Conflict-Driven Clause Learning

Marijn J.H. Heule

Carnegie Mellon University

http://www.cs.cmu.edu/~mheule/15816-f24/

Automated Reasoning and Satisfiability

September 11, 2024

The Satisfiability (SAT) problem

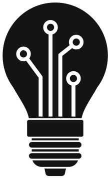
$$
\begin{aligned}
&(x_5 \lor x_8 \lor \overline{x}_2) \land (x_2 \lor \overline{x}_1 \lor \overline{x}_3) \land (\overline{x}_8 \lor \overline{x}_3 \lor \overline{x}_7) \land (\overline{x}_5 \lor x_3 \lor x_8) \land \\
&(\overline{x}_6 \lor \overline{x}_1 \lor \overline{x}_5) \land (x_8 \lor \overline{x}_9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

SAT Solver Paradigms Overview

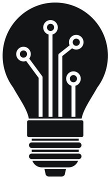


DPLL: Aims at finding a small search-tree by selecting effective splitting variables (e.g. via looking ahead).

Strength: Effective on small, hard formulas.

Weakness: Expensive.

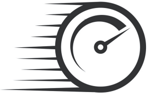
SAT Solver Paradigms Overview



DPLL: Aims at finding a small search-tree by selecting effective splitting variables (e.g. via looking ahead).

Strength: Effective on small, hard formulas.

Weakness: Expensive.

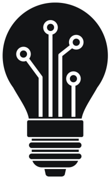


Local search: Given a full assignment for a formula Γ, flip the truth values of variables until satisfying Γ.

Strength: Can quickly find solutions for hard formulas.

Weakness: Cannot prove unsatisfiability.

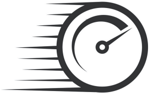
SAT Solver Paradigms Overview



DPLL: Aims at finding a small search-tree by selecting effective splitting variables (e.g. via looking ahead).

Strength: Effective on small, hard formulas.

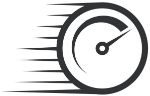
Weakness: Expensive.



Local search: Given a full assignment for a formula Γ, flip the truth values of variables until satisfying Γ.

Strength: Can quickly find solutions for hard formulas.

Weakness: Cannot prove unsatisfiability.



Conflict-driven clause learning (CDCL): Makes fast decisions and converts conflicts into learned clauses.

Strength: Effective on large, "easy" formulas.

Weakness: Hard to parallelize.

Conflict-driven Clause Learning: Overview

- Most successful architecture
- Superior on industrial benchmarks
- Brute-force?
  - Addition conflict clauses
  - Fast unit propagation
- Complete local search (for a refutation)?
- State-of-the-art (sequential) CDCL solvers: Kissat, CaDiCaL, Glucose, CryptoMiniSAT

Clause Learning

Data-structures

Heuristics

Clause Management

Conflict-Clause Minimization

Recent Advances and Conclusions

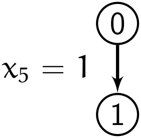
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
Γ_{extra}

0

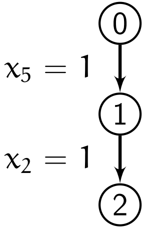
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
Γ_{extra}



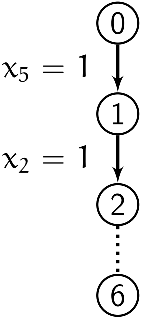
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \lor x_4) \land$
$(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land$
$(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land$
Γ_{extra}



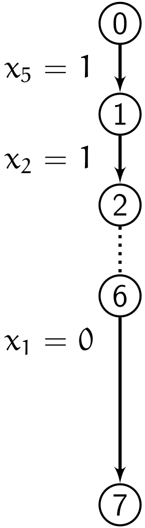
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
Γ_{extra}



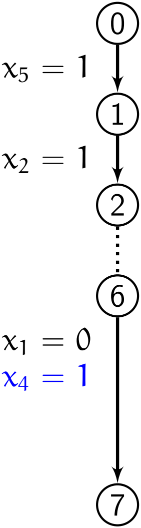
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \lor x_4) \land$
$(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land$
$(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land$
Γ_{extra}



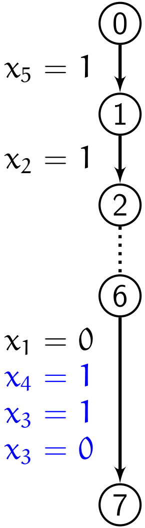
Conflict-driven SAT solvers: Search and Analysis

$$
\begin{aligned}
&(x_1 \lor x_4) \land \\
&(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land \\
&(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land \\
&\Gamma_{\text{extra}}
\end{aligned}
$$



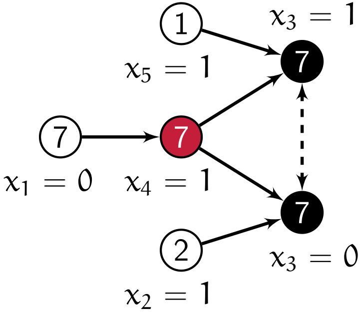
Conflict-driven SAT solvers: Search and Analysis

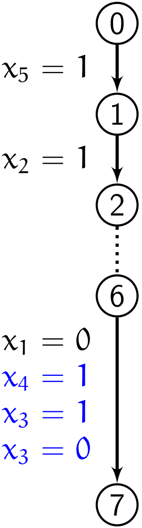
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
Γ_{extra}



Conflict-driven SAT solvers: Search and Analysis

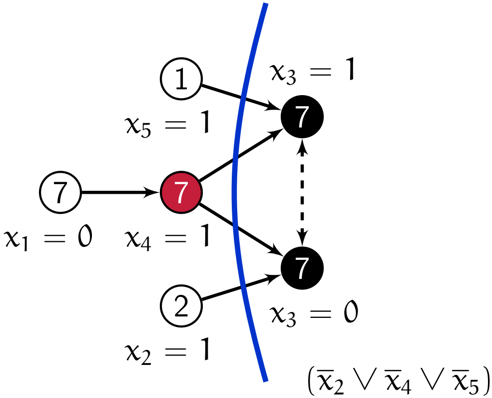
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
Γ_{extra}

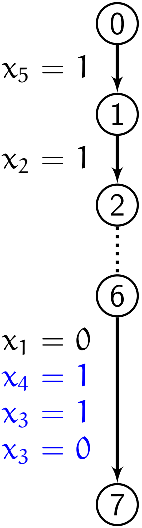




Conflict-driven SAT solvers: Search and Analysis

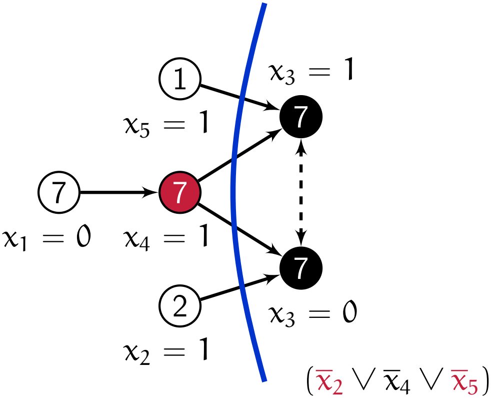
$(x_1 \lor x_4) \land$
$(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land$
$(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land$
Γ_{extra}

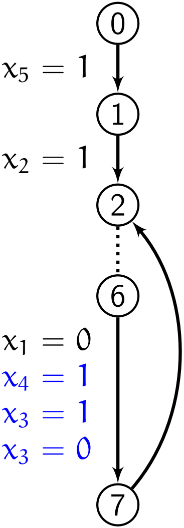




Conflict-driven SAT solvers: Search and Analysis

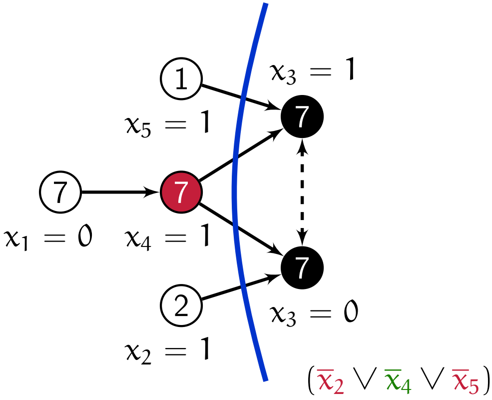
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
Γ_{extra}

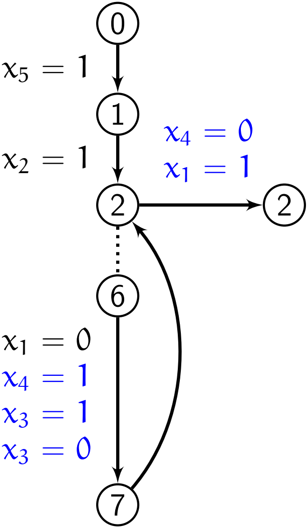




Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
Γ_{extra}

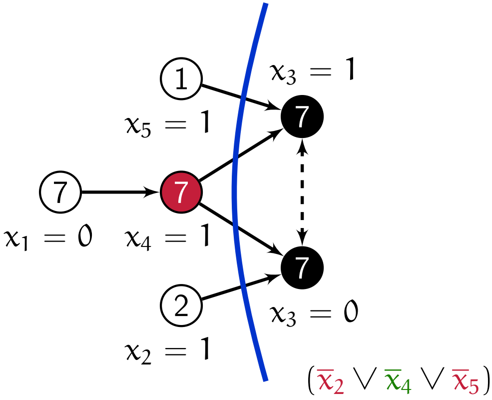


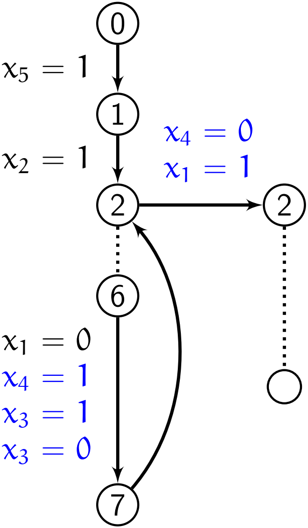


marijn@cmu.edu

Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
Γ_{extra}





Implication graph [Marques-SilvaSakallah '96]

CDCL in a nutshell:

1. Main loop combines efficient problem simplification with cheap, but effective decision heuristics; ($> 90\%$ of time)
2. Reasoning kicks in if the current state is conflicting;
3. The current state is analyzed and turned into a constraint;
4. The constraint is added to the problem, the heuristics are updated, and the algorithm (partially) restarts.

Implication graph [Marques-SilvaSakallah '96]

CDCL in a nutshell:

1. Main loop combines efficient problem simplification with cheap, but effective decision heuristics; ($> 90\%$ of time)
2. Reasoning kicks in if the current state is conflicting;
3. The current state is analyzed and turned into a constraint;
4. The constraint is added to the problem, the heuristics are updated, and the algorithm (partially) restarts.

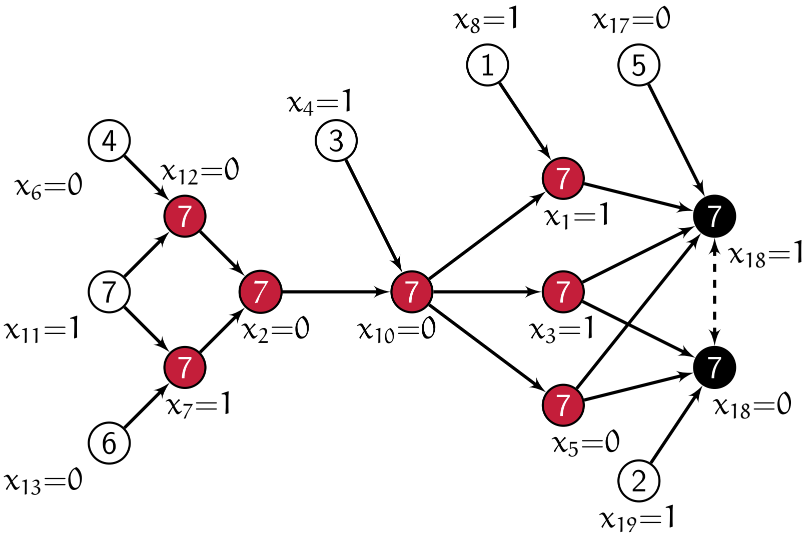
However, it has three weaknesses:

- CDCL is notoriously hard to parallelize;
- the representation impacts CDCL performance; and
- CDCL has exponential runtime on some “simple” problems.

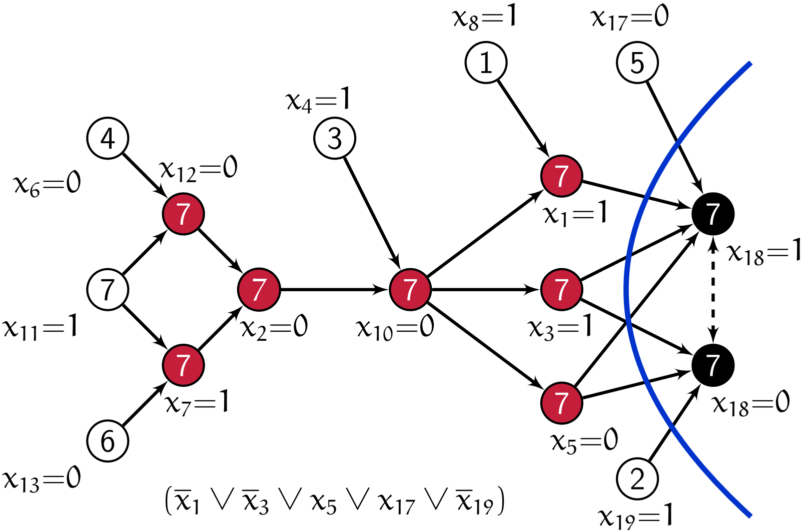
Conflict-driven Clause Learning: Pseudo-code

```
while TRUE do
    l_decision := Decide ()
    If no l_decision then return satisfiable
    Γ := Simplify (Γ(l_decision ← 1))
    while Γ contains C_falsified do
        C_conflict := Analyze (C_falsified)
        If C_conflict = ⊥ then return unsatisfiable
        BackTrack (C_conflict)
        Γ := Simplify (Γ ∪ {C_conflict})
    end while
end while
```

Learning conflict clauses [Marques-SilvaSakallah'96]



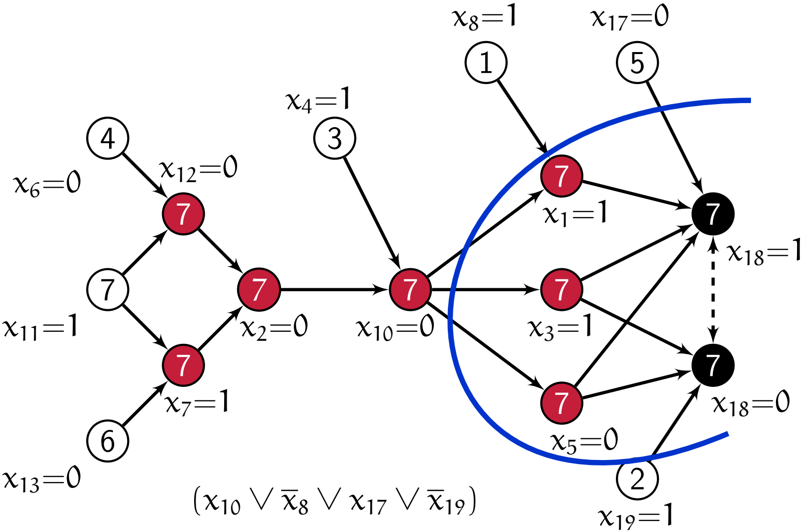
Learning conflict clauses [Marques-SilvaSakallah'96]



$(\overline{x}_1 \vee \overline{x}_3 \vee x_5 \vee x_{17} \vee \overline{x}_{19})$

tri-asserting clause

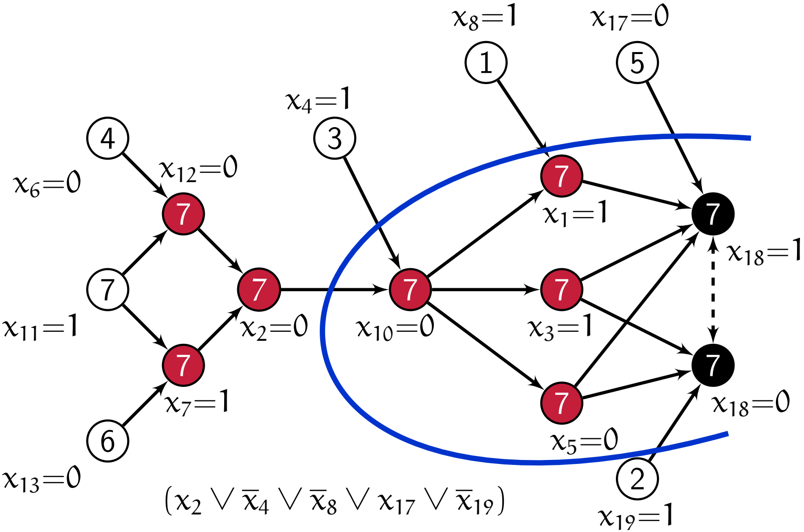
Learning conflict clauses [Marques-SilvaSakallah'96]



$(x_{10} \vee \overline{x}_8 \vee x_{17} \vee \overline{x}_{19})$

first unique implication point

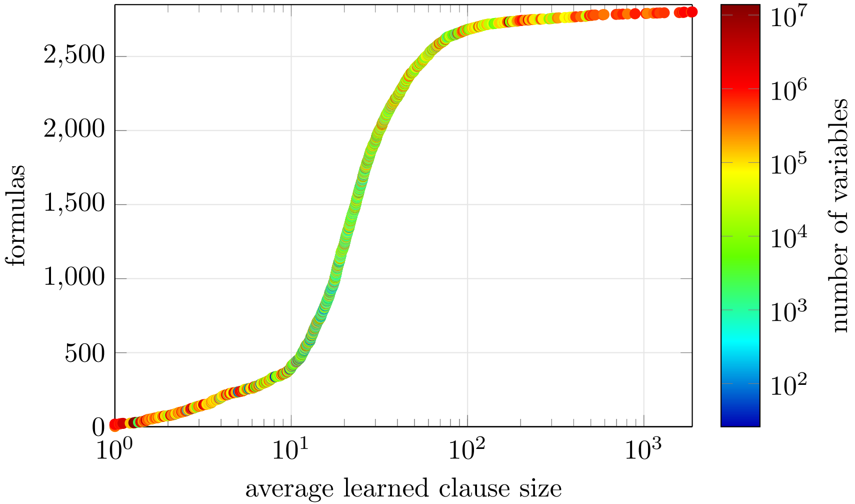
Learning conflict clauses [Marques-SilvaSakallah'96]



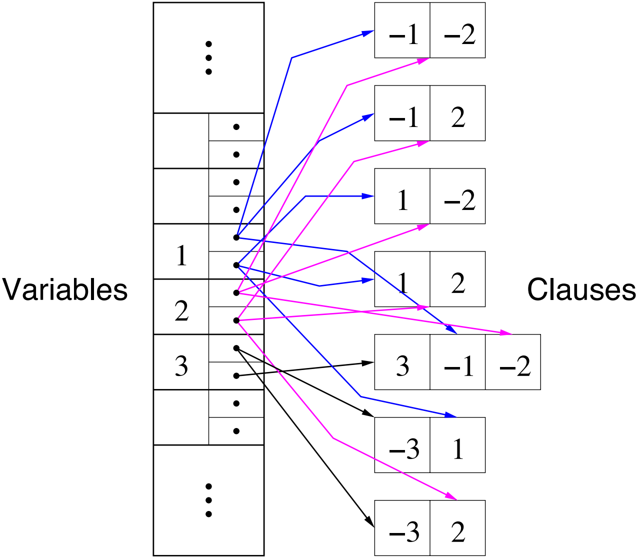
$(x_2 \vee \overline{x}_4 \vee \overline{x}_8 \vee x_{17} \vee \overline{x}_{19})$

second unique implication point

Average Learned Clause Length



Simple data structure for unit propagation



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

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

$\overline{x}_1$	x_2	$\overline{x}_3$	$\overline{x}_5$	x_6

x_1	$\overline{x}_3$	x_4	$\overline{x}_5$	$\overline{x}_6$

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

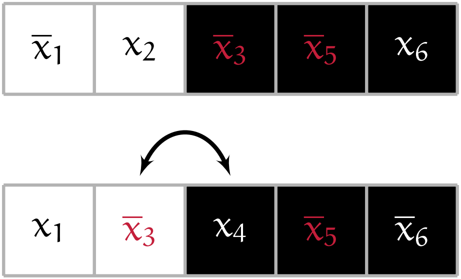
$$\alpha = \{x_1 = *, x_2 = *, x_3 = *, x_4 = *, x_5 = 1, x_6 = *\}$$

$\overline{x}_1$	x_2	$\overline{x}_3$	$\overline{x}_5$	x_6

x_1	$\overline{x}_3$	x_4	$\overline{x}_5$	$\overline{x}_6$

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

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



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

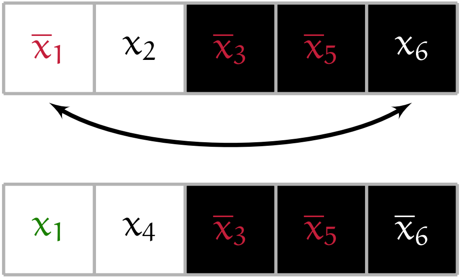
$$\alpha = \{x_1 = *, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = *\}$$

$\bar{x}_1$	x_2	$\bar{x}_3$	$\bar{x}_5$	x_6

x_1	x_4	$\bar{x}_3$	$\bar{x}_5$	$\bar{x}_6$

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

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



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

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

x_6	x_2	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_1$

x_1	x_4	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_6$

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

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

x_6	x_2	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_1$

x_1	x_4	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_6$

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

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

x_6	x_2	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_1$

x_1	x_4	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_6$

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

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

x_6	x_2	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_1$

x_1	x_4	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_6$

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

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

x_6	x_2	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_1$

x_1	x_4	$\overline{x}_3$	$\overline{x}_5$	$\overline{x}_6$

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

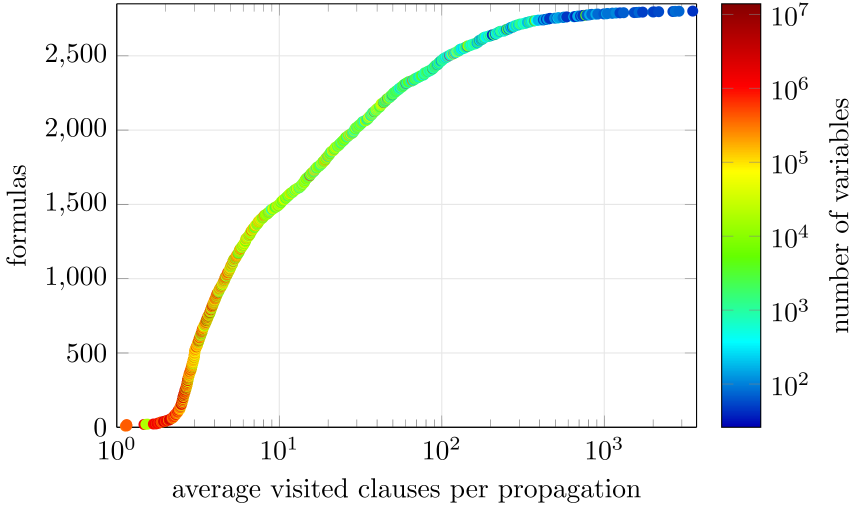
While backjumping, just unassign variables

Conflict clauses $\rightarrow$ watch pointers

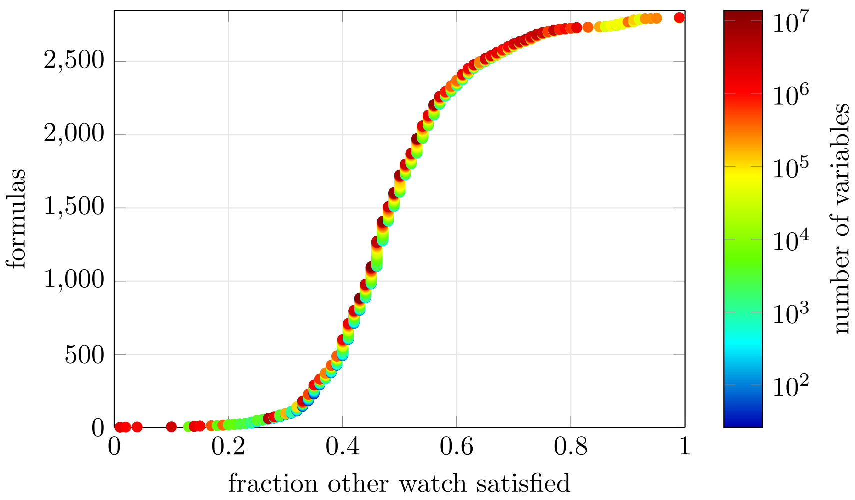
No detailed information available

Not used for binary clauses

Average Number of Clauses Visited Per Propagation



Percentage visited clauses with other watched literal true



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

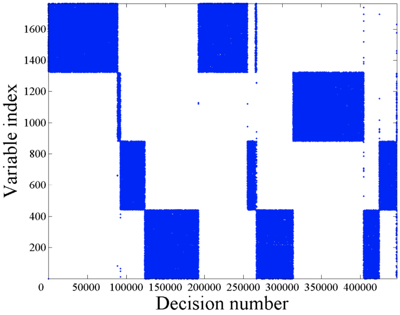
- negative branching (early MiniSAT) [EenSörensson'03]

Based on the last implied value (phase-saving)

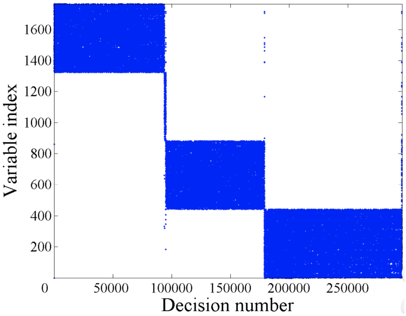
- introduced to CDCL [PipatsrisawatDarwiche'07]
- already used in local search [HirschKojevnikov'01]

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: $1, 1, 2, 1, 1, 2, 4, \ldots$

Heuristics: SAT vs UNSAT [Oh'15]

The best heuristics choices depend on satisfiability: E.g.

- Restart frequently for UNSAT instances to get conflict early
- Restart sporadically for SAT instances to keep "progress"

Also, keeping learned clauses is less important on SAT instances and can actually slow down the search.

State-of-the-art CDCL solvers, such as CaDiCaL, have separate modes for SAT and UNSAT and they alternate between them.

Clause delection [EenSörensson'03, AudemardSimon'09]

Conflict clauses can significantly slow down CDCL solvers:

- Conflict clauses can quickly outnumber the original clauses
- Conflict clauses consists of important variables

Clause deletion is used to reduce the overhead:

- When the learned clause reach a limit, remove half
- Increase limit after every removal (completeness)

Clause delection [EenSörensson'03, AudemardSimon'09]

Conflict clauses can significantly slow down CDCL solvers:

- Conflict clauses can quickly outnumber the original clauses
- Conflict clauses consists of important variables

Clause deletion is used to reduce the overhead:

- When the learned clause reach a limit, remove half
- Increase limit after every removal (completeness)

Clause deletion heuristics:

- length of the clause
- relevance of the clause (when was it used in Analyze)
- the number of involved decision levels

Self-Subsumption

Use self-subsumption to shorten conflict clauses

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

Conflict clause minimization is an important optimization.

Self-Subsumption

Use self-subsumption to shorten conflict clauses

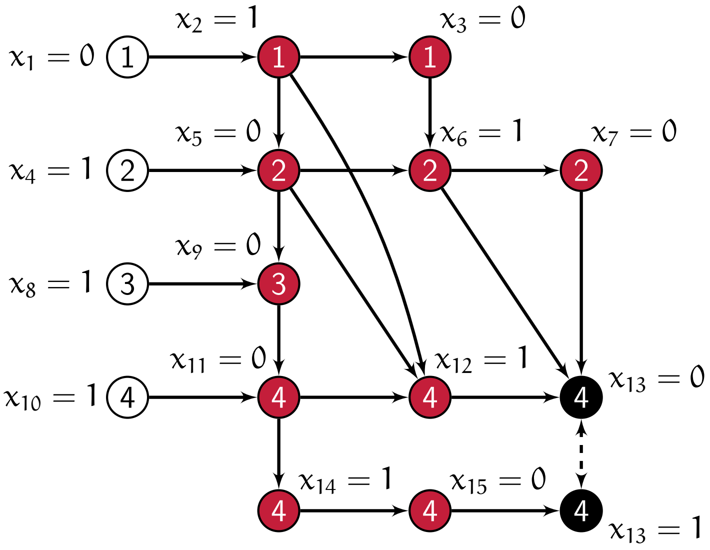
$$\frac{C \vee l \quad D \vee \bar{l}}{D}\; C \subseteq D \qquad \frac{(a \vee b \vee l) \quad (a \vee b \vee c \vee \bar{l})}{(a \vee b \vee c)}$$

Conflict clause minimization is an important optimization.

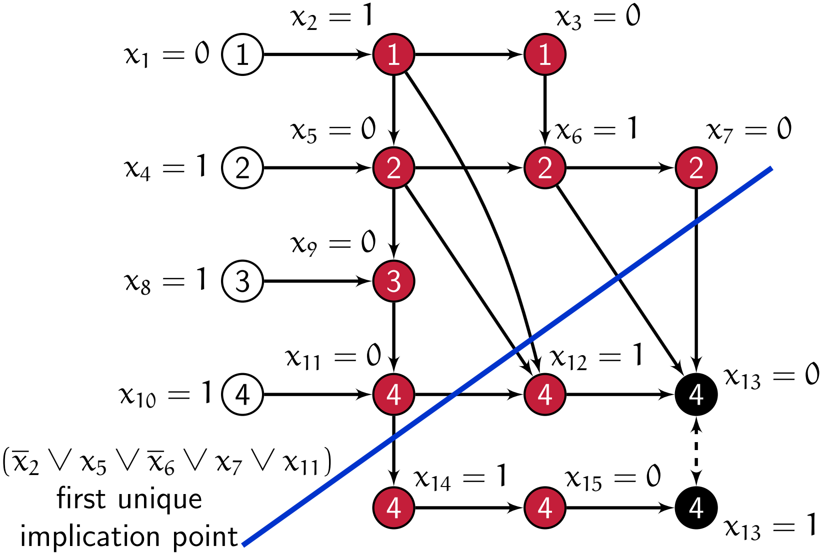
Use implication chains to further minimization:

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

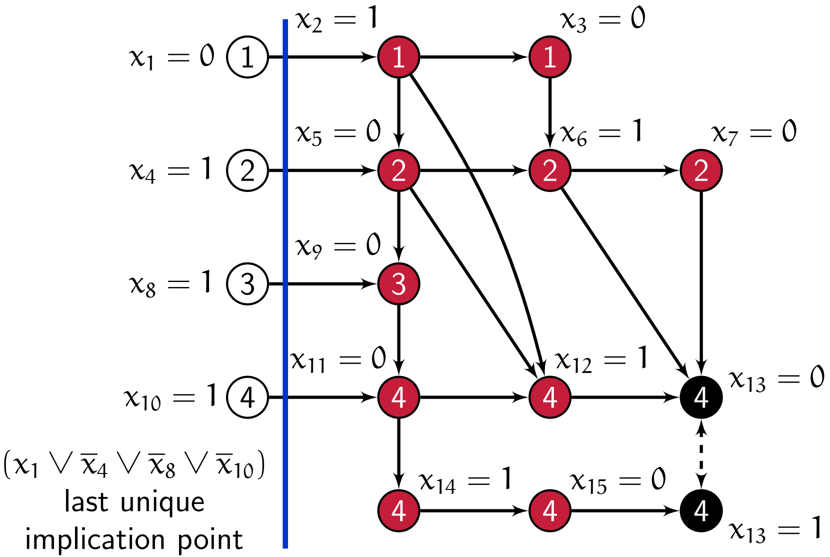
Conflict-clause minimization [SörenssonBiere'09]



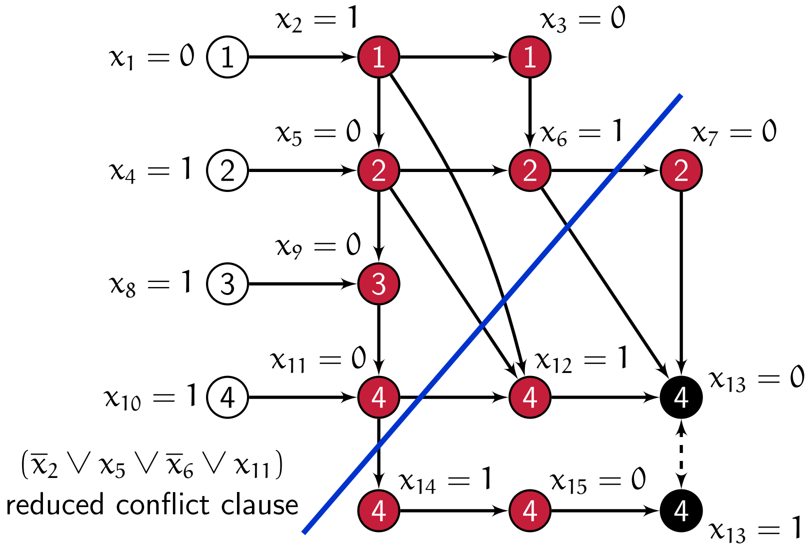
Conflict-clause minimization [SörenssonBiere'09]



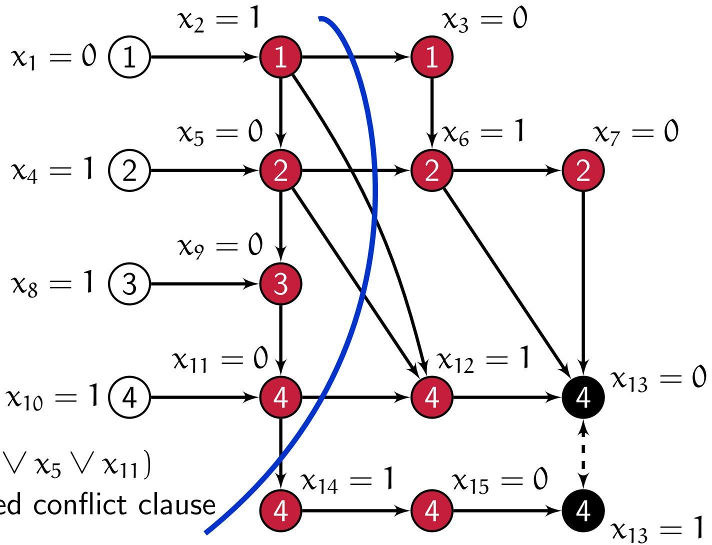
Conflict-clause minimization [SörenssonBiere'09]



Conflict-clause minimization [SörenssonBiere'09]



Conflict-clause minimization [SörenssonBiere'09]



$(\overline{x}_2 \vee x_5 \vee x_{11})$
minimized conflict clause

marijn@cmu.edu

Recent Advances (I)

A new idea contributes to winning the competition.

Winner 2017: Clause vivification during search
[LuoLiXiaoManyáLü'17]

Winner 2018: Chronological backtracking
[NadelRyvchin'18]

Winner 2019: Multiple learnt clauses per conflict
[KochemazovZaikinKondratievSemenov'19]

Winner 2020: Back to C and "target phases"
[BiereFleury'20]

Recent Advances (II)

A new idea contributes to winning the competition.

Winner 2021: Selecting decision heuristics using RL [CherifHabetTerrioux'21]

Winner 2022: Decision tree of local search strategies [ZhengHeChenZhouLi'22]

Winner 2023: Structured Reencoding of Formulas [HaberlandtGreen'23]

The 2023 winning solver was on a course project

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]
- SAT vs UNSAT [Oh'15]

+ Pre- and in-processing techniques