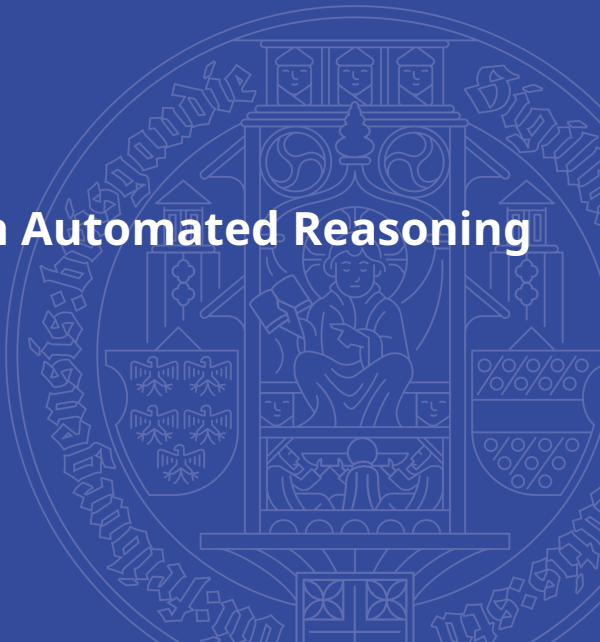
The Role of SAT Solving in Automated Reasoning

KLEE Workshop 2026 aff. ASE'26

Armin Biere | University of Freiburg

Munich | Germany | October 15, 2026

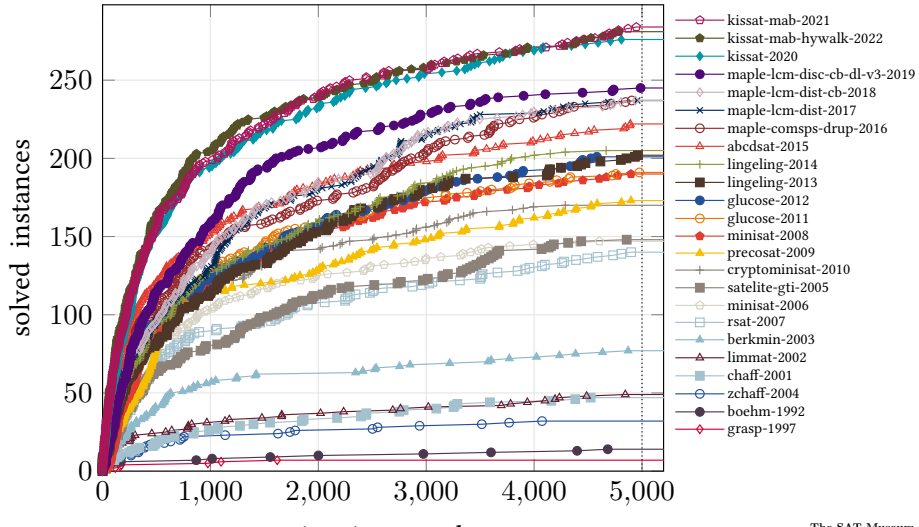


Outline

- SAT solvers get faster and faster!
- How do SAT solvers get faster and faster?
- Why do we care in the age of AI?

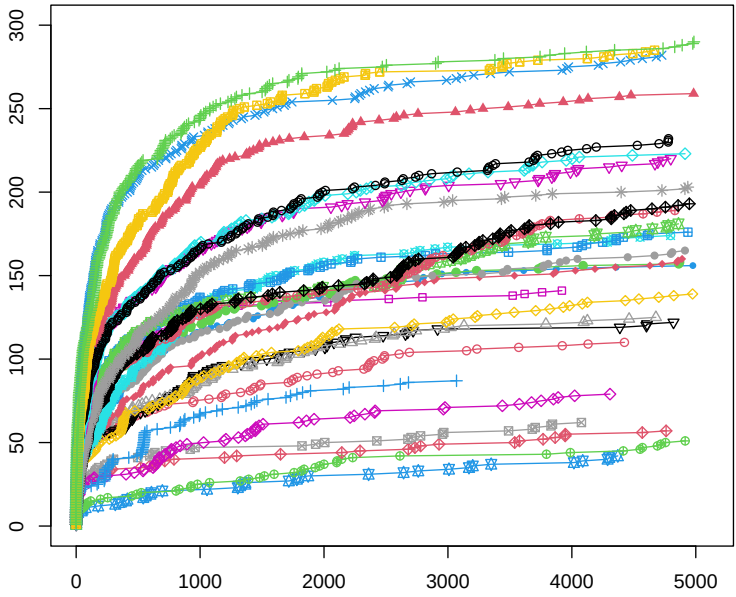
SAT Solvers Get Faster and Faster

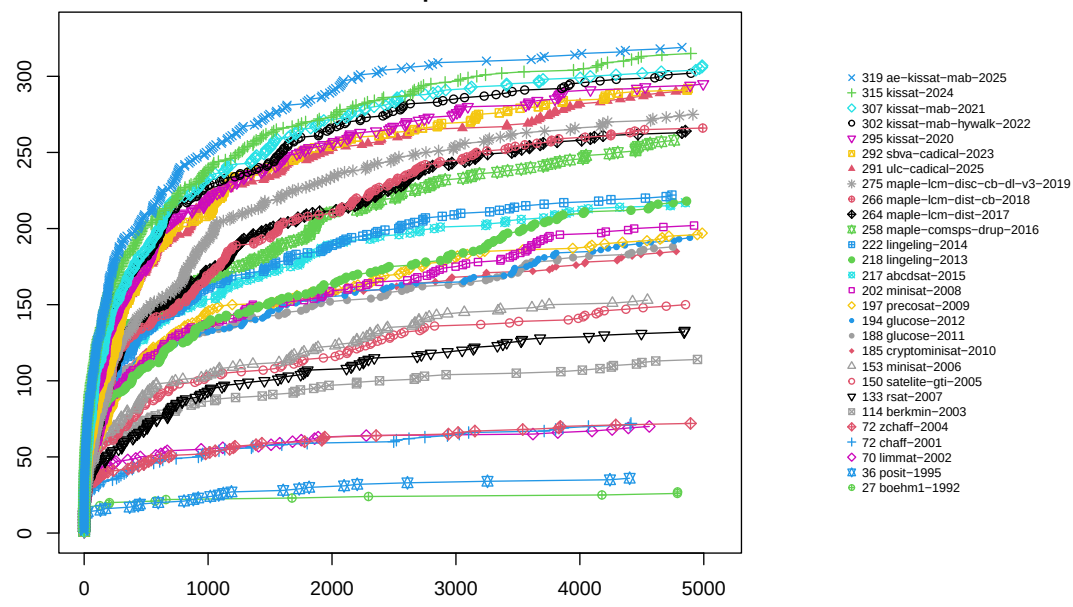
SAT Competition All Time Winners on SAT Competition 2022 Benchmarks

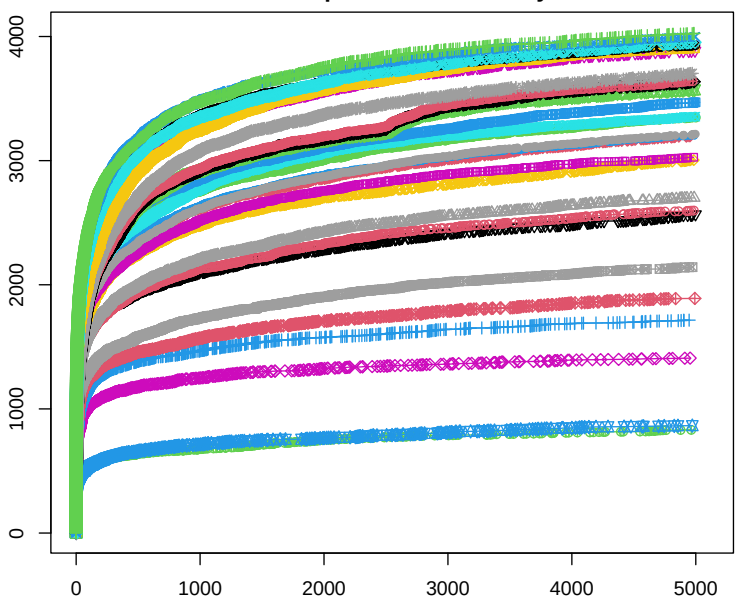


https://cca.informatik.uni-freiburg.de/satmuseum

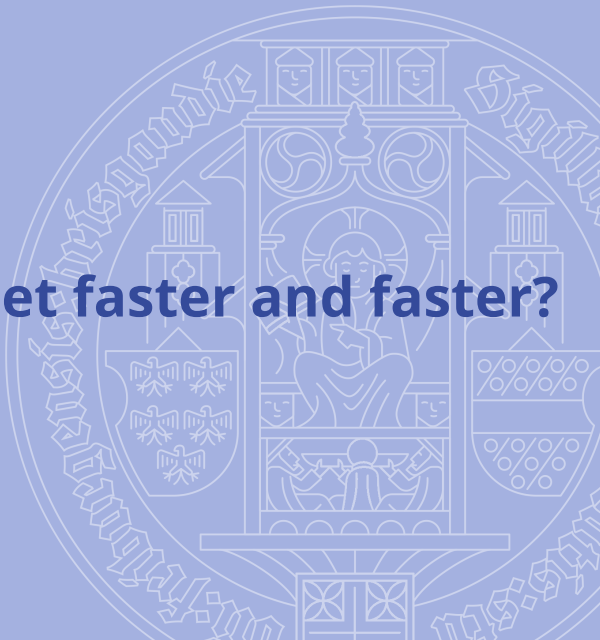
The SAT Museum.
Armin Biere and Mathias Fleury and Nils Froleyks and Marijn J.H. Heule.
In *Proceedings 14th International. Workshop on Pragmatics of SAT (POS'23)*, vol. 3545, CEUR Workshop Proceedings, pages 72-87, CEUR-WS.org 2023.
[paper - bibtex - data - zenodo - ceur - workshop - proceedings]


All Time Winners on SAT Competition 2023 Benchmarks
0
50
100
150
200
250
300
0
1000
2000
3000
4000
5000
290 kissat−2024
285 sbva−cadical−2023
282 ae−kissat−mab−2025
259 ulc−cadical−2025
232 kissat−mab−hywalk−2022
223 kissat−mab−2021
220 kissat−2020
203 maple−lcm−disc−cb−dl−v3−2019
193 maple−lcm−dist−2017
189 maple−lcm−dist−cb−2018
181 maple−comsps−drup−2016
176 lingeling−2014
174 abcdsat−2015
165 glucose−2011
160 cryptominisat−2010
158 lingeling−2013
156 glucose−2012
141 minisat−2008
139 precosat−2009
125 minisat−2006
122 rsat−2007
110 satelite−gti−2005
87 chaff−2001
79 limmat−2002
62 berkmin−2003
57 zchaff−2004
51 boehm1−1992
42 posit−1995

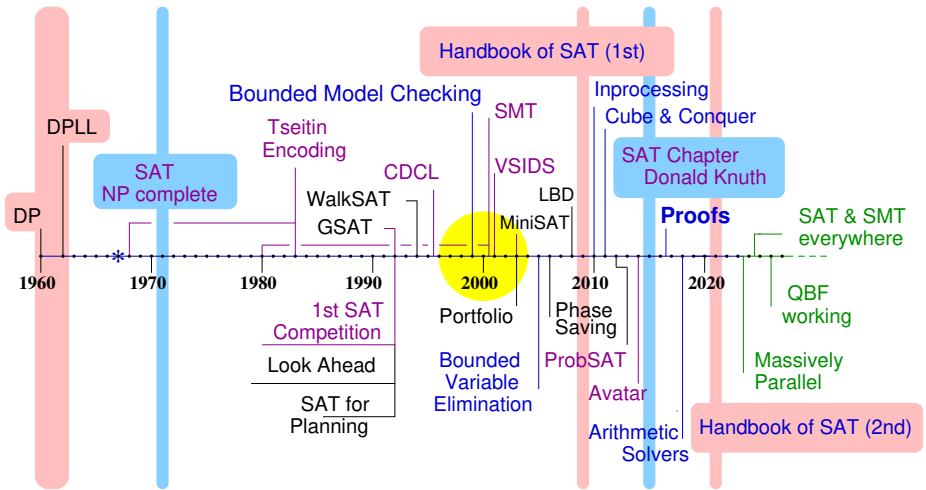

All Time Winners on SAT Competition 2025 Benchmarks
0
50
100
150
200
250
300
0
1000
2000
3000
4000
5000
319 ae−kissat−mab−2025
315 kissat−2024
307 kissat−mab−2021
302 kissat−mab−hywalk−2022
295 kissat−2020
292 sbva−cadical−2023
291 ulc−cadical−2025
275 maple−lcm−disc−cb−dl−v3−2019
266 maple−lcm−dist−cb−2018
264 maple−lcm−dist−2017
258 maple−comsps−drup−2016
222 lingeling−2014
218 lingeling−2013
217 abcdsat−2015
202 minisat−2008
197 precosat−2009
194 glucose−2012
188 glucose−2011
185 cryptominisat−2010
153 minisat−2006
150 satelite−gti−2005
133 rsat−2007
114 berkmin−2003
72 zchaff−2004
72 chaff−2001
70 limmat−2002
36 posit−1995
27 boehm1−1992


Winners on SAT Competition Anniversary Benchmarks
0
1000
2000
3000
4000
0
1000
2000
3000
4000
5000
4025 kissat−2024
3998 ae−kissat−mab−2025
3953 kissat−mab−2021
3930 kissat−mab−hywalk−2022
3906 sbva−cadical−2023
3886 kissat−2020
3703 maple−lcm−disc−cb−dl−v3−2019
3640 maple−lcm−dist−cb−2018
3636 maple−lcm−dist−2017
3570 maple−comsps−drup−2016
3470 lingeling−2014
3354 abcdsat−2015
3347 lingeling−2013
3214 glucose−2011
3199 glucose−2012
3181 cryptominisat−2010
3027 minisat−2008
3004 precosat−2009
2700 minisat−2006
2598 satelite−gti−2005
2568 rsat−2007
2142 berkmin−2003
1890 zchaff−2004
1715 chaff−2001
1409 limmat−2002
865 posit−1995
831 boehm1−1992

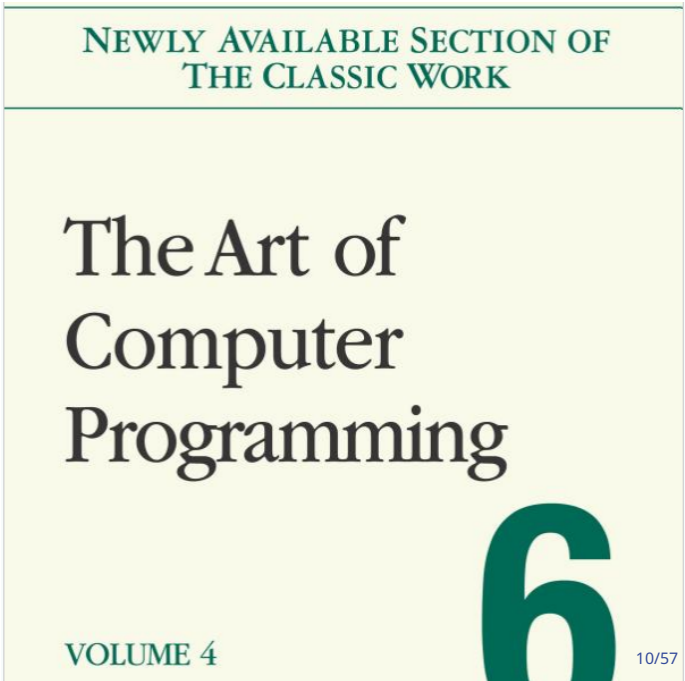
How do SAT solvers get faster and faster?



60 Years of SAT Solving



Frontiers in Artificial Intelligence and Applications
HANDBOOK
of satisfiability
Editors:
Armin Biere
Marijn Heule
Hans van Maaren
Toby Walsh


NEWLY AVAILABLE SECTION OF
THE CLASSIC WORK
The Art of
Computer
Programming
6
VOLUME 4

File Edit View Document Tools Window Help
5 / 318 150% Find

Special thanks are due to Armin Biere, Randy Bryant, Sam Buss, Niklas Eén, Ian Gent, Marijn Heule, Holger Hoos, Svante Janson, Peter Jeavons, Daniel Kroening, Oliver Kullmann, Massimo Lauria, Wes Pegden, Will Shortz, Carsten Sinz, Niklas Sörensson, Udo Wermuth, Ryan Williams, and . . . for their detailed comments on my early attempts at exposition, as well as to numerous other correspondents who have contributed crucial corrections. Thanks also to Stanford's Information Systems Laboratory for providing extra computer power when my laptop machine was inadequate.

Biere
Bryant
Buss
Eén
Gent
Heule
Hoos
Janson
Jeavons
Kroening
Kullmann
Lauria
Pegden
Shortz
Sinz
Sörensson
Wermuth
Williams
Internet
MPR
Internet

* * *

Wow — Section 7.2.2.2 has turned out to be the longest section, by far, in *The Art of Computer Programming*. The SAT problem is evidently a "killer app," because it is key to the solution of so many other problems. Consequently I can only hope that my lengthy treatment does not also kill off my faithful readers! As I wrote this material, one topic always seemed to flow naturally into another, so there was no neat way to break this section up into separate subsections. (And anyway the format of *TAOCP* doesn't allow for a Section 7.2.2.2.1.)

Important Solver Improvements

- bounded variable elimination (2005)
- …
- stable-focused-mode switching (2016)
- vivification on redundant clauses (2019)
- target phases, rephasing (2020)
- bounded variable addition (2023)
- clausal congruence closure (2024)
- some code optimization through machine learning (2025)
- effective symmetry breaking with proofs (2026)

  + (massif) optimization through machine learning (2026)?

Portfolio / Stable-Focused / SAT-UNSAT Mode [Oh-SAT'15]

- Chanseok Oh conjectured and proved empirically:
  - solving *satisfiable* instances is different from
  - solving *unsatisfiable* instances
  - if we do not know whether the formula is satisfiable or not
    run an interleaved portfolio of two different solver configurations
- **SAT** phase tries to solve satisfiable instances faster:
  - benefits from few restarts (surprisingly) and
  - from slower score decay (interesting variables slowly emerge)
  - all-in-all forms more **stable** search behavior
- **UNSAT** phase tries to solve unsatisfiable instances faster:
  - benefits from frequent aggressive restarts (also surprisingly) and
  - from fast score decay (chase variables involved in recent conflicts)
  - all-in-all forms more **focused** search behavior
- introduced in COMSPS solver, became effective in MapleCOMSPS

Lessons from Stable-Focused aka. SAT/UNSAT Mode Switching

- using solving “portfolios” specialized for different outcomes is very powerful
- could be parallel but we use a sequential portfolio though
  (and thus scheduling it is actually non-trivial)
- improving on *satisfiable* instances is much easier

Bounded Variable Elimination (BVE)

[DavisPutnam'58] unbounded
[EénBiere-SAT'05] bounded

Replace $(\boxed{x} \vee a)_1$ $(\boxed{x} \vee b)_2$ $(\boxed{x} \vee c)_3$ $(\boxed{\bar{x}} \vee \bar{a} \vee \bar{b})_4$ $(\boxed{\bar{x}} \vee d)_5$ by $\cancel{(a \vee \bar{a} \vee \bar{b})_{14}}$ $(a \vee d)_{15}$ $\cancel{(b \vee \bar{a} \vee \bar{b})_{24}}$ $(b \vee d)_{25}$ $(c \vee \bar{a} \vee \bar{b})_{34}$ $(c \vee d)_{45}$

- number of clauses *not increasing*
- strengthen and remove subsumed clauses too
- most important and most effective preprocessing we have

Bounded Variable Addition (BVA)

[MantheyHeuleBiere-HVC'12] original
[HaberlandtGreenHeule-SAT'23] revisited
[PollittBattlemanFleuryVizelHeuleBiereBryant-SAT'26] XORs

Replace $(d \vee a)$ $(e \vee a)$ $(d \vee b)$ $(e \vee b)$ $(d \vee c)$ $(e \vee c)$ by $(\boxed{x} \vee a)$ $(d \vee \boxed{\bar{x}})$ $(\boxed{x} \vee b)$ $(e \vee \boxed{\bar{x}})$ $(\boxed{x} \vee c)$

- number of clauses has to *decrease strictly* when combined with BVE
- reencodes for instance naive at-most-one constraint encodings
- actually an interesting form of extended resolution

Full Factorization through Bounded Variable Addition

```
c full1
p cnf 5 6
1 3 0 2 3 0
1 4 0 2 4 0
1 5 0 2 5 0
```

```
c full2
p cnf 6 9
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0
1 6 0 2 6 0 3 6 0
```

```
c full3
p cnf 7 12
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0
1 7 0 2 7 0 3 7 0 4 7 0
```

```
c full1factored
p cnf 6 5
6 3 0 1 -6 0
6 4 0 2 -6 0
6 5 0
```

```
c full2factored
p cnf 7 6
7 4 0 1 -7 0
7 5 0 2 -7 0
7 6 0 3 -7 0
```

```
c full3factored
p cnf 8 7
8 5 0 1 -8 0
8 6 0 2 -8 0
8 7 0 3 -8 0
      4 -8 0
```

```
#1 clause less
```

```
#3 clauses less
```

```
#5 clauses less
```

BVA $\approx$ Algebraic Division

Synthesis concept on DNF (polynomials / SOP)

Turns two-level circuit into three-level circuit
(reduces area / increases delay)

$$abc + bcd + ce \;=\; bc(a+d) + ce \;=\; c(ab + bd + e)$$

Dual versions (DNF and CNF) of our example in this context:

$$\begin{aligned}
da + db + dc + ea + eb + ec &= \underbrace{(a+b+c)}_{\text{quotients}} \cdot^{x} \underbrace{(d+e)}_{\text{factors}} \\
&= \forall x[xa + xb + xc + d\bar{x} + e\bar{x}] \\
(d \vee a)(d \vee b)(d \vee c)(e \vee a)(e \vee b)(e \vee c) &= \underbrace{abc}_{\text{quotients}} \vee_{x} \underbrace{de}_{\text{factors}} \\
&= \exists x[(x \vee a)(x \vee b)(x \vee c)(d \vee \bar{x})(e \vee \bar{x})]
\end{aligned}$$

Factorization of Naive At-Most-One Constraint Encodings

```
c amo4
p cnf 4 6
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
```

```
c amo5a
p cnf 5 10
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
```

```
c factoredamo4
p cnf 5 6
1 2 0
5 3 0 1 -5 0
5 4 0 2 -5 0 3 4 0
```

```
c factoredamo5a
p cnf 6 9
1 2 0
6 3 0 1 -6 0
6 4 0 2 -6 0 3 4 0
6 5 0 3 5 0 4 5 0
```

```
#0 clauses reduced
```

```
#1 clause reduced
```

Two Factorizations of Naive At-Most-One Constraint Encodings

```
c amo5a
p cnf 5 10
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
```

```
c amo5b
p cnf 5 10
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
```

```
c factoredamo5a
p cnf 6 9
1 2 0
6 3 0 1 -6 0
6 4 0 2 -6 0 3 4 0
6 5 0 3 5 0 4 5 0

#1 clause reduced
```

```
c factoredamo5b
p cnf 6 9
1 2 0
1 3 0 2 3 0
6 4 0 1 -6 0
6 5 0 2 -6 0 4 5 0
3 -6 0

#1 clause reduced
```

First Factor in At-Most-One Constraints over 9 Variables

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Search for first factor $\boxed{1}$ with most occurrences.

Ties can be broken in different ways (smallest one here).

Quotients are all other literals $2 \ldots 9$

Let Q = #quotients, F = #factors, then reduction $\boxed{Q \cdot F - Q - F} = 8 \cdot 1 - 8 - 1 = -\mathbf{1}$

Two Factors in At-Most-One Constraints over 9 Variables

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Next factor $\boxed{2}$ with most common occurrences in previous quotients.

Ties can be broken in different ways (smallest one here).

Reduce quotients to common ones $3 \ldots 9$.

Reduction $Q \cdot F - Q - F = 7 \cdot 2 - 7 - 2 = 14 - 9 = \mathbf{5}$

Three Factors in At-Most-One Constraints over 9 Variables

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Next factor $\boxed{3}$ with most common occurrences in previous quotients.

Ties can be broken in different ways (smallest one here).

Reduce quotients to common ones $4 \ldots 9$.

Reduction $Q \cdot F - Q - F = 6 \cdot 3 - 6 - 3 = 18 - 9 = \mathbf{9}$

Four Factors in At-Most-One Constraints over 9 Variables

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Next factor $\boxed{4}$ with most common occurrences in previous quotients.

Ties can be broken in different ways (smallest one here).

Reduce quotients to common ones $5 \ldots 9$.

Reduction $Q \cdot F - Q - F = 5 \cdot 4 - 5 - 4 = 20 - 9 = \mathbf{11}$

Five Factors in At-Most-One Constraints over 9 Variables

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Next factor $\boxed{5}$ with most common occurrences in previous quotients.

Ties can be broken in different ways (smallest one here).

Reduce quotients to common ones $6 \ldots 9$.

Reduction $Q \cdot F - Q - F = 4 \cdot 5 - 4 - 5 = 20 - 9 = \mathbf{11}$

Six Factors in At-Most-One Constraints over 9 Variables

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Next factor $\boxed{6}$ with most common occurrences in previous quotients.

Ties can be broken in different ways (smallest one here).

Reduce quotients to common ones $7 \ldots 9$.

Reduction $Q \cdot F - Q - F = 3 \cdot 6 - 3 - 6 = 18 - 9 = \mathbf{9}$

Seven Factors in At-Most-One Constraints over 9 Variables

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Next factor $\boxed{7}$ with most common occurrences in previous quotients.

Ties can be broken in different ways (smallest one here).

Reduce quotients to the two remaining common ones 8, 9.

Reduction $Q \cdot F - Q - F = 2 \cdot 7 - 2 - 7 = 14 - 9 = \mathbf{5}$

Result of Maximum Reduction of 11 Clauses with Four Factors

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 10 25
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
10 5 0
10 6 0 1 -10 0          5 6 0
10 7 0 2 -10 0          5 7 0 6 7 0
10 8 0 3 -10 0          5 8 0 6 8 0 7 8 0
10 9 0 4 -10 0                  5 9 0 6 9 0 7 9 0 8 9 0
```

Tetris

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 10 25
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
10 5 0
10 6 0 1 -10 0 5 6 0
10 7 0 2 -10 0 5 7 0 6 7 0
10 8 0 3 -10 0 5 8 0 6 8 0 7 8 0
10 9 0 4 -10 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Continue Factorization?

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 10 25
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
10 5 0
10 6 0 1 -10 0 5 6 0
10 7 0 2 -10 0 5 7 0 6 7 0
10 8 0 3 -10 0 5 8 0 6 8 0 7 8 0
10 9 0 4 -10 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Maximum Second Factorization with Reduction $3 \cdot 3 - 3 - 3 = \mathbf{3}$

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 10 25
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
10 5 0
10 6 0 1 -10 0 5 6 0
10 7 0 2 -10 0 5 7 0 6 7 0
10 8 0 3 -10 0 5 8 0 6 8 0 7 8 0
10 9 0 4 -10 0 5 9 0 6 9 0 7 9 0 8 9 0
```

Result of Maximum Second Factorization

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 11 22
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
10 5 0
10 6 0 1 -10 0 5 6 0
11 7 0 2 -10 0 10 -11 0
11 8 0 3 -10 0 5 -11 0      7 8 0
11 9 0 4 -10 0 6 -11 0      7 9 0 8 9 0
```

Tetris

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 11 22
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
10 5 0
10 6 0 1 -10 0 5 6 0
11 7 0 2 -10 0 10 -11 0
11 8 0 3 -10 0 5 -11 0 7 8 0
11 9 0 4 -10 0 6 -11 0 7 9 0 8 9 0
```

Continue Factorization the Third Time?

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 11 22
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
10 5 0
10 6 0 1 -10 0 5 6 0
11 7 0 2 -10 0 10 -11 0
11 8 0 3 -10 0 5 -11 0 7 8 0
11 9 0 4 -10 0 6 -11 0 7 9 0 8 9 0
```

Maximum Reduction in Third Factorization of $2 \cdot 3 - 2 - 3 = \mathbf{1}$

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 11 22
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
10 5 0
10 6 0 1 -10 0 5 6 0
11 7 0 2 -10 0 10 -11 0
11 8 0 3 -10 0 5 -11 0 7 8 0
11 9 0 4 -10 0 6 -11 0 7 9 0 8 9 0
```

Result of Third Factorization

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 12 21
1 2 0
12 3 0    1 -12 0
12 4 0    2 -12 0 3 4 0
10 5 0 -10 -12 0
10 6 0 1 -10 0 5 6 0
11 7 0 2 -10 0 10 -11 0
11 8 0          5 -11 0 7 8 0
11 9 0          6 -11 0 7 9 0 8 9 0
```

Tetris

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 12 21
1 2 0
12 3 0 1 -12 0
12 4 0 2 -12 0 3 4 0
10 5 0 -10 -12 0
10 6 0 1 -10 0 5 6 0
11 7 0 2 -10 0 10 -11 0
11 8 0 5 -11 0 7 8 0
11 9 0 6 -11 0 7 9 0 8 9 0
```

BVA Adds 3 Variables and Reduces Formula by 15 Clauses

```
p cnf 9 36
1 2 0
1 3 0 2 3 0
1 4 0 2 4 0 3 4 0
1 5 0 2 5 0 3 5 0 4 5 0
1 6 0 2 6 0 3 6 0 4 6 0 5 6 0
1 7 0 2 7 0 3 7 0 4 7 0 5 7 0 6 7 0
1 8 0 2 8 0 3 8 0 4 8 0 5 8 0 6 8 0 7 8 0
1 9 0 2 9 0 3 9 0 4 9 0 5 9 0 6 9 0 7 9 0 8 9 0
```

```
p cnf 12 21
1 2 0
12 3 0 1 -12 0
12 4 0 2 -12 0 3 4 0
10 5 0 -10 -12 0
10 6 0 1 -10 0 5 6 0
11 7 0 2 -10 0 10 -11 0
11 8 0 5 -11 0 7 8 0
11 9 0 6 -11 0 7 9 0 8 9 0
```

Lessons from Bounded Variable Addition (BVA)

- simple syntactic reasoning has large semantic effects
- BVA was invented to do “extended resolution” in practice
- competition driven research program works

Target Phases / Rephasing / Local-Search [CaiZhangFleuryBiere-JAIR'22]

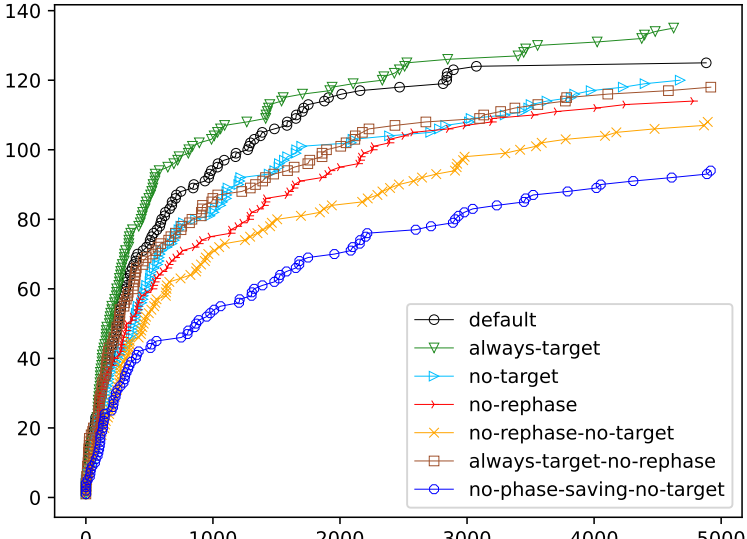
Target Phases

- idea: maximize "consistent" trail (motivated by "blocking restarts" on progress)
- save maximum consistent trail as *phase* (value) assigned to a variable and after picking a decision variable assign it to this *target phase*
- mostly useful in "stable mode" focusing on satisfiable instances
- earlier work (*phase-saving*) reused the last saved assigned value

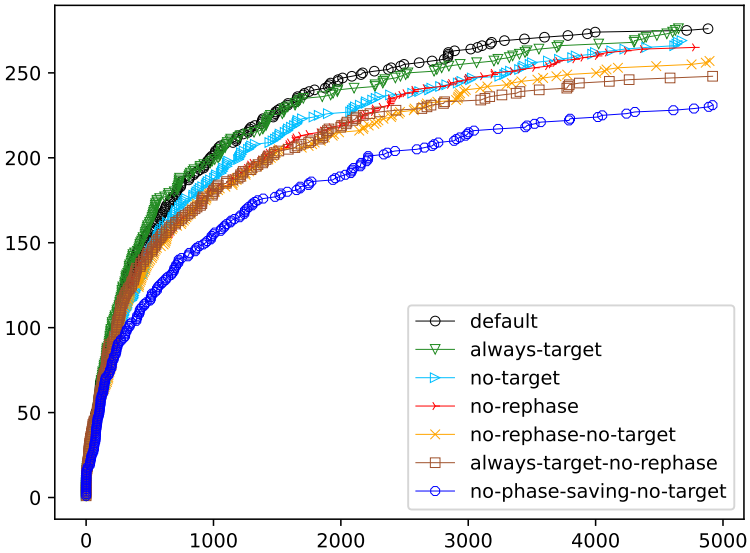
Rephasing

- complements the *intensification* technique of target phases with the (obvious) *diversification* technique of *resetting phases*:
  - set to constant *original* or *inverted original* phase, or
  - the *cached best* (save best target phases on rephasing), or
  - phases are minimized through *local-search*

Kissat on Satisfiable SAT Competition 2021 Main Track Instances



Kissat on All SAT Competition 2021 Main Track Instances



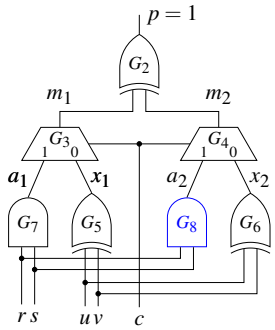
Lessons from Target Phases

- motivated by the challenge of “make local search work” (on industrial instances)
- improving on *satisfiable* instances is much easier
- competition driven research

Circuit Equivalence Checking — Isomorphic Miter

$$p \underset{1}{=} 1$$
$$p \underset{2}{=} m_1 \oplus m_2$$
$$m_1 \underset{3}{=} c \,?\, a_1 : x_1$$
$$m_2 \underset{4}{=} c \,?\, a_2 : x_2$$
$$x_1 \underset{5}{=} u \oplus v$$
$$x_2 \underset{6}{=} u \oplus v$$
$$a_1 \underset{7}{=} r \wedge s$$
$$a_2 \underset{8}{=} r \wedge s$$

(a) gates $G_1, \ldots, G_8$



(b) miter circuit

$$(p)_1$$
$$(\overline{p}\, m_1 m_2)_2\,(\overline{p}\,\overline{m}_1\overline{m}_2)_3\,(p\,\overline{m}_1 m_2)_4\,(p\, m_1\overline{m}_2)_5$$
$$(\overline{m}_1 \overline{c}\, a_1)_6\,(\overline{m}_1 c x_1)_7\,(m_1\overline{c}\,\overline{a}_1)_8\,(m_1 c \overline{x}_1)_9$$
$$(\overline{m}_2 \overline{c}\, a_2)_{10}\,(\overline{m}_2 c x_2)_{11}\,(m_2\overline{c}\,\overline{a}_2)_{12}\,(m_2 c \overline{x}_2)_{13}$$
$$(\overline{x}_1\overline{u}\,\overline{v})_{14}\,(\overline{x}_1 u v)_{15}\,(x_1\overline{u}\,v)_{16}\,(x_1 u\overline{v})_{17}$$
$$(\overline{x}_2\overline{u}\,\overline{v})_{18}\,(\overline{x}_2 u v)_{19}\,(x_2\overline{u}\,v)_{20}\,(x_2 u\overline{v})_{21}$$
$$(\overline{a}_1 r)_{22}\,(\overline{a}_1 s)_{23}\,(a_1\overline{r}\,\overline{s})_{24}$$
$$(\overline{a}_2 r)_{25}\,(\overline{a}_2 s)_{26}\,(a_2\overline{r}\,\overline{s})_{27}$$

(c) CNF with clauses $C_1, \ldots, C_{27}$

XOR of the output of two (isomorphic) circuits is constant 0

Miter circuit assumes that the outputs can be different (XOR is 1)

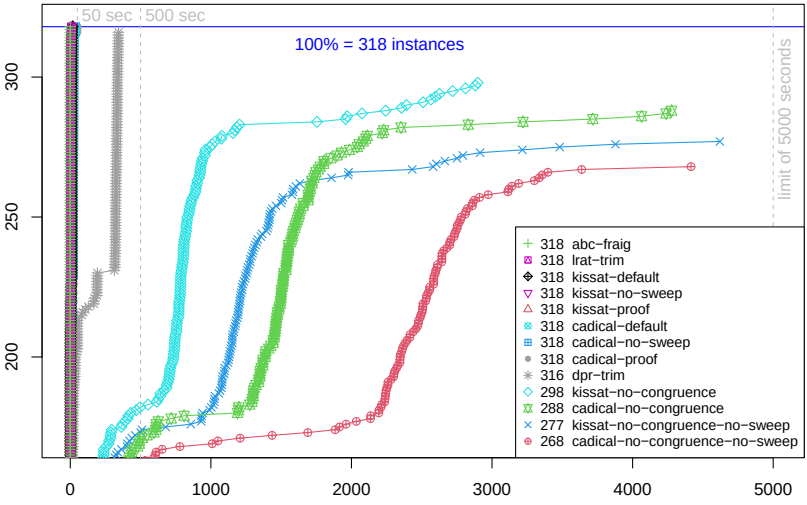
Thus Tseitin Encoding into CNF of the miter is unsatisfiable

- simulates "structural hashing" on the CNF level

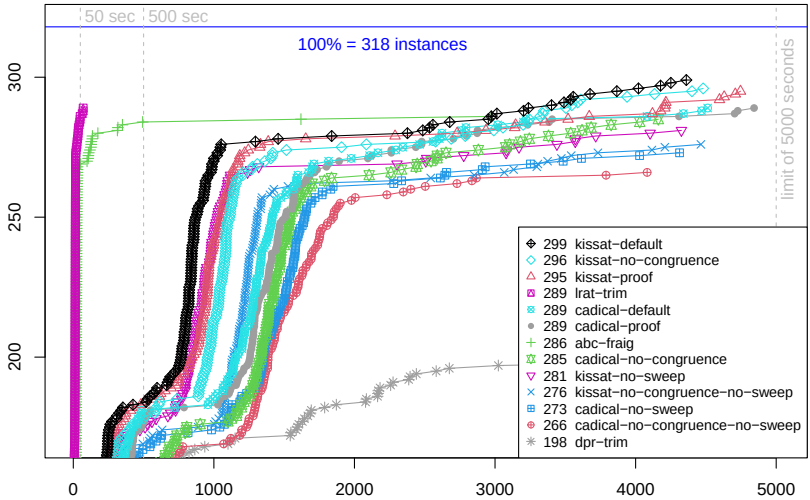
$$\frac{(\bar{x} \vee a)(\bar{x} \vee b)(x \vee \bar{a} \vee \bar{b}) \qquad (\bar{y} \vee a)(\bar{y} \vee b)(y \vee \bar{a} \vee \bar{b})}{(\bar{x} \vee y) \quad (x \vee \bar{y})}$$

- key technique is to **extract gates** (AND, XOR, ITE) efficiently
  (until completion for millions of clauses initially and during inprocessing)
- many real-life SAT encodings have such redundancy initially
  - particularly important for hardware equivalence checking
    (our motivation for more than two decades)
  - observed similar reduction for software model checking instances
    (CBMC instances from Amazon in the competition)
- this redundancy also emerges during solving (inprocessing)
- nicely combines with "clausal equivalence sweeping"
  (using a sub-solver, e.g., Kitten) [BiereFazekasFleuryFrolekys-FMCAD'24]
- actually competetive with dedicated algorithms (ABC)
- ported to CaDiCaL 3.0 [PollittFleuryFazekasFroleyksSchidlerSchreiberBiere-SAT'26]

Hardware Model Checking Competition 2024 Isomorphic Miters



Hardware Model Checking Competition 2024 Optimized Miters



Lessons from Clausal Congruence Closure

- idea is old: gate extraction + structural hashing (congruence closure)
- relaxation of the problem: circuits vs. cyclic constraints
- key are improved algorithms for extraction

however it turns out running until completion can be harmful
(disabled/delayed for industrial users with expected short solving time)

The Results of SAT Competition 2026

Katalin Fazekas, Marijn Heule, **Ashlin Iser**

SAT 2026 Conference, Lisbon (Portugal)

July 23, 2026

Competition Anniversary

Anniversary of SAT Competitions

- 3 competitions in the 90s (1992,1993, 1996)
- 18 SAT Competitions (2002–)
- 5 SAT Races (2006, 2008, 2010, 2015, 2019)
- 1 SAT Challenge (2012)

Goals

- Promotion of SAT solvers and their development
- Compilation of new challenging benchmarks
- Evaluation of current state-of-the-art solvers

News

This Year

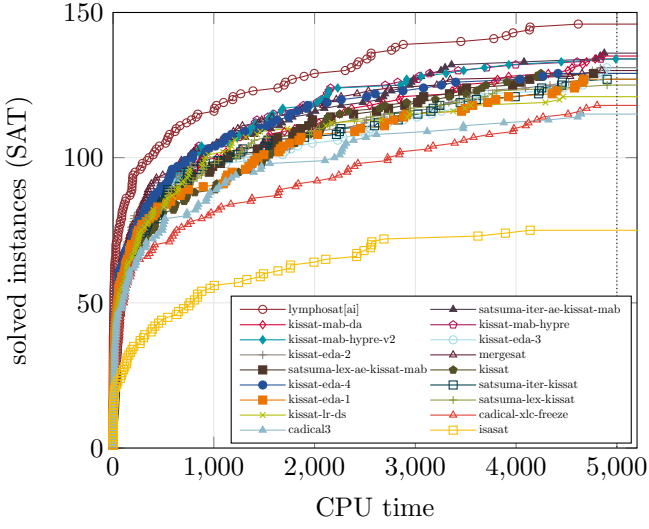
- Benchmark selection script released before submission deadline
  - Seed derived from hash of submitted instances
  - Even organizers cannot predict which instances will be selected
- Special Subtracks for AI Generated / AI Tuned Solvers
  - Need to outperform the best solver in the category to receive an award
- No-Limits track renamed to Experimental Track
- First year sequential track ran on NHR infrastructure at KIT
  - absence of memory issues that had plagued the previous competition

Main Track SAT

Winning Solvers

	Solver	Authors	PAR-2	Solved
AI	Lymphosat	Harrison Green	2813.50	146
1	satsuma-iter+ae-kissat-mab	Markus Anders, Cayden Codel	3430.95	136
2	Kissat-MAB-HyPre	Jiongzhi Zheng et al.	3445.92	134
3	AE-kissat-HUCB	Jianmin Zhang et al.	3507.77	133

Main Track SAT



Discussion Points

Lymphosat uses over 100 family-specific LLM-generated solvers using mostly specialized preprossing (including proofs).

Discussion Point 1: AI Generated / AI Tuned Solvers

- Instance data acts as long-term memory for solvers across runs
  - Mirrors how humans build on past insight (not inherently a problem)
  - Natural consequence of generative AI / data-driven heuristics
- Real challenge: measuring generalization beyond the training database, not just memorization
- Possible mitigations
  - Avoid reusing structurally similar instances in the AI track?
  - Is it a problem at all if solvers recognize a family?

Code Optimization of (SAT) Solvers with LLMs

- initial attempts two years ago lead to small “interesting” patches
- more ambitious projects last year claimed much more without proving checkable experimental evidence
- this year (2026) there are less succesful attempts for SMT
- SAT competition AI track saw AI solver excelling on *satisfiable* instances
- **crucial is that SAT solvers have machine checkable models / proofs**

$\Rightarrow$ Certificates (Models / Proofs)

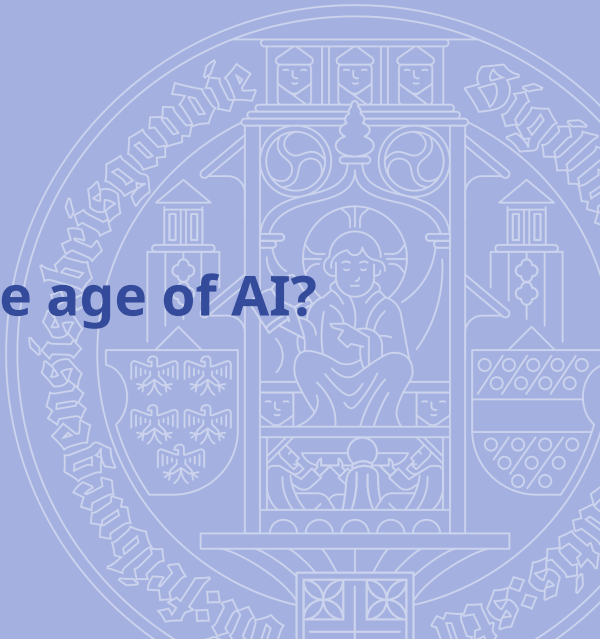
```
$ pwd
/Users/biere/satmuseum/solvers/ae-kissat-mab-2025/AE_kissat2025_MAB
$ git log | head -32
commit da40eb952502cc1b5e0a262a8a400d492584c4d8
Author: FSQH-dh <108135087+FSQH-dh@users.noreply.github.com>
Date:   Wed May 7 20:02:08 2025 +0800

    Update restart.c

    2025 sat competition

commit a518dc22d0b259030be749f4076fd6d60b8f7118
Author: cu909o <1523788588@qq.com>
Date:   Tue May 6 21:31:18 2025 +0800

    first commit

commit ed1c7976127a414f0bb0ebf4795f9b074bdbb24a
Author: seonwee <seonwee@qq.com>
Date:   Wed Apr 30 11:29:02 2025 +0800

    Add mab_conflicts tracking and update restart_mab logic

commit a37db7d62c5244d26789548d1d9362b94aab445b
Author: daweiguo <daweigu0@qq.com>
Date:   Mon Apr 28 20:50:15 2025 +0800

    增加chb分支策略 添加mab策略

commit f7e57764c19cf35198830575492dc9d80e872b55
Author: Armin Biere <armin.biere@gmail.com>
Date:   Thu Jan 23 20:33:16 2025 +0100

    bumped minor version number

$
```

```
Author: seonwee <seonwee@qq.com>
Date:   Wed Apr 30 11:29:02 2025 +0800

    Add mab_conflicts tracking and update restart_mab logic

commit a37db7d62c5244d26789548d1d9362b94aab445b
Author: daweiguo <daweigu0@qq.com>
Date:   Mon Apr 28 20:50:15 2025 +0800

    增加chb分支策略 添加mab策略

commit f7e57764c19cf35198830575492dc9d80e872b55
Author: Armin Biere <armin.biere@gmail.com>
Date:   Thu Jan 23 20:33:16 2025 +0100

    bumped minor version number

$ git show HEAD~~ | tail -20
diff --git a/src/restart.c b/src/restart.c
index 11aea52..35aab5d 100644
--- a/src/restart.c
+++ b/src/restart.c
@@ -110,12 +110,13 @@ static unsigned reuse_trail (kissat *solver) {
   return res;
 }

-void restart_mab(kissat * solver){
+void restart_mab(kissat * solver){
        unsigned stable_restarts = 0;
-       solver->mab_reward[solver->heuristic] += !solver->mab_chosen_tot?0:log2(solver->mab_decisions)/solver->mab_chosen_tot;
+       solver->mab_reward[solver->heuristic] += log2(solver->mab_decisions)/log2(solver->mab_conflicts);
        for (all_variables (idx)) solver->mab_chosen[idx]=0;
        solver->mab_chosen_tot = 0;
        solver->mab_decisions = 0;
+  solver->mab_conflicts = 0;
        for(unsigned i=0;i<solver->mab_heuristics;i++) stable_restarts +=  solver->mab_select[i];
        if(stable_restarts < solver->mab_heuristics) {
                solver->heuristic = solver->heuristic==0?1:0;
$
```

```
/Users/biere/sc2026/solvers/green/lymphosat/src/solver/solvers
$ ls
01-integer-programming          fdmus                          multiplier-circuits            rooks
algebra                         fermat                         multiplier-verification        sat-x
algorithm-equivalence-checking  fixed-shape-random             mutilated-chessboard           satcoin
alloy-vpn-models                floodit-puzzle                 oddball-weighing               set-covering
antibandwidth                   fpga-routing                   or_randxor                     sgen
auto-correlation                genurq                         ordering-principle             sgen-balanced
automata-synchronization        gf3-sat                        ordering-principle-xor         sliding-puzzle
battleship                      glassy-gen                     p-center                       social-golfer
binary-pigeon-hole              graceful-production            pebbling                       software-bmc
binary-tree-parity              grandtour-puzzle               petrinet-concurrency           software-verification
brent-equations                 greentao                       philips                        sorting-networks
circuit-equialence-checking     hamiltonian-cycle              phnf                           ssp-0
circuit-minimization            hanoi                          pigeon-hole                    st-connectivity-principle
clique-coloring                 heule-nol                      planning                       station-repacking
clique-formulas                 hgen                           product-configuration          stedman-triples
clique-width                    hidoku                         profitable-robust-production   stone
clustered-random                hypertree-decomposition        purdom-instances               subgraph-isomorphism
coloring-clique                 independent-set                pythagorean-triples            subsumptiontest
coloring-mycielski-graph        independent-set-reconfiguration railway-safety                sum-of-3-cubes
core-based-generator            influence-maximization         ramsey                         summle
cover                           interval-matching              ramsey-numbers                 tensors
cryptography-ascon              karatsuba-multiplication       ramseycube                     test-configuration
cryptography-cbmc               knights-problem                random                         testpattern-generation
cryptography-simon              ktf                            random-circuits                theorem-proving
design-debugging                long-learned-clauses           random-clustered               tree-decomposition
diagnosis                       maximum-constraint-partition   random-csp                     trigonometric-functions
discrete-logarithm              md5-equivalence-checking       random-mus                     tseitin-formulas
edge-matching                   mechanical-master-key          random-planted-solution        uniform-random
edit-distance                   minimal-superpermutation       rbsat                          waerden
equivalence-chain-principle     minimum-disagreement-parity    register-allocation            xor_op
erdos-discrepancy               modcircuits                    relational-dependencies        xor-chain
fallback                        mosoi-289                      relativized-pigeon-hole
$
```

```
coloring-clique                 independent-set                 pythagorean-triples             subsumptiontest
coloring-mycielski-graph        independent-set-reconfiguration railway-safety                  sum-of-3-cubes
core-based-generator            influence-maximization          ramsey                          summle
cover                           interval-matching               ramsey-numbers                  tensors
cryptography-ascon              karatsuba-multiplication        ramseycube                      test-configuration
cryptography-cbmc               knights-problem                 random                          testpattern-generation
cryptography-simon              ktf                             random-circuits                 theorem-proving
design-debugging                long-learned-clauses            random-clustered                tree-decomposition
diagnosis                       maximum-constraint-partition    random-csp                      trigonometric-functions
discrete-logarithm              md5-equivalence-checking        random-mus                      tseitin-formulas
edge-matching                   mechanical-master-key           random-planted-solution         uniform-random
edit-distance                   minimal-superpermutation        rbsat                           waerden
equivalence-chain-principle     minimum-disagreement-parity     register-allocation             xor_op
erdos-discrepancy               modcircuits                     relational-dependencies         xor-chain
fallback                        mosoi-289                       relativized-pigeon-hole
$ cd software-verification/matcher
$ head match.cpp
#include "formula_profile.h"

LYMPHOSAT_MATCHER {
    const bool aws_encryption_like =
        profile.binary_clause_fraction >= 0.165 &&
        profile.binary_clause_fraction <= 0.169 &&
        profile.ternary_clause_fraction >= 0.711 &&
        profile.ternary_clause_fraction <= 0.714 &&
        profile.all_positive_clause_fraction >= 0.0018 &&
        profile.all_positive_clause_fraction <= 0.0024 &&
$ tail match.cpp
          linked_list_swap_unwind63_like ||
          linked_list_swap_unwind80_like ||
          itox_a_like || itox_b_like || itox_wide_like ||
          hard_like || hard_u7061_like || hard_u10652_like ||
          hard_9007_like || hard_15228_like || hard_15228_u10_like ||
          hard_8013_like || newton34_like || newton36_like ||
          zfcp_like || commutativity_like || string_compare_like ||
          aws_priority_like || filter1_like || stack25_like ||
          barrier3_like || sine8_like;
}
$
```

Why do we care in the age of AI?



Where are SAT Solvers useful?

- probably *should* be used whenever you have an NP-hard subproblem
- factoring out the NP-problem reasoning algorithms is beneficial
- *Satisfiability Modulo Theories* (SMT) solvers are driven by a SAT solver
- state-of-the-art in *Constraint Programming* (CP) is lazy clause generation with SAT
- state-of-the-art (first-order) automatic theorem proving relies on SAT splitting (AVATAR, subsumption)
- (higher-order) interactive theorem provers (Lean, Isabelle) rely on SAT, SMT or first-order hammers (automatically discharging simple facts, also *cf.* bv-decide)

Certificates and SAT as Key-Enabler for certain LLM Tasks

- SAT has mandatory proof checking since 2013
- almost all proofs have some propositional part
- interactive theorem provers rely for decades on a small “kernel”
- there is recent research on HW & SW model checking certificates
- SMT is still struggling (has been announcing proof support for a long time)
- being able to check certificates allows to use LLMs to generate them
  (results in challenges for mathematics and theoretical computer science)
- needed for producing correct code and scalability of LLM based code generation

Challenges for SAT and Automated Reasoning in the age of AI

- LLMs (generative / foundational models) do need precise reasoning
- (again) machine checkable certificates are tremendously useful
- can AI be used to improve the state-of-the-art of AR?
- what is the role of falsification techniques?