State-of-the-art SAT Solving

Marijn J.H. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

Summer School on Formal Techniques, May 22, 2017

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?

The Satisfiability (SAT) problem

$$
\begin{aligned}
&(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
&(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
&(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
&(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
&(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
&(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
&(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
&(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
&(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
&(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
&(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{aligned}
$$

Does there exist an assignment satisfying all clauses?

Search for a satisfying assignment (or proof none exists)

$$
\begin{aligned}
&(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
&(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
&(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
&(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
&(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
&(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
&(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
&(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
&(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
&(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
&(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{aligned}
$$

$\mathcal{NP}$-Complete: Good or Bad News?

SAT is the first $\mathcal{NP}$-complete problem [Cook'71]

$\mathcal{NP}$-Complete: Good or Bad News?

SAT is the first $\mathcal{NP}$-complete problem [Cook'71]

Bad?

- Only exponential time solving algorithms are known
- Probably no polynomial time algorithm exists ($\mathcal{P} \neq \mathcal{NP}$)

$\mathcal{NP}$-Complete: Good or Bad News?

SAT is the first $\mathcal{NP}$-complete problem [Cook'71]

Bad?

- Only exponential time solving algorithms are known
- Probably no polynomial time algorithm exists ($\mathcal{P} \neq \mathcal{NP}$)

Good!

- SAT solvers are powerful tools for real world problems
- All problems in $\mathcal{NP}$ can be translated into SAT in polynomial time

Motivation

From 100 variables, 200 constraints (early 90s)
to 1,000,000 vars. and 20,000,000 cls. in 20 years.

Motivation

From 100 variables, 200 constraints (early 90s)
to 1,000,000 vars. and 20,000,000 cls. in 20 years.

Applications:
Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.

Motivation

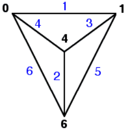
From 100 variables, 200 constraints (early 90s)
to 1,000,000 vars. and 20,000,000 cls. in 20 years.

Applications:
Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.

SAT used to solve many other problems!

Capitalise on the performance of SAT solvers

formal verification

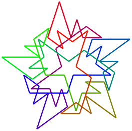


graph theory

bioinformatics

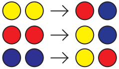
train safety

timetabling



number theory

cryptography



rewriting termination

encode

SAT solver

decode

Overview

Search for Lemmas (Today)

- Learning Lemmas
- Data-structures
- Heuristics

Search for Simplification (Tomorrow)

- Variable elimination
- Blocked clause elimination
- Unhiding redundancy

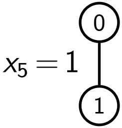
Conflict-driven SAT solvers: Search and Analysis

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$$
$$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$$
$$\mathcal{F}_{\mathrm{extra}}$$

0

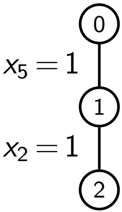
Conflict-driven SAT solvers: Search and Analysis

$$
\begin{aligned}
&(x_1 \vee x_4) \wedge \\
&(x_3 \vee \bar{x}_4 \vee {\color{red}\bar{x}_5}) \wedge \\
&(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge \\
&\mathcal{F}_{\mathrm{extra}}
\end{aligned}
$$



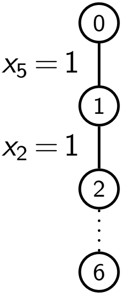
Conflict-driven SAT solvers: Search and Analysis

$$
\begin{aligned}
&(x_1 \vee x_4) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \color{red}{\bar{x}_5}) \wedge \\
&(\bar{x}_3 \vee \color{red}{\bar{x}_2} \vee \bar{x}_4) \wedge \\
&\mathcal{F}_{\mathrm{extra}}
\end{aligned}
$$



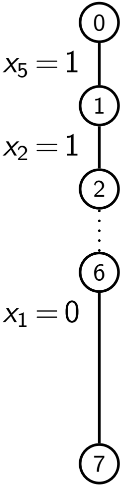
Conflict-driven SAT solvers: Search and Analysis

$$
\begin{aligned}
&(x_1 \vee x_4) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge \\
&(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge \\
&\mathcal{F}_{\mathrm{extra}}
\end{aligned}
$$



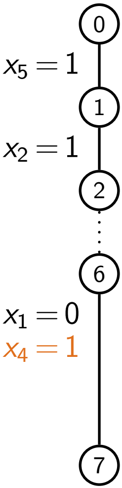
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$



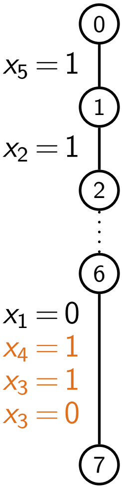
Conflict-driven SAT solvers: Search and Analysis

$$
\begin{aligned}
&(x_1 \vee x_4) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge \\
&(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge \\
&\mathcal{F}_{\text{extra}}
\end{aligned}
$$



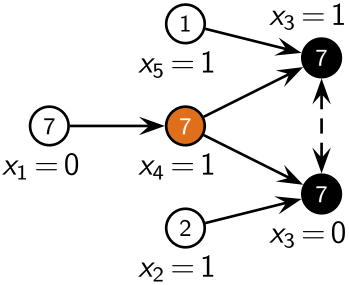
Conflict-driven SAT solvers: Search and Analysis

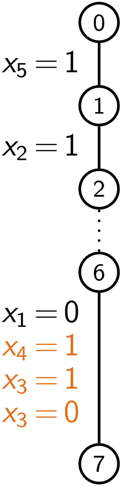
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$



Conflict-driven SAT solvers: Search and Analysis

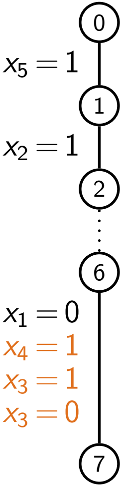
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$

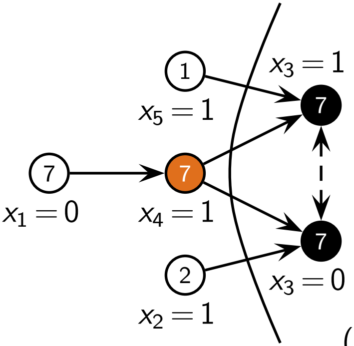




Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \ \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \ \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \ \wedge$
$\mathcal{F}_{\text{extra}}$

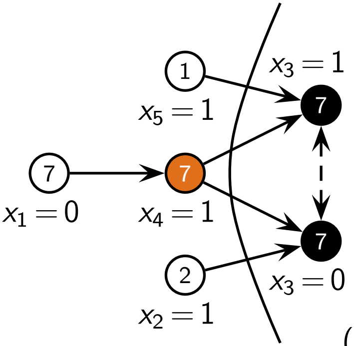


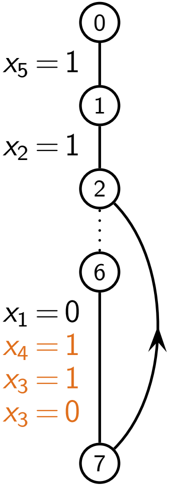


$(\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5)$

Conflict-driven SAT solvers: Search and Analysis

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$$
$$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$

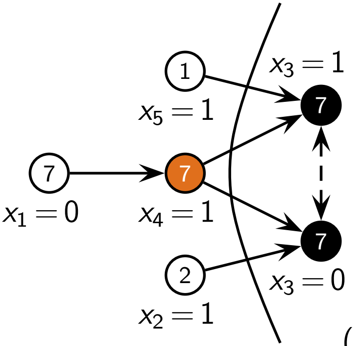


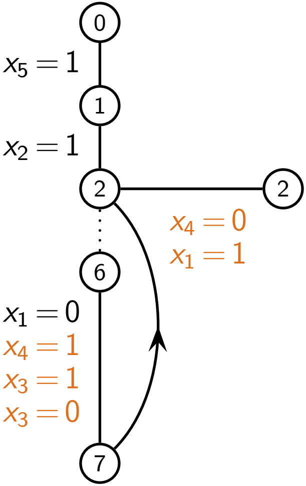


$$(\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5)$$

Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4)\ \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5)\ \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4)\ \wedge$
$\mathcal{F}_{\text{extra}}$

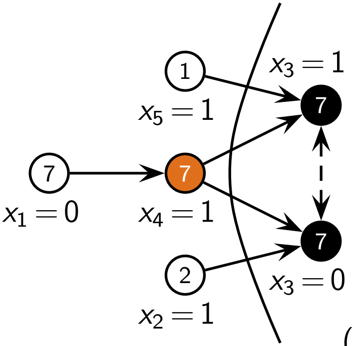


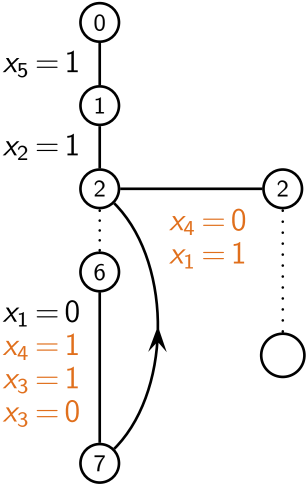


$(\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5)$

Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\mathrm{extra}}$



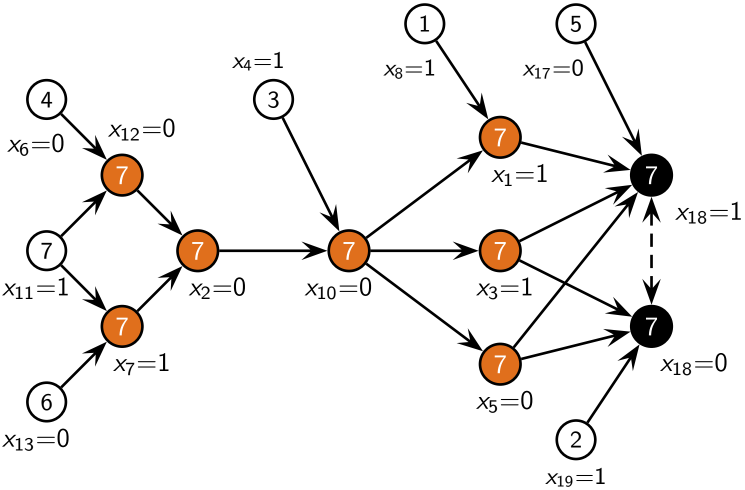


$(\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5)$

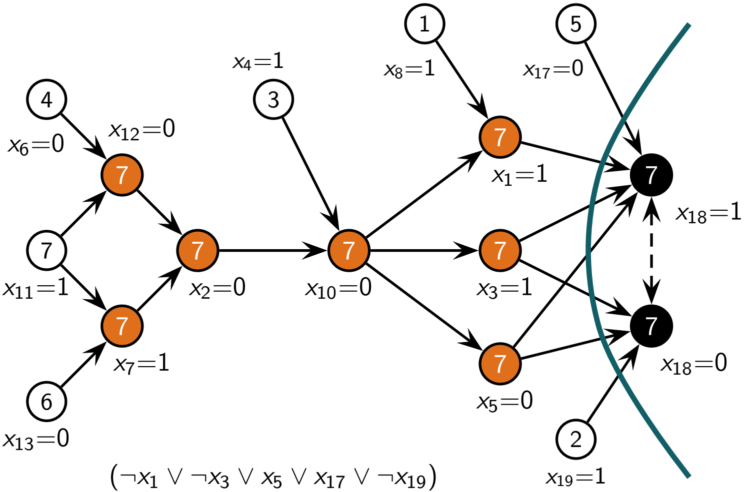
Conflict-driven SAT solvers: Pseudo-code

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]



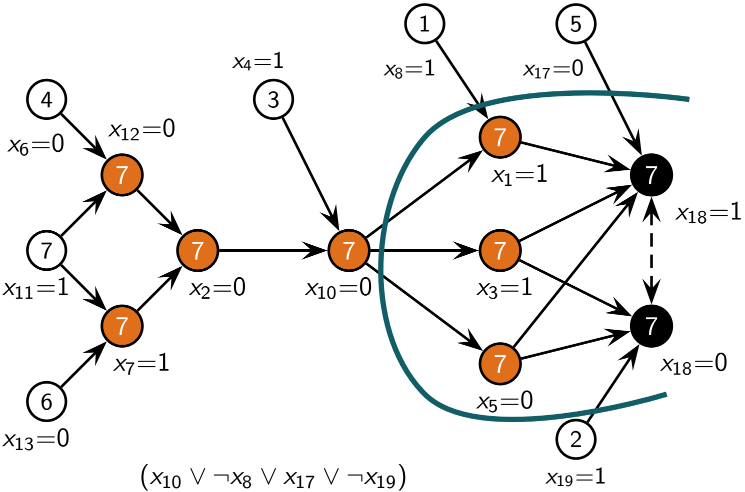
Learning conflict clauses [Marques-SilvaSakallah'96]



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

tri-asserting clause

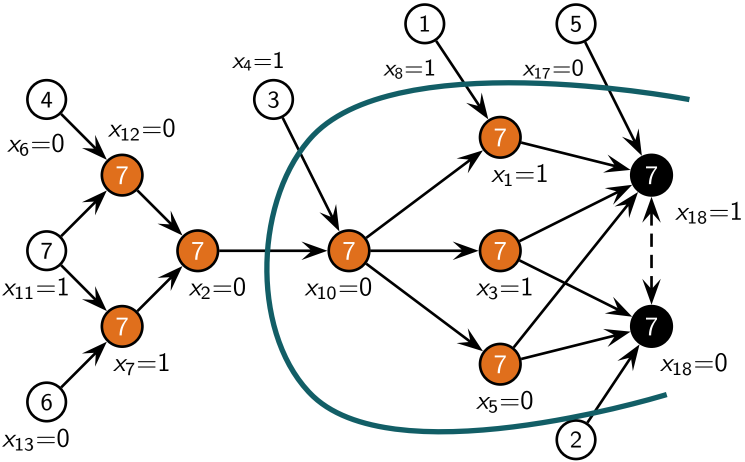
Learning conflict clauses [Marques-SilvaSakallah'96]



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

first unique implication point

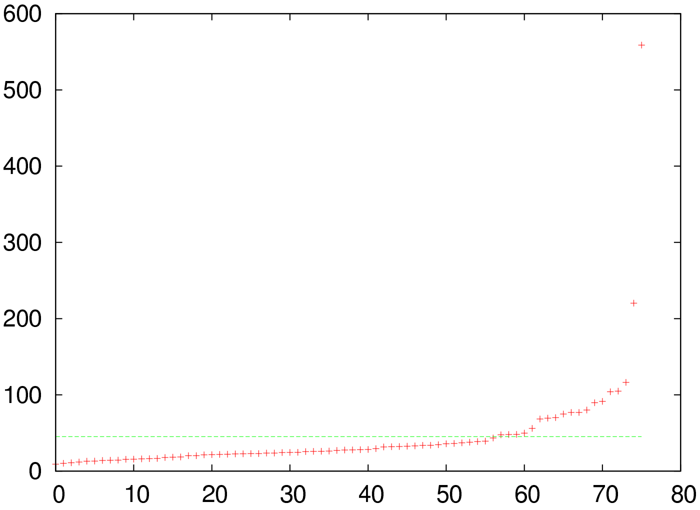
Learning conflict clauses [Marques-SilvaSakallah'96]



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

second unique implication point

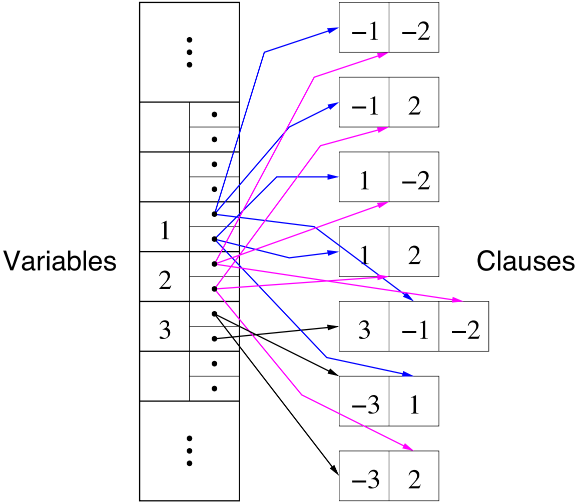
Average Learned Clause Length



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]

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

$\neg x_1$	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]

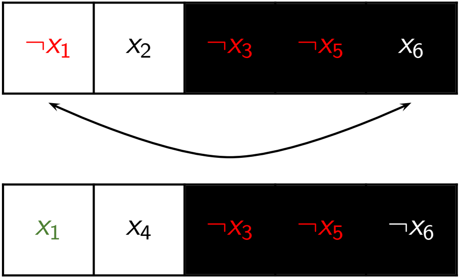
$$\varphi = \{x_1 = *, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = *\}$$

$\neg x_1$	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

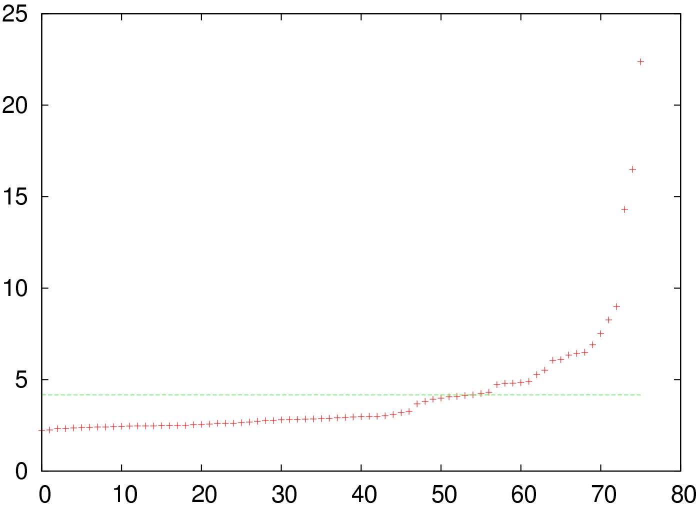
While backjumping, just unassign variables

Conflict clauses $\rightarrow$ watch pointers

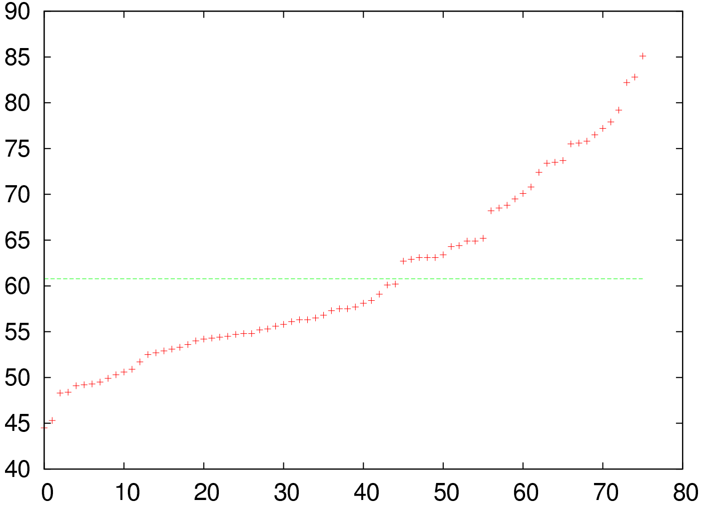
No detailed information available

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

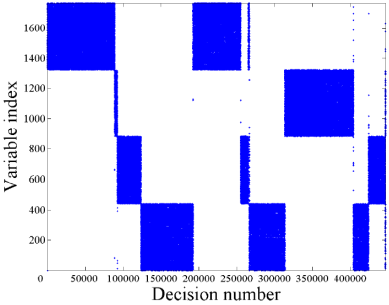
- negative branching (early MiniSAT) [EenSörensson'03]

Based on the last implied value (phase-saving)

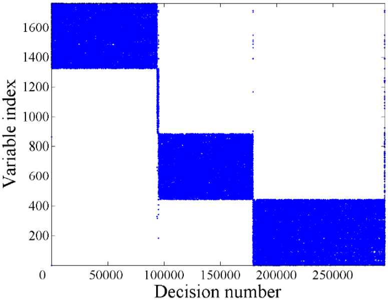
- introduced to CDCL [PipatsrisawatDarwiche'07]
- already used in local search [HirschKojevnikov'01]

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. $100, 150, 225, 333, 500, 750, \ldots$
- Luby sequence: e.g. $100, 100, 200, 100, 100, 200, 400, \ldots$

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. $100, 150, 225, 333, 500, 750, \ldots$
- Luby sequence: e.g. $100, 100, 200, 100, 100, 200, 400, \ldots$

Rapid restarts by reusing trail: [vanderTakHeuleRamos'11]

- Partial restart same effect as full restart
- Optimal strategy Luby-1: $1, 1, 2, 1, 1, 2, 4, \ldots$

Conflict-Clause Minimization

Self-Subsumption

Use self-subsumption to shorten conflict clauses

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

Conflict clause minimization is an important optimization.

Self-Subsumption

Use self-subsumption to shorten conflict clauses

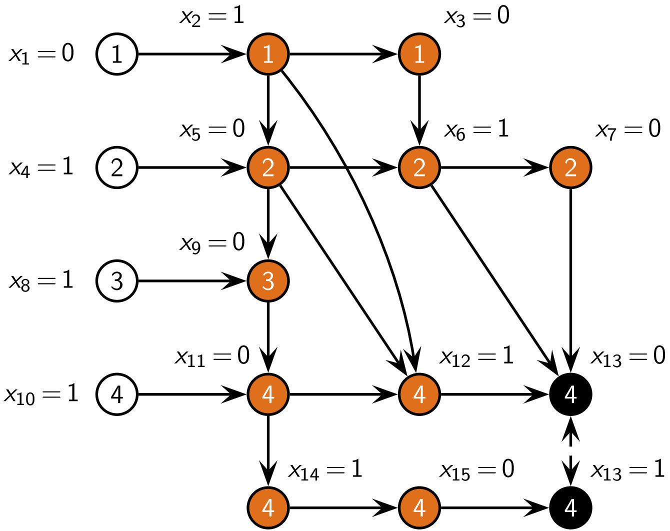
$$\frac{C \vee l \quad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{(a \vee b \vee l) \quad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

Conflict clause minimization is an important optimization.

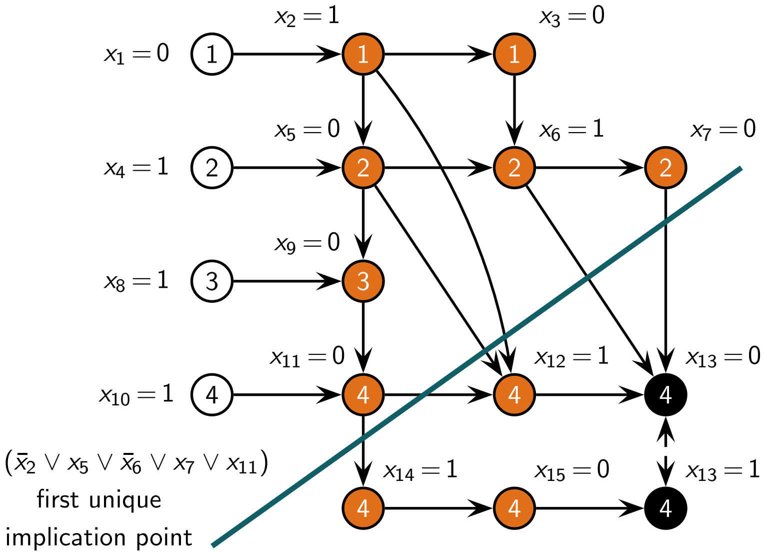
Use implication chains to further minimization:

$$\ldots(\bar{a} \vee b)(\bar{b} \vee c)(a \vee c \vee d)\ldots \quad \Rightarrow \quad \ldots(\bar{a} \vee b)(\bar{b} \vee c)(c \vee d)\ldots$$

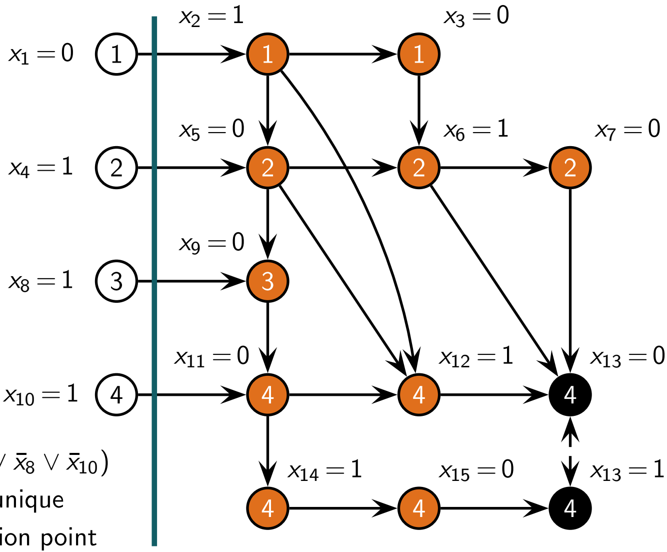
Conflict-clause minimization [SörenssonBiere'09]



Conflict-clause minimization [SörenssonBiere'09]



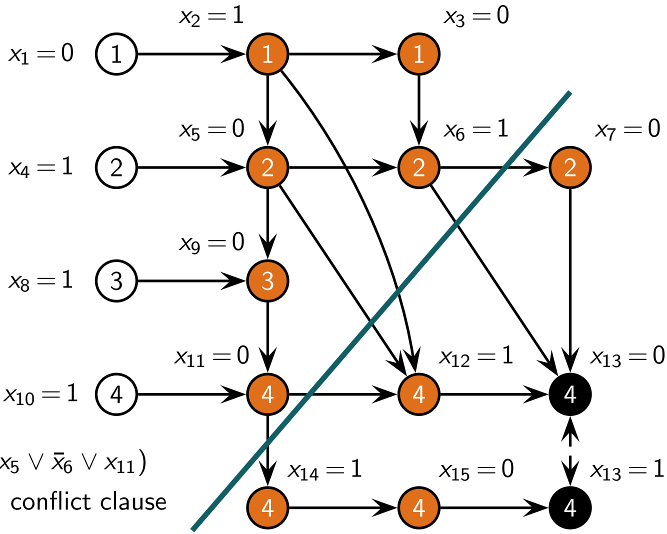
Conflict-clause minimization [SörenssonBiere'09]



$(x_1 \vee \bar{x}_4 \vee \bar{x}_8 \vee \bar{x}_{10})$

last unique implication point

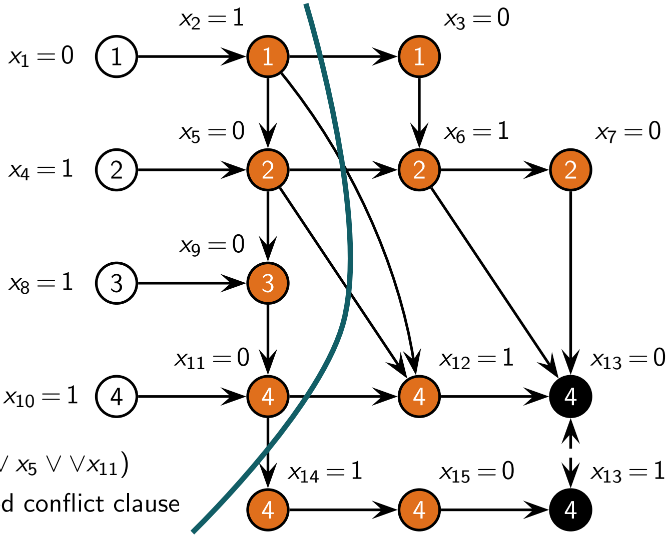
Conflict-clause minimization [SörenssonBiere'09]



$(\bar{x}_2 \vee x_5 \vee \bar{x}_6 \vee x_{11})$

reduced conflict clause

Conflict-clause minimization [SörenssonBiere'09]



$(\bar{x}_2 \vee x_5 \vee \vee x_{11})$

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