_4 \vee \neg x_5) \wedge$$
$$(\neg x_3 \vee \neg x_2 \vee \neg x_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$



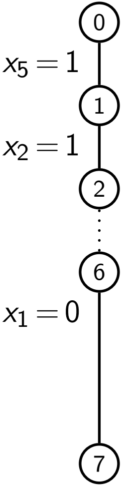
Conflict-driven SAT solvers: Search and Analysis

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \neg x_4 \vee \neg x_5) \wedge$$
$$(\neg x_3 \vee \neg x_2 \vee \neg x_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$



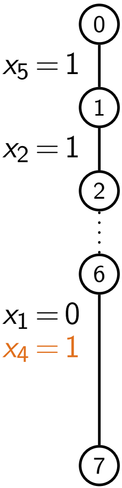
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \neg x_4 \vee \neg x_5) \wedge$
$(\neg x_3 \vee \neg x_2 \vee \neg x_4) \wedge$
$\mathcal{F}_{\text{extra}}$



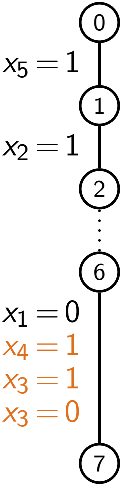
Conflict-driven SAT solvers: Search and Analysis

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \neg x_4 \vee \neg x_5) \wedge$$
$$(\neg x_3 \vee \neg x_2 \vee \neg x_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$



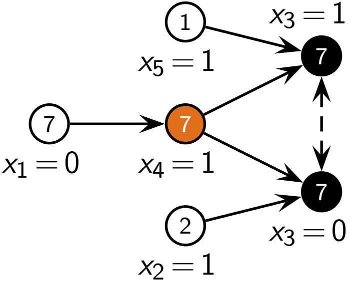
Conflict-driven SAT solvers: Search and Analysis

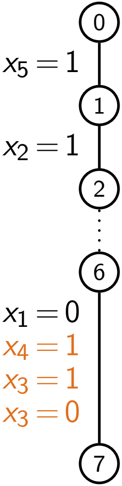
$$
\begin{aligned}
&(x_1 \lor x_4) \land \\
&(x_3 \lor \neg x_4 \lor \neg x_5) \land \\
&(\neg x_3 \lor \neg x_2 \lor \neg x_4) \land \\
&\mathcal{F}_{\text{extra}}
\end{aligned}
$$



Conflict-driven SAT solvers: Search and Analysis

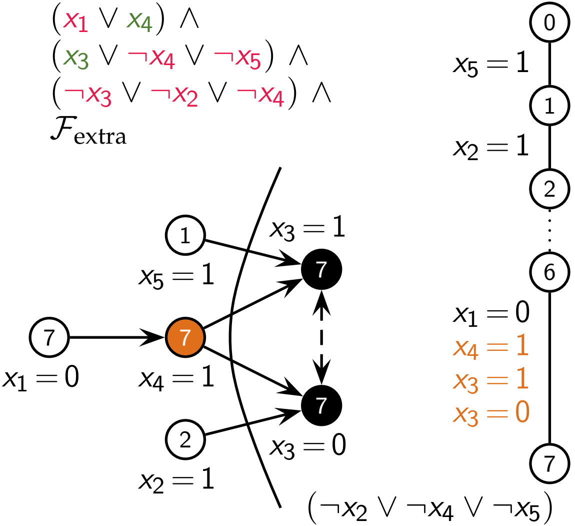
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \neg x_4 \vee \neg x_5) \wedge$
$(\neg x_3 \vee \neg x_2 \vee \neg x_4) \wedge$
$\mathcal{F}_{\text{extra}}$





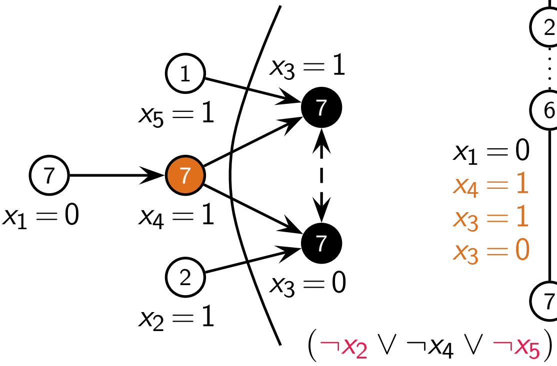
Conflict-driven SAT solvers: Search and Analysis

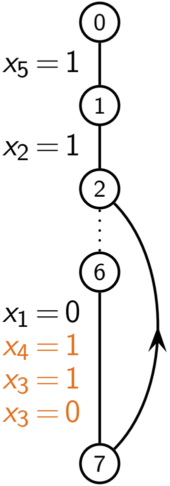
$(x_1 \lor x_4) \land$
$(x_3 \lor \neg x_4 \lor \neg x_5) \land$
$(\neg x_3 \lor \neg x_2 \lor \neg x_4) \land$
$\mathcal{F}_{\text{extra}}$



Conflict-driven SAT solvers: Search and Analysis

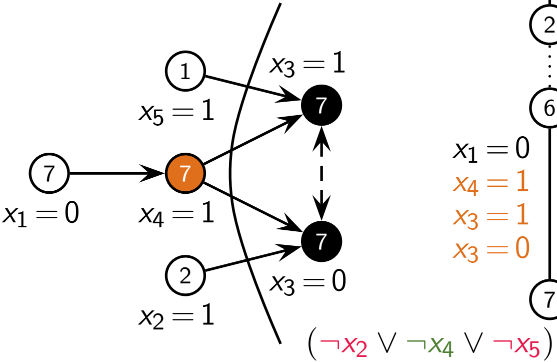
$$(x_1 \lor x_4) \land$$
$$(x_3 \lor \neg x_4 \lor \neg x_5) \land$$
$$(\neg x_3 \lor \neg x_2 \lor \neg x_4) \land$$
$$\mathcal{F}_{\text{extra}}$$

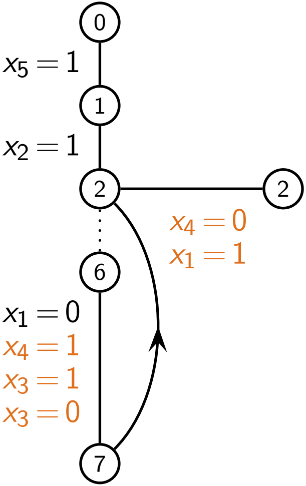




Conflict-driven SAT solvers: Search and Analysis

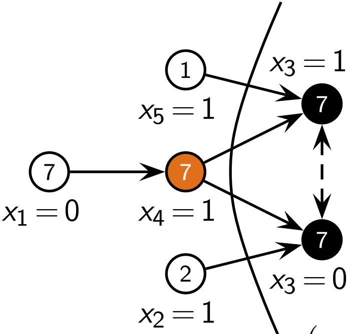
$$(x_1 \lor x_4) \land$$
$$(x_3 \lor \neg x_4 \lor \neg x_5) \land$$
$$(\neg x_3 \lor \neg x_2 \lor \neg x_4) \land$$
$$\mathcal{F}_{\text{extra}}$$



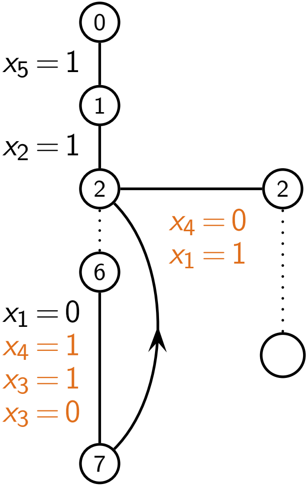


Conflict-driven SAT solvers: Search and Analysis

$(x_1 \lor x_4) \land$
$(x_3 \lor \neg x_4 \lor \neg x_5) \land$
$(\neg x_3 \lor \neg x_2 \lor \neg x_4) \land$
$\mathcal{F}_{\text{extra}}$



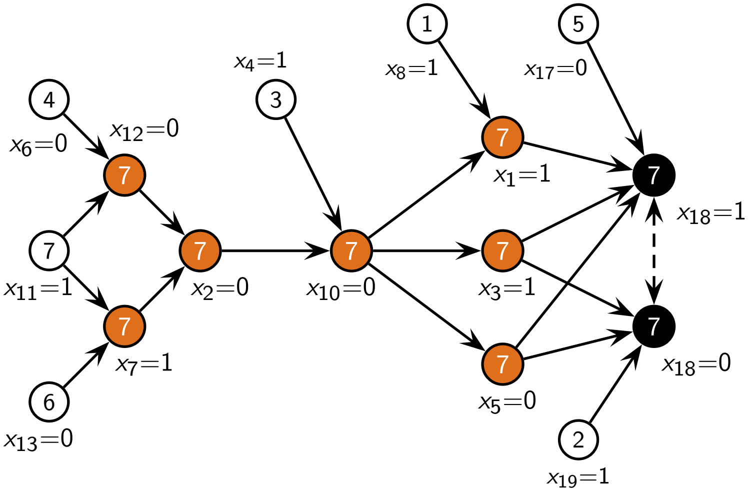
$(\neg x_2 \lor \neg x_4 \lor \neg x_5)$



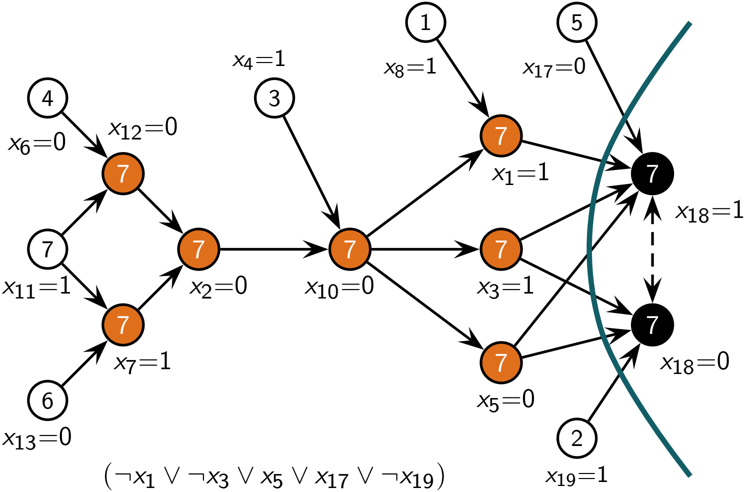
Conflict-driven SAT solvers: Pseudo-code

while TRUE **do**
    $l_{\text{decision}} := \textsc{GetDecisionLiteral}(\ )$
    If no l_{decision} **then return** `satisfiable`
    $\mathcal{F} := \textsc{Simplify}(\ \mathcal{F}(l_{\text{decision}} \leftarrow 1)\ )$
    while $\mathcal{F}$ contains $C_{\text{falsified}}$ **do**
        $C_{\text{conflict}} := \textsc{AnalyzeConflict}(\ C_{\text{falsified}}\ )$
        If $C_{\text{conflict}} = \emptyset$ **then return** `unsatisfiable`
        $\textsc{BackTrack}(\ C_{\text{conflict}}\ )$
        $\mathcal{F} := \textsc{Simplify}(\ \mathcal{F} \cup \{C_{\text{conflict}}\}\ )$
    end while
end while

Learning conflict clauses [Marques-SilvaSakallah'96]



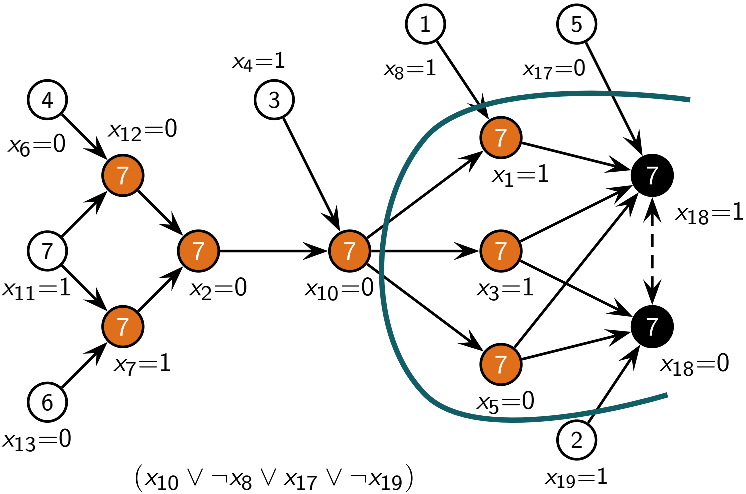
Learning conflict clauses [Marques-SilvaSakallah'96]



$(\neg x_1 \vee \neg x_3 \vee x_5 \vee x_{17} \vee \neg x_{19})$

tri-asserting clause

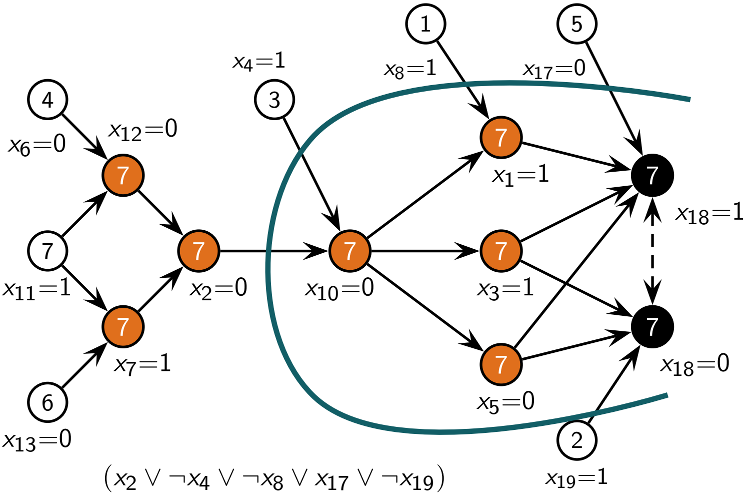
Learning conflict clauses [Marques-SilvaSakallah'96]



$(x_{10} \vee \neg x_8 \vee x_{17} \vee \neg x_{19})$

first unique implication point

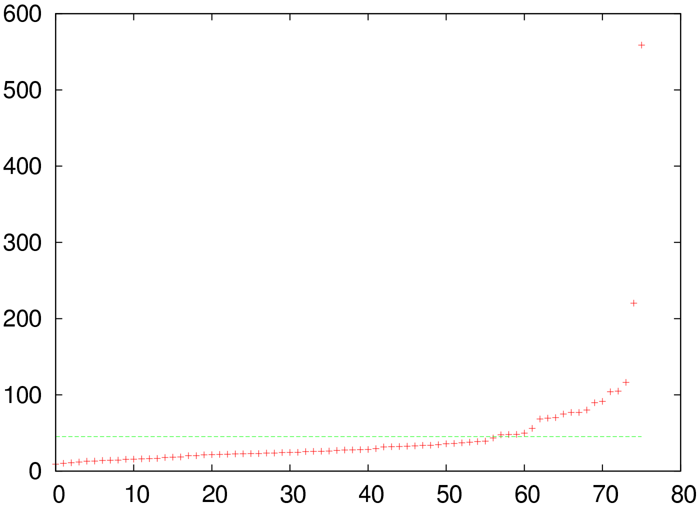
Learning conflict clauses [Marques-SilvaSakallah'96]



$(x_2 \vee \neg x_4 \vee \neg x_8 \vee x_{17} \vee \neg x_{19})$

second unique implication point

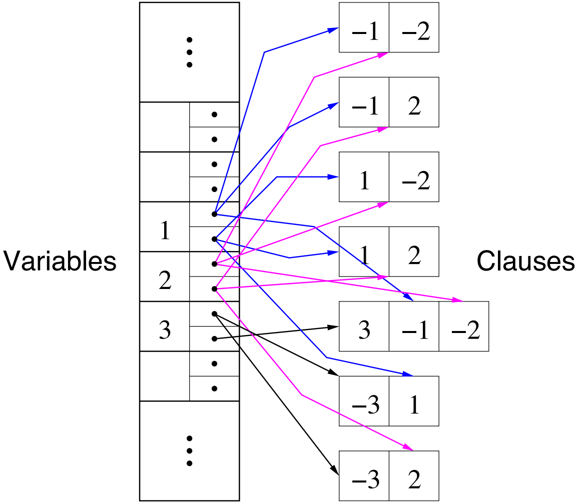
Average Learned Clause Length



Data-structures

Watch pointers

Simple data structure for unit propagation



Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

$$\varphi = \{x_1 = *, x_2 = *, x_3 = *, x_4 = *, x_5 = *, x_6 = *\}$$

$\neg x_1$	x_2	$\neg x_3$	$\neg x_5$	x_6

x_1	$\neg x_3$	x_4	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

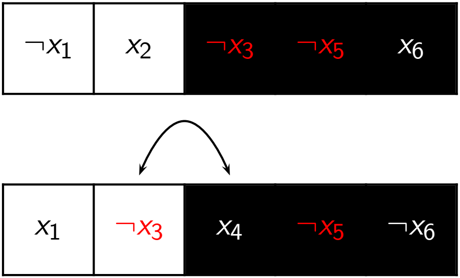
$$\varphi = \{x_1 = *, x_2 = *, x_3 = *, x_4 = *, x_5 = \mathbf{1}, x_6 = *\}$$

$\neg x_1$	x_2	$\neg x_3$	$\neg x_5$	x_6

x_1	$\neg x_3$	x_4	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

$$\varphi = \{x_1 = *, x_2 = *, x_3 = \mathbf{1}, x_4 = *, x_5 = 1, x_6 = *\}$$



Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

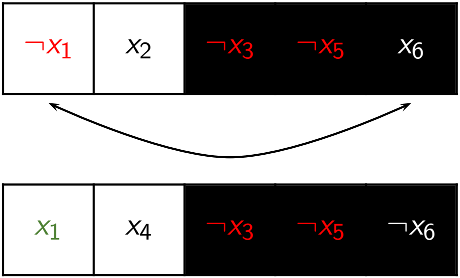
$$\varphi = \{x_1 = *, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = *\}$$

$\neg x_1$	x_2	$\neg x_3$	$\neg x_5$	x_6

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

$$\varphi = \{x_1 = \mathbf{1}, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = *\}$$



Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

$$\varphi = \{x_1 = 1, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = *\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

$$\varphi = \{x_1 = 1, x_2 = *, x_3 = 1, x_4 = \mathbf{0}, x_5 = 1, x_6 = *\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

$$\varphi = \{x_1 = 1, x_2 = \mathbf{0}, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = *\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

$$\varphi = \{x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = \mathbf{1}\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1) [MoskewiczMZZM'01]

$$\varphi = \{x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 1\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (2) [MoskewiczMZZM'01]

Only examine (get in the cache) a clause when both

- a watch pointer gets falsified
- the other one is not satisfied

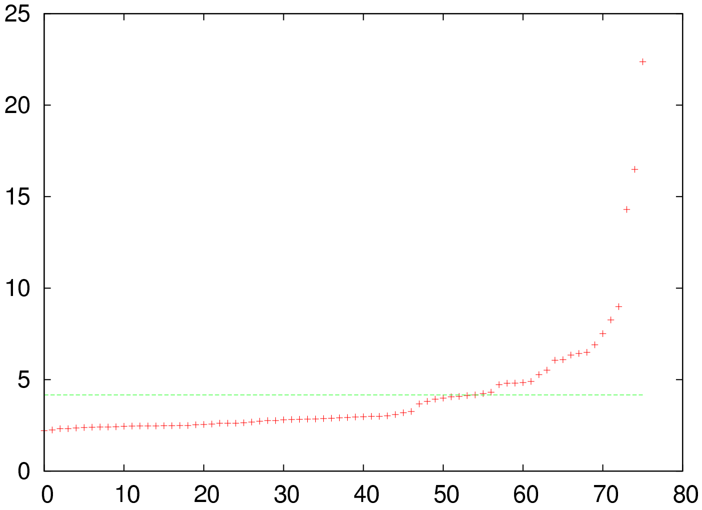
While backjumping, just unassign variables

Conflict clauses $\rightarrow$ watch pointers

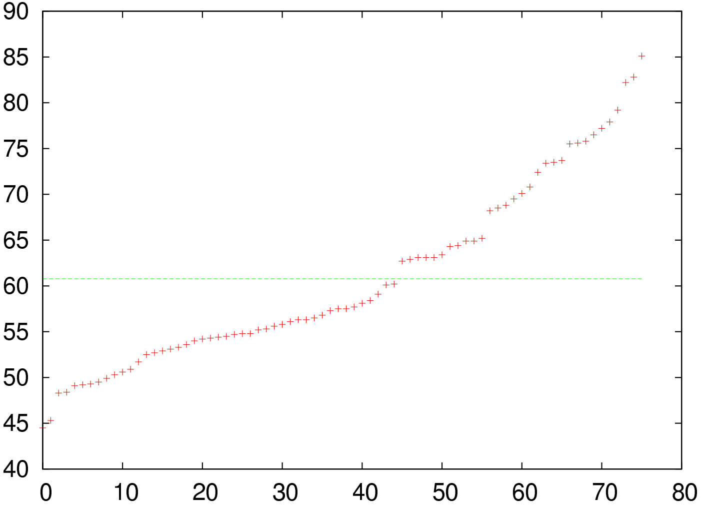
No detailed information available

Not used for binary clauses

Average Number Clauses Visited Per Propagation



Percentage visited clauses with other watched literal true



Heuristics

Most important CDCL heuristics

Variable selection heuristics

- aim: minimize the search space
- plus: could compensate a bad value selection

Most important CDCL heuristics

Variable selection heuristics

- aim: minimize the search space
- plus: could compensate a bad value selection

Value selection heuristics

- aim: guide search towards a solution (or conflict)
- plus: could compensate a bad variable selection, cache solutions of subproblems [PipatsrisawatDarwiche'07]

Most important CDCL heuristics

Variable selection heuristics

- aim: minimize the search space
- plus: could compensate a bad value selection

Value selection heuristics

- aim: guide search towards a solution (or conflict)
- plus: could compensate a bad variable selection, cache solutions of subproblems [PipatsrisawatDarwiche'07]

Restart strategies

- aim: avoid heavy-tail behavior [GomesSelmanCrato'97]
- plus: focus search on recent conflicts when combined with dynamic heuristics

Variable selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Variable selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Variable State Independent Decaying Sum (VSIDS)

- original idea (zChaff): for each conflict, increase the score of involved variables by 1, half all scores each 256 conflicts [MoskewiczMZZM'01]
- improvement (MiniSAT): for each conflict, increase the score of involved variables by δ and increase $\delta := 1.05\delta$ [EenSörensson'03]

Visualization of VSIDS in PicoSAT

http://www.youtube.com/watch?v=MOjhFywLre8

Value selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Value selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Based on the encoding / consequently

- negative branching (early MiniSAT) [EenSörensson'03]

Value selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Based on the encoding / consequently

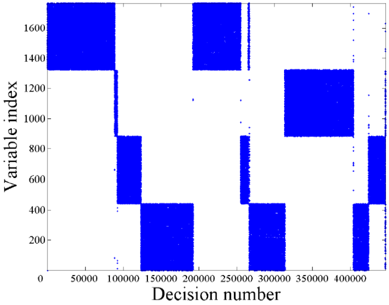
- negative branching (early MiniSAT) [EenSörensson'03]

Based on the last implied value (phase-saving)

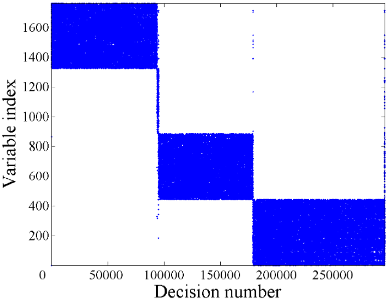
- introduced to CDCL [PipatsrisawatDarwiche'07]
- already used in local search [HirschKojevnikov'01]

Heuristics: Phase-saving [PipatsrisawatDarwiche'07]

Selecting the last implied value remembers solved components



negative branching



phase-saving

Restarts

Restarts in CDCL solvers:

- Counter heavy-tail behavior [GomesSelmanCrato'97]
- Unassign all variables but keep the (dynamic) heuristics

Restarts

Restarts in CDCL solvers:

- Counter heavy-tail behavior [GomesSelmanCrato'97]
- Unassign all variables but keep the (dynamic) heuristics

Restart strategies: [Walsh'99, LubySinclairZuckerman'93]

- Geometrical restart: e.g. 100, 150, 225, 333, 500, 750, . . .
- Luby sequence: e.g. 100, 100, 200, 100, 100, 200, 400, . . .

Restarts

Restarts in CDCL solvers:

- Counter heavy-tail behavior [GomesSelmanCrato'97]
- Unassign all variables but keep the (dynamic) heuristics

Restart strategies: [Walsh'99, LubySinclairZuckerman'93]

- Geometrical restart: e.g. 100, 150, 225, 333, 500, 750, . . .
- Luby sequence: e.g. 100, 100, 200, 100, 100, 200, 400, . . .

Rapid restarts by reusing trail:
[vanderTakHeuleRamos'11]

- Partial restart same effect as full restart
- Optimal strategy Luby-1: 1, 1, 2, 1, 1, 2, 4, . . .

Conflict-Clause Minimization

Self-Subsumption

Use self-subsumption to shorten conflict clauses

$$\frac{C \vee l \quad D \vee \neg l}{D} \quad C \subseteq D \qquad \frac{(a \vee b \vee l) \quad (a \vee b \vee c \vee \neg l)}{(a \vee b \vee c)}$$

Conflict clause minimization is an important optimization.

Self-Subsumption

Use self-subsumption to shorten conflict clauses

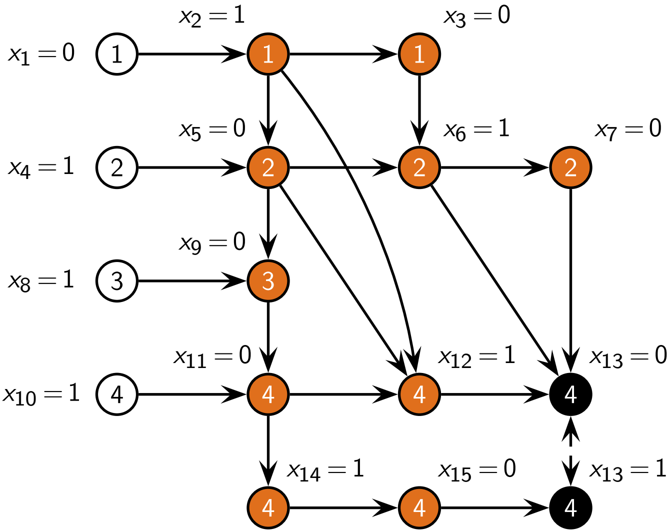
$$\frac{C \vee l \quad D \vee \neg l}{D} \quad C \subseteq D \qquad \frac{(a \vee b \vee l) \quad (a \vee b \vee c \vee \neg l)}{(a \vee b \vee c)}$$

Conflict clause minimization is an important optimization.

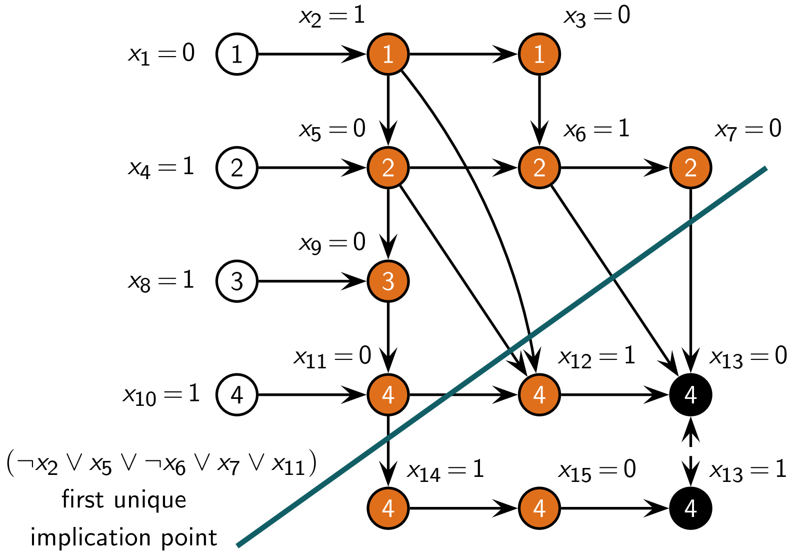
Use implication chains to further minimization:

$$\ldots(\neg a \vee b)(\neg b \vee c)(a \vee c \vee d)\ldots \quad \Rightarrow$$
$$\ldots(\neg a \vee b)(\neg b \vee c)(c \vee d)\ldots$$

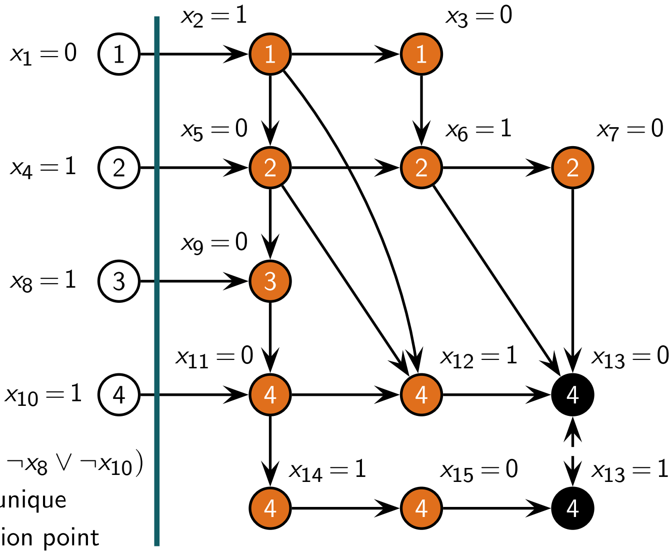
Conflict-clause minimization [SörenssonBiere'09]



Conflict-clause minimization [SörenssonBiere'09]



Conflict-clause minimization [SörenssonBiere'09]

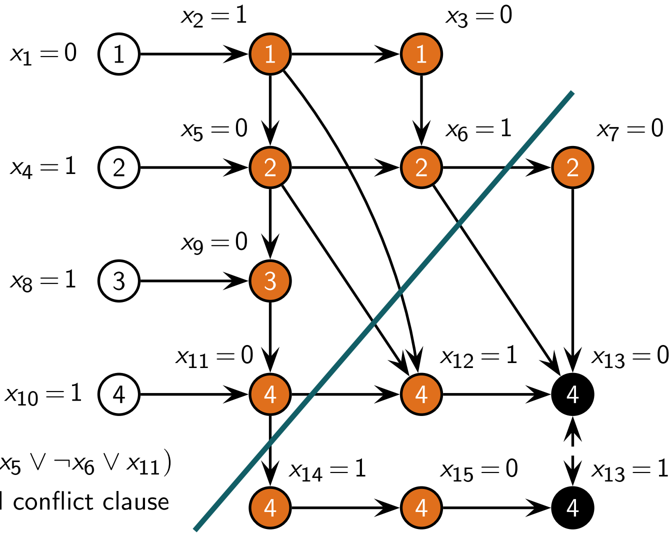


$(x_1 \vee \neg x_4 \vee \neg x_8 \vee \neg x_{10})$

last unique

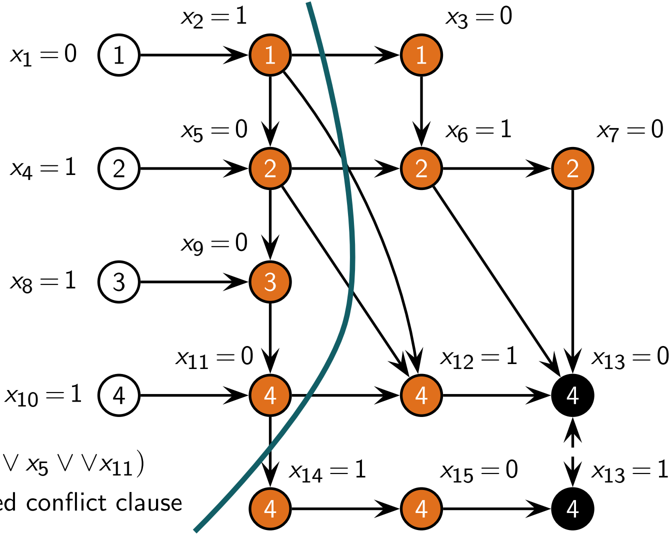
implication point

Conflict-clause minimization [SörenssonBiere'09]



$(\neg x_2 \vee x_5 \vee \neg x_6 \vee x_{11})$
reduced conflict clause

Conflict-clause minimization [SörenssonBiere'09]



$(\neg x_2 \vee x_5 \vee \vee x_{11})$
minimized conflict clause

Conclusions: state-of-the-art CDCL solver

Key contributions to CDCL solvers:

- concept of conflict clauses (grasp) [Marques-SilvaSakallah'96]
- restart strategies [GomesSC'97,LubySZ'93]
- 2-watch pointers and VSIDS (zChaff) [MoskewiczMZZM'01]
- efficient implementation (Minisat) [EenSörensson'03]
- phase-saving (Rsat) [PipatsrisawatDarwiche'07]
- conflict-clause minimization [SörenssonBiere'09]

+ Pre- and in-processing techniques

State-of-the-art SAT Solving

Marijn J.H. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

SC^2 Summer School, July 31, 2017