Advanced SAT Tutorial
Quantum Leaps and Local Search

Armin Biere
universität freiburg

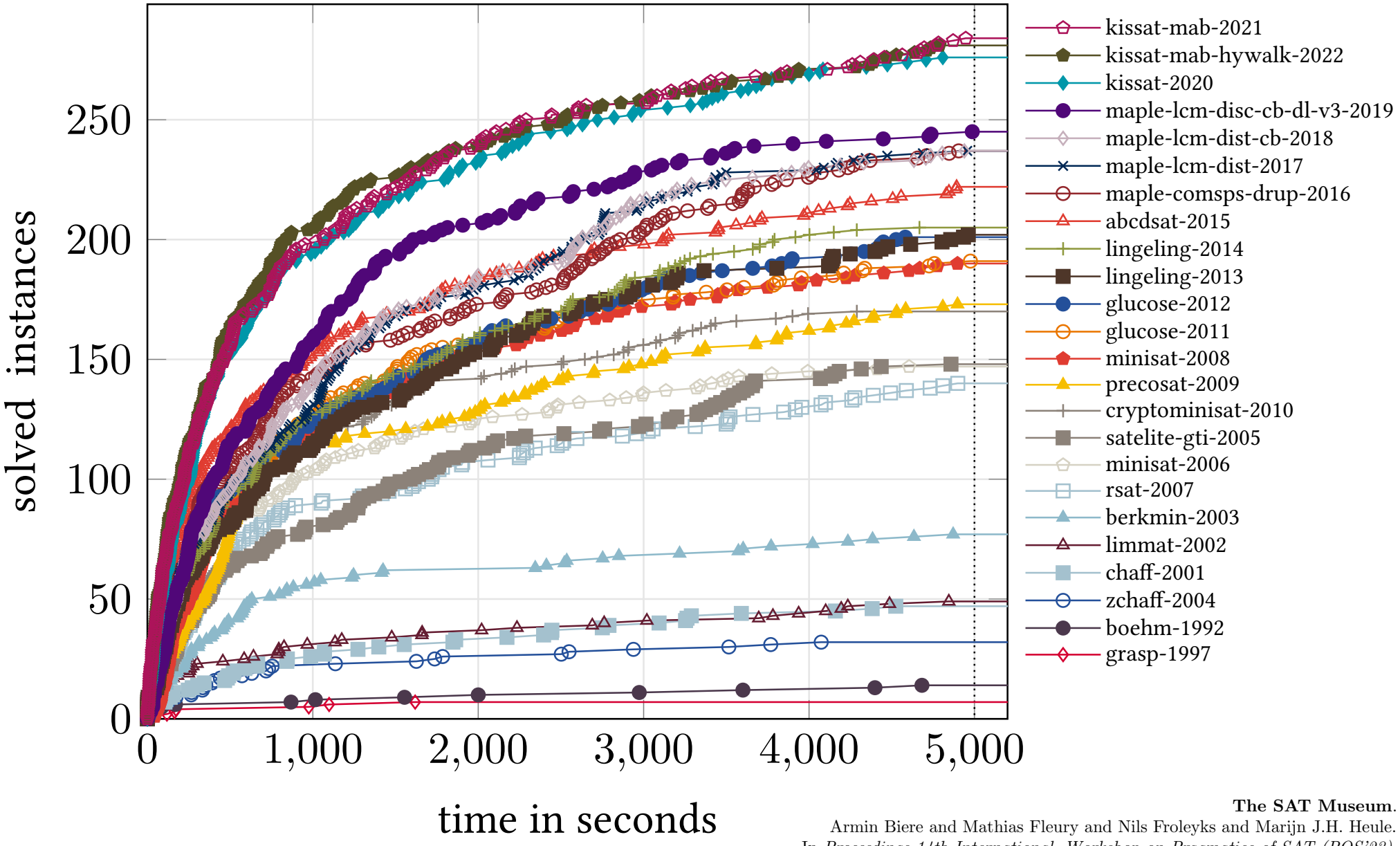
SAT + SMT Summer School 2024

27th Intl. Conf. on Theory and Applications of Satisfiability Testing

August 19, 2024, Pune, India

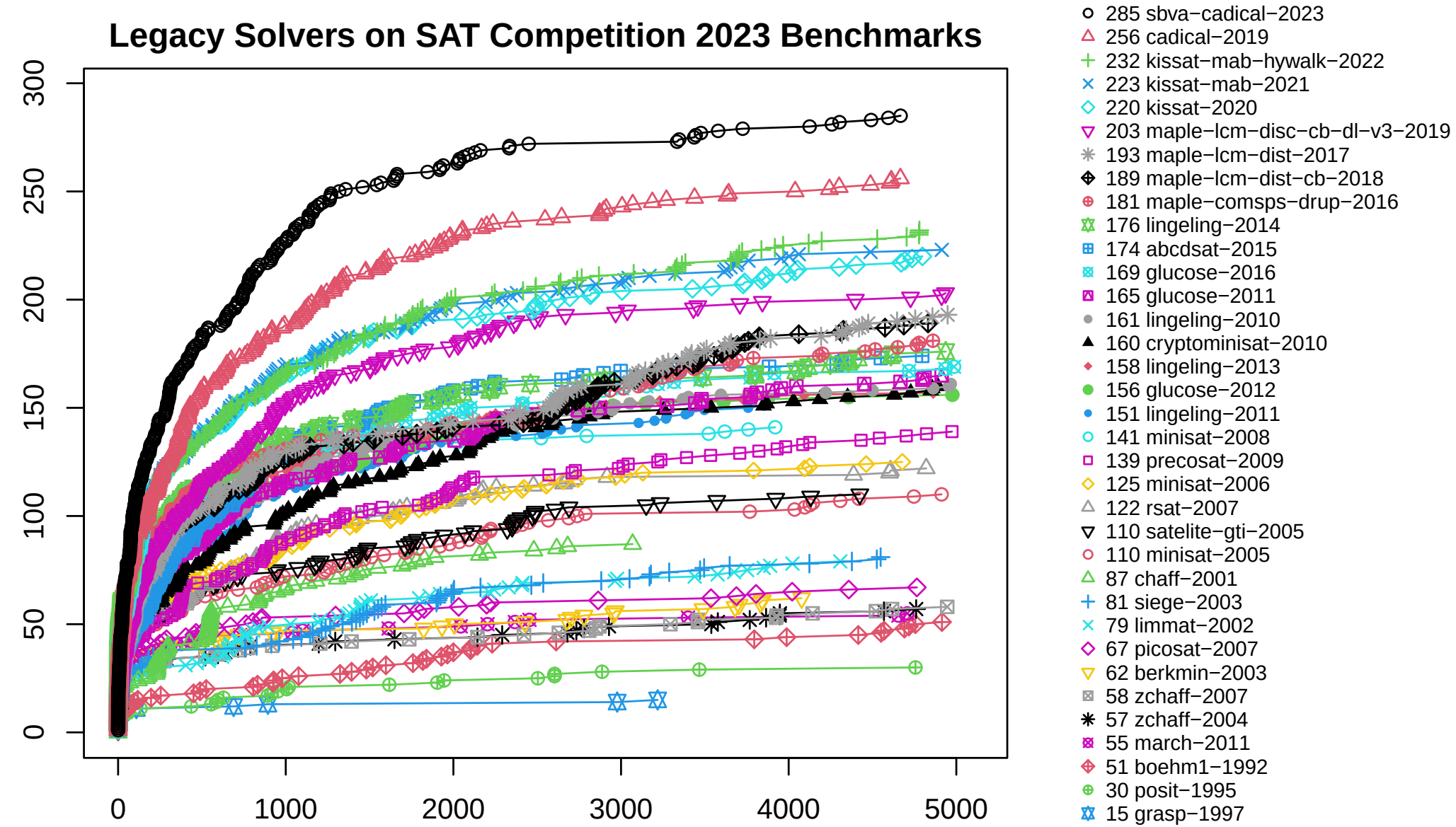
https://cca.informatik.uni-freiburg.de/biere/talks/Biere-SATSMT24-tutorial.pdf

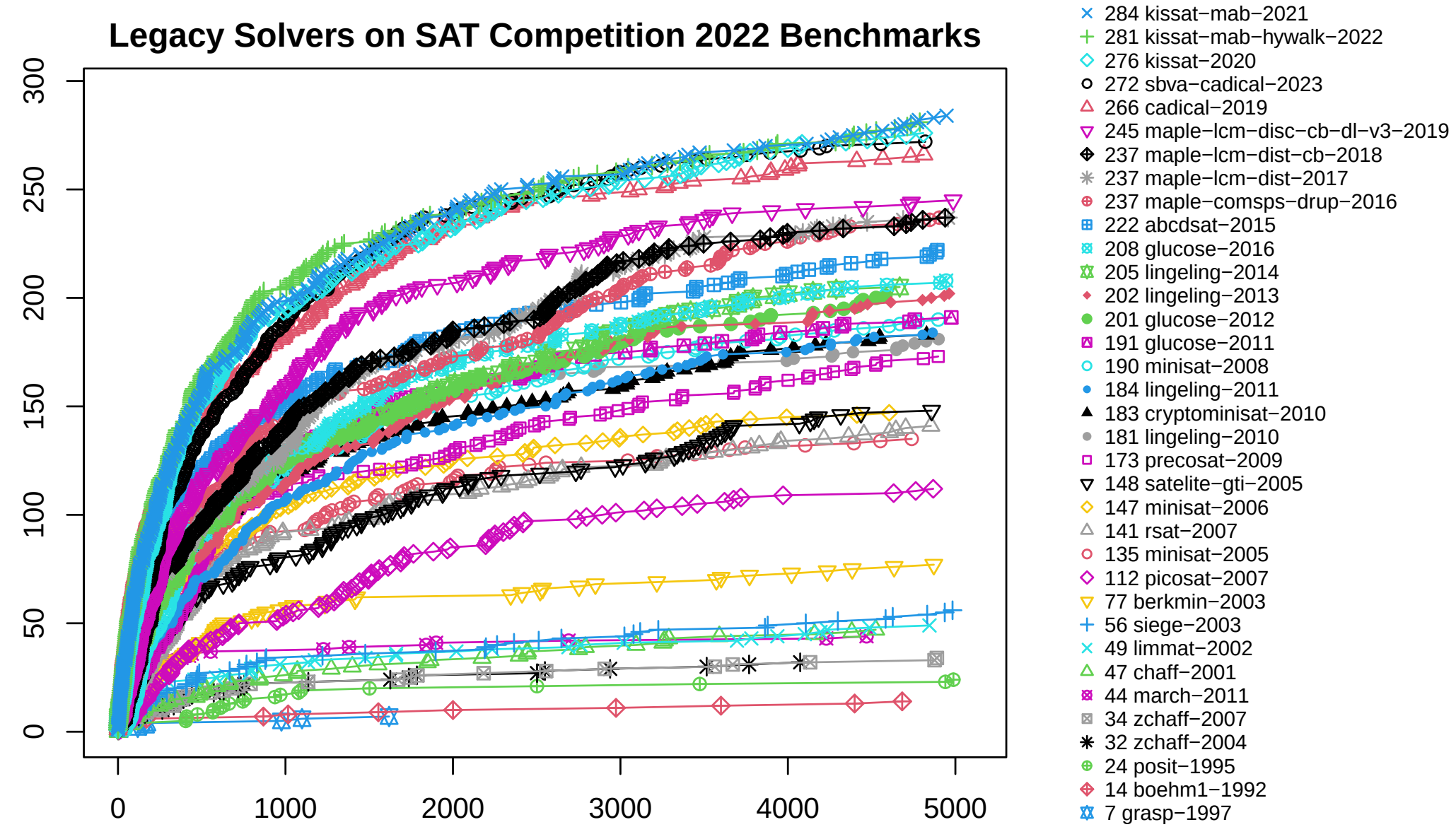
SAT Competition All Time Winners on SAT Competition 2022 Benchmarks



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]


Legacy Solvers on SAT Competition 2023 Benchmarks
0
50
100
150
200
250
300
0
1000
2000
3000
4000
5000
285 sbva−cadical−2023
256 cadical−2019
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
169 glucose−2016
165 glucose−2011
161 lingeling−2010
160 cryptominisat−2010
158 lingeling−2013
156 glucose−2012
151 lingeling−2011
141 minisat−2008
139 precosat−2009
125 minisat−2006
122 rsat−2007
110 satelite−gti−2005
110 minisat−2005
87 chaff−2001
81 siege−2003
79 limmat−2002
67 picosat−2007
62 berkmin−2003
58 zchaff−2007
57 zchaff−2004
55 march−2011
51 boehm1−1992
30 posit−1995
15 grasp−1997


Legacy Solvers on SAT Competition 2022 Benchmarks
0
50
100
150
200
250
300
0
1000
2000
3000
4000
5000
284 kissat−mab−2021
281 kissat−mab−hywalk−2022
276 kissat−2020
272 sbva−cadical−2023
266 cadical−2019
245 maple−lcm−disc−cb−dl−v3−2019
237 maple−lcm−dist−cb−2018
237 maple−lcm−dist−2017
237 maple−comsps−drup−2016
222 abcdsat−2015
208 glucose−2016
205 lingeling−2014
202 lingeling−2013
201 glucose−2012
191 glucose−2011
190 minisat−2008
184 lingeling−2011
183 cryptominisat−2010
181 lingeling−2010
173 precosat−2009
148 satelite−gti−2005
147 minisat−2006
141 rsat−2007
135 minisat−2005
112 picosat−2007
77 berkmin−2003
56 siege−2003
49 limmat−2002
47 chaff−2001
44 march−2011
34 zchaff−2007
32 zchaff−2004
24 posit−1995
14 boehm1−1992
7 grasp−1997

Overview

Part 1

- competitions and quantum leaps
- preprocessing: bounded variable elimination / addition
- portfolio / stable-focused-mode
- target phases, rephasing

Part 2

- random local search, restarts
- Las Vegas algorithm, probabilistic approximate complete (PAC)
- focused random search, random walks
- critical clauses, make-, break-value
- dynamic local search, clause weighting

Bounded Variable Elimination (BVE) [EénBiere-SAT'05]

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 $\cancel{(a \vee \bar{a} \vee \bar{b})_{14}}$ $(a \vee d)_{15}$ $\cancel{(b \vee \bar{a} \vee \bar{b})_{24}}$ $(b \vee d)_{25}$ $(c \vee \bar{a} \vee \bar{b})_{34}$ $(c \vee d)_{45}$

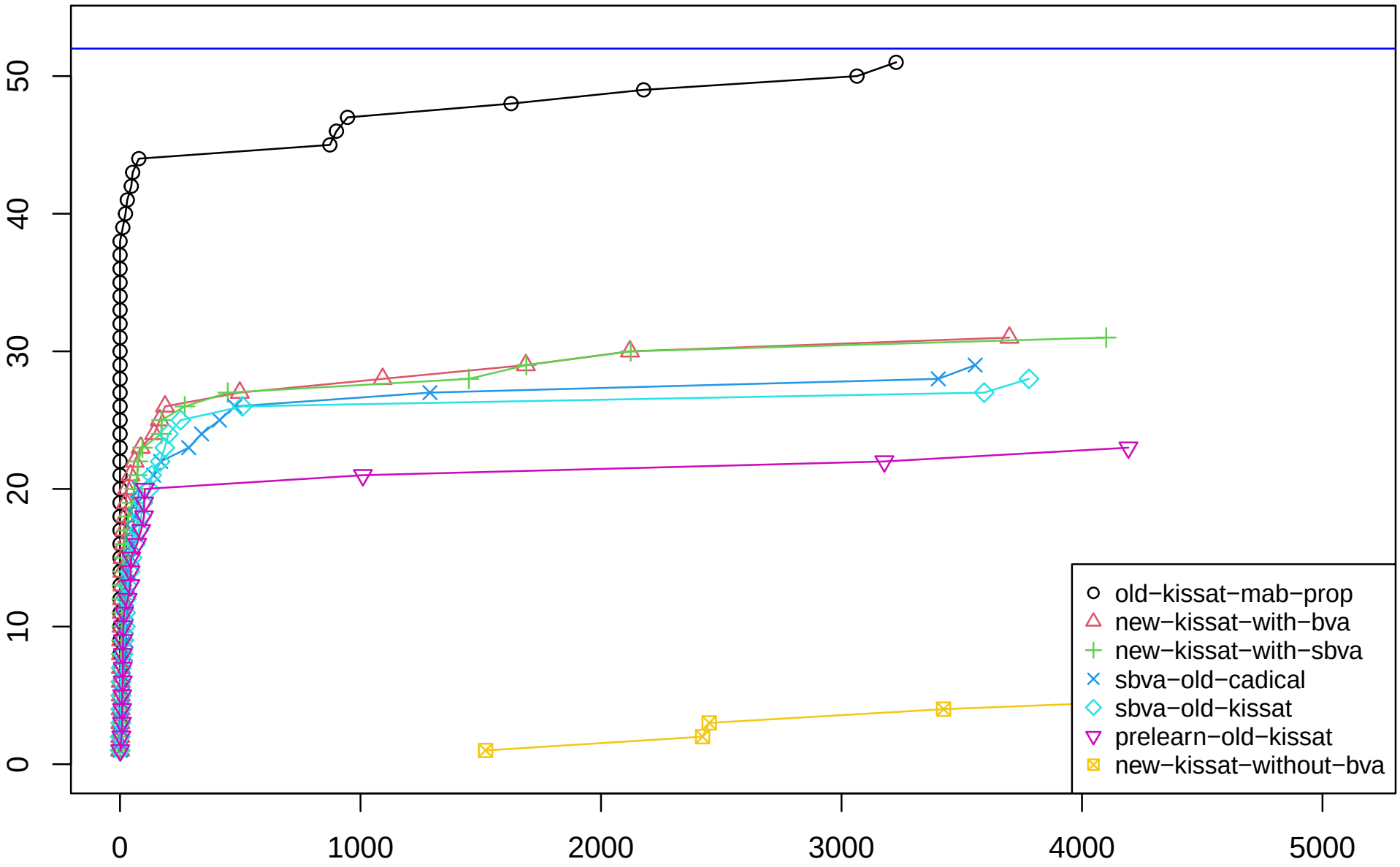
- number of clauses *not increasing*
- strengthen and remove subsumed clauses too
- most important and most effective preprocessing we have

Bounded Variable Addition (BVA) [MantheyHeuleBiere-HVC'12] [HaberlandtGreenHeule-SAT'23]

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

- 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

BVA on 52 Hard SAT Competition 2023 Main Track Instances



Clause Elimination / Addition Preprocessing

- eliminating pure literals (and clauses in which they occur)
  - eliminating "autarkies": partial assignment satisfying all touched clauses
  - theory [Kullmann-Handbook'09], implementation [CaiZhangFleuryBiere'22]
- subsumed clauses + strengthening by self-subsuming resolution [EenBiere'05]
  - bounded variable elimination (BVE) generates many opportunities here
- adding "blocked clauses" [Kullmann'96] proposed (only theory)
- eliminating "blocked clauses" (BCE) [JärvisaloBiereHeule'10]
  - started journey for more powerful proof systems:

    Covered Clauses'10, Asymmetric Tautolgies (AT)'10, Resolution AT (RAT)'12
    Super Blocked'16, Propagation Redundant (PR)'17, Globally BC'19
    Substitution Redundant (SR)'19, veriPB'20, incremental proofs'24
  - increases trust on SAT solver results (math, industry, competition)
  - spectacular new math results with SAT solvers (by Marijn Heule et.al.)

    Pythagorean Triples Problem [HeuleKullmanMarek'16], ...
- **How to <u>add</u> interesting Blocked/PR/SR clauses?** beware: removes models
  - practically widely successful only through BVA

Blocked Clauses

addition [Kullmann'96] elimination [JärvisaloBiereHeule'10]

given CNF F

given clause C usually assumed not to be in F

given literal $\ell \in C$ fixed from here on

clause $D \in F$ with $\bar{\ell} \in D$ is called resolution candidate on ℓ in F

C is blocked in F on $\ell \in C$ if
resolution of C on ℓ with any resolution candidate D in F is tautological

$$\underbrace{(x \vee \bar{a} \vee \bar{b})_4\,(\bar{x} \vee a)_5\,(\bar{x} \vee b)_6}_{x = a \wedge b}\ \overbrace{(a \vee c)_9\,(b \vee c)_8\,(\bar{a} \vee \bar{b} \vee \bar{c})_7}^{\bar{c} = a \wedge b}\ \underbrace{(\bar{y} \vee c)_3\,(\bar{y} \vee x)_2\,(y \vee \bar{c} \vee \bar{x})_1}_{y = c \wedge x}$$

last clause (1) is blocked on y and can be removed and then the $\bar{y}$ clauses (2,3)

first clause (4) is blocked on x and can be removed and then the $\bar{x}$ clauses (5,6)

finally the remaining ternary clause is blocked on $\bar{c}$ and the rest (8,9) on c

Blocked Clause Elimination (BCE) Completeness

F satisfiable then $F \wedge C$ satisfiable

C blocked in F on $\ell \in C$

assume F satisfiable and pick model σ of F $\sigma(F) = 1$

w.l.o.g. assume $\sigma(C) = 0$ otherwise $\sigma(F \wedge C) = 1$ also note $\sigma(\ell) = 0$

let σ' be identical to σ but with the value on ℓ flipped $\sigma'(\ell) = \neg\sigma(\ell) = 1$

let $D \in F$ has three cases:

1. if $\ell, \bar{\ell} \notin D$ then $\sigma'(D) = \sigma(D) = \sigma(F) = 1$
2. if $\ell \in D$ clearly $\sigma'(D) = 1$ it only becomes "more" true as $\sigma(\ell) = 0$, $\sigma'(\ell) = 1$
3. if $\bar{\ell} \in D$ then D is a resolution candidate in F on ℓ

   resolvent of C and D on ℓ tautological

   exists $k \in C$ and $\bar{k} \in D$ with $k \neq \ell, \bar{\ell}$

   $\sigma'(\bar{k}) = \sigma(k) = 1 = \sigma'(D)$

D "double satisfied" by σ

Blocked Clause Addition (BCA) Soundness

$F \wedge C$ unsatisfiable then F unsatisfiable by contraposition

Extended Resolution (ER) as Blocked Clause Addition (BCA)

ER adds Tseitin encoding $(\bar{x} \vee a)_1 (\bar{x} \vee b)_2 (x \vee \bar{a} \vee b)_3$ of $x = a \wedge b$ AND gate with fresh x

simulate by adding clauses 1,2 with pure literal $\bar{x}$ and then clause 3 blocked on x

Resolution Asymmetric Tautology (RAT)

C is an asymmetric tautology (AT) if it is unit-implied by F

propagating its negation leads to a conflict — also called RUP — learned clauses in CDCL are AT

C is a resolution asymmetric tautology (RAT) if all resolvents on fixed $\ell \in C$ are AT

Bounded Variable Addition (BVA) as RAT Addition

$$\begin{array}{ll} (d \vee a)_6 & (e \vee a)_9 \\ (d \vee b)_7 & (e \vee b)_{10} \\ (d \vee c)_8 & (e \vee c)_{11} \end{array} \quad \text{by} \quad \begin{array}{ll} (\boxed{x} \vee a)_1 & (d \vee \boxed{\bar{x}})_4 \\ (\boxed{x} \vee b)_2 & (e \vee \boxed{\bar{x}})_5 \\ (\boxed{x} \vee c)_3 & \end{array}$$

First add the three quotient clauses (1-3) as blocked clauses on x.

Then add the two factor clauses (4,5) as RAT on $\bar{x}$.

Finally remove the original six factored clauses (6-11) as AT.

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:

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

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

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 `1` with most occurrences.

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

Quotients are all other literals `2` ... `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 3 with most common occurrences in previous quotients.

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

Reduce quotients to common ones 4 ... 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 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 $Q \cdot F - Q - F = 2 \cdot 7 - 2 - 7 = 14 - 9 = \mathbf{5}$

Eight 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 $\boxed{8}$ with most common occurrences in previous quotients.

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

Reduce quotients to single remaing common one 9.

Reduction $Q \cdot F - Q - F = 1 \cdot 8 - 1 - 8 = 8 - 9 = -\mathbf{1}$

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
```

Potential Second Factorization with Reduction $4 \cdot 2 - 4 - 2 = \mathbf{2}$

```
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 $2 \cdot 3 - 2 - 3 = 1$

```
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
```

Generic Reduction Table with Two Example Reduction Traces

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3	-1	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33
4	-1	2	5	8	**11**	14	17	20	23	26	29	32	35	38	41	44	47	50
5	-1	3	7	**11**	15	19	23	27	31	35	39	43	47	51	55	59	63	67
6	-1	4	9	14	19	24	29	34	39	44	49	54	59	64	69	74	79	84
7	-1	5	11	17	23	29	35	41	47	53	59	65	**71**	77	83	89	95	101
8	-1	6	13	20	27	34	41	48	55	62	69	76	83	90	97	104	111	118
9	-1	7	15	23	31	39	47	55	63	71	79	87	95	103	111	119	127	135
10	-1	8	17	26	35	44	53	62	**71**	80	89	98	107	116	125	134	143	152
11	-1	9	19	29	39	49	59	69	79	89	99	109	119	129	139	149	159	169
12	-1	10	21	32	43	54	65	76	87	98	109	120	131	142	153	164	175	186
13	-1	11	23	35	47	59	71	83	95	107	119	131	143	155	167	179	191	203
14	-1	12	25	38	51	64	77	90	103	116	129	142	155	168	181	194	207	220
15	-1	13	27	41	55	69	83	97	111	125	139	153	167	181	195	209	223	237
16	-1	14	29	44	59	74	89	104	119	134	149	164	179	194	209	224	239	254
17	-1	15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255	271
18	-1	16	33	50	67	84	101	118	135	152	169	186	203	220	237	254	271	288

Rows quotients Q, columns factors F.

Adding one new factor moves one column to the right.

But such a step does need to reduce quotients. So horizontal moves possible.

Continue factor selection and pick maximum reduction.

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

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

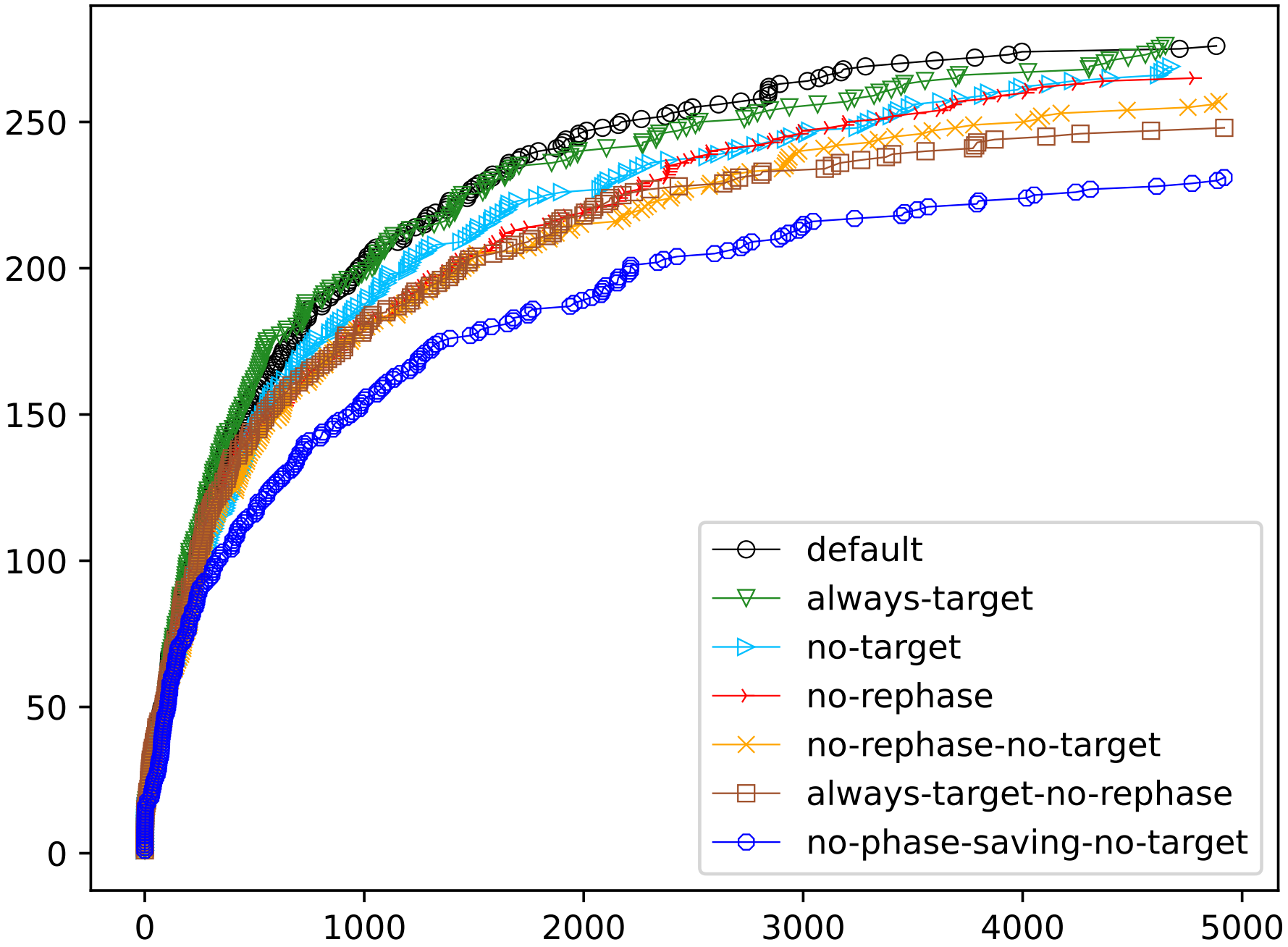
Target Phases

- 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 (use *saved phase*) used the last assigned value

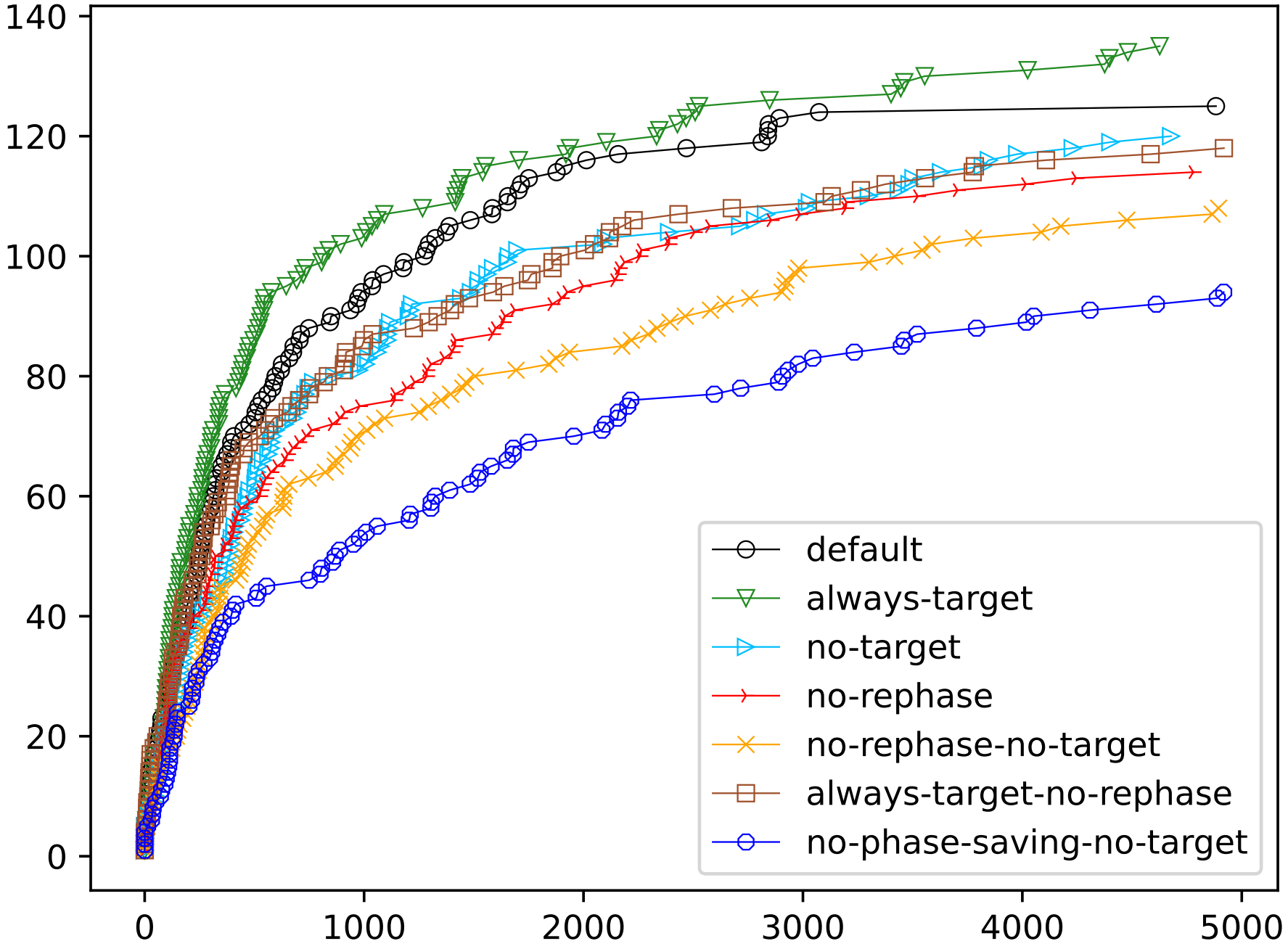
Rephasing

- 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*

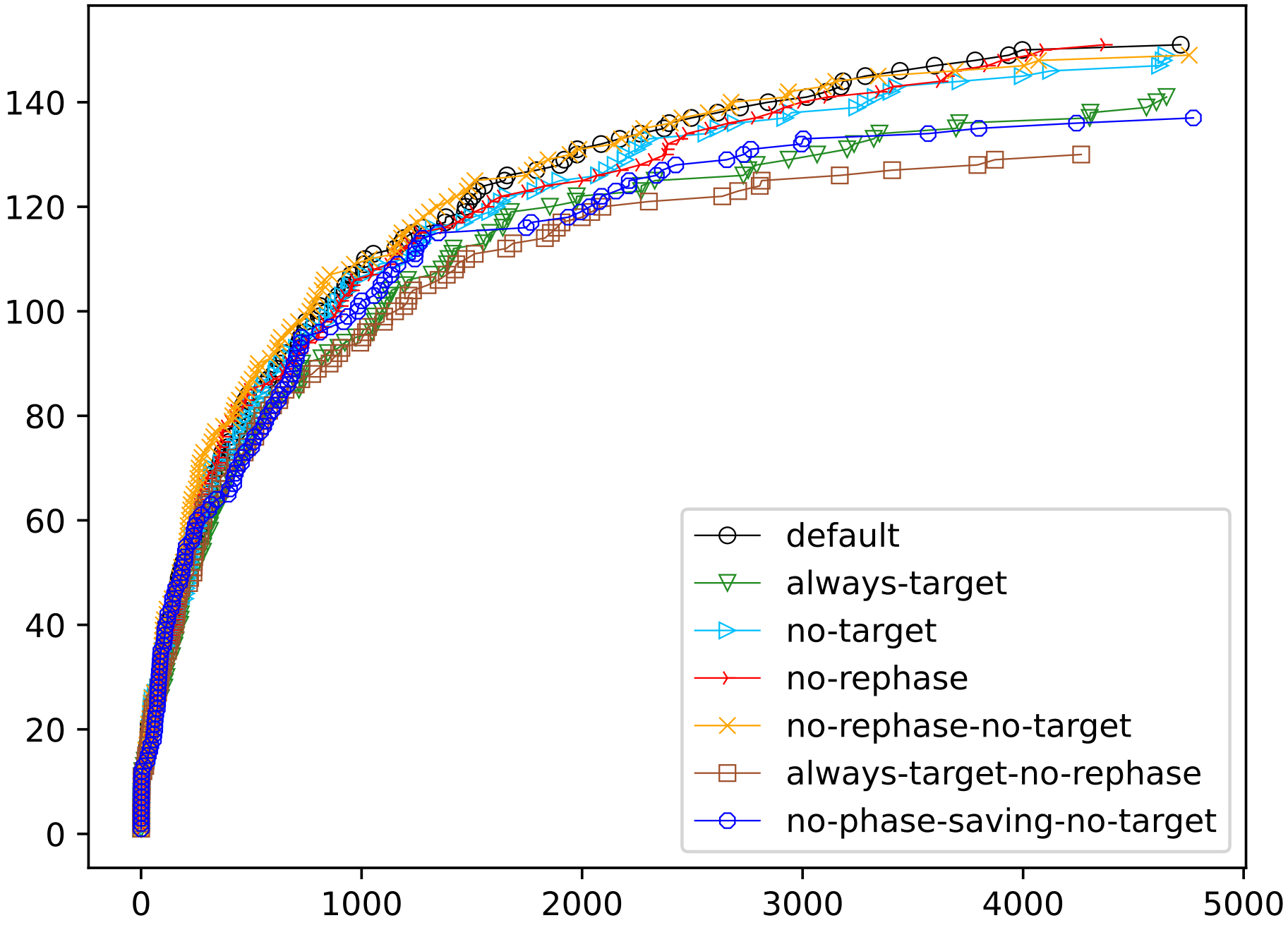
Kissat on All SAT Competition 2021 Main Track Instances



Kissat on Satisfiable SAT Competition 2021 Main Track Instances



Kissat on Unsatisfiable SAT Competition 2021 Main Track Instances



Random Local Search Algorithm

flip-variable (variable assignment α, variable x) // α is call-by-reference

1 $\alpha \leftarrow \alpha'$ with $\alpha'(y) = \begin{cases} \neg\alpha(x) & \text{if } x = y \\ \alpha(x) & \text{otherwise } (x \neq y) \end{cases}$

random-local-search (CNF F)

2 $\alpha \leftarrow$ random total assignment of variables in F

3 **while** $\alpha(F) \neq 1$

4 pick variable x in F randomly

5 *flip-variable* (α, x)

6 **return** α

Completeness

- considered local search algorithms are “incomplete” in two ways:
  - either might not terminate or return “unknown”
  - only can return satisfying assignments
- fall in the class of *Las Vegas* randomized algorithms
  - Monte Carlo: might return incorrect result with some (low) probability
  - Las Vegas: result guaranteed correct but with varying runtime
- applications
  - practical instances: generate test cases, hit coverage holes, ...
  - excellent for randomly generated instances or hard combinatorial problems
  - “Local Search for SMT” [Fröhlich...-AAAI’15] [Niemetz...-FMCAD’15]
  - phase assignment for complete algorithms (CaDiCaL, Kissat, ...) [CaiZhangFleuryBiere-JAIR’22] plus target phases
- probabilistic approximate complete (PAC) — [Hoos99]
  - algorithm does not get trapped and can escape local minima
  - always has a non-zero probability to find satisfying assignment

Random Local Search Algorithm with Restart

```
random-local-search-with-restarts (CNF F)
1    α ← random total assignment of variables in F
2    while α(F) ≠ 1
3        if it is time to restart then
4            α ← random total assignment of variables in F
5        else
6            pick variable x in F randomly
7            flip-variable (α, x)
8    return α
```

When it is time to restart?

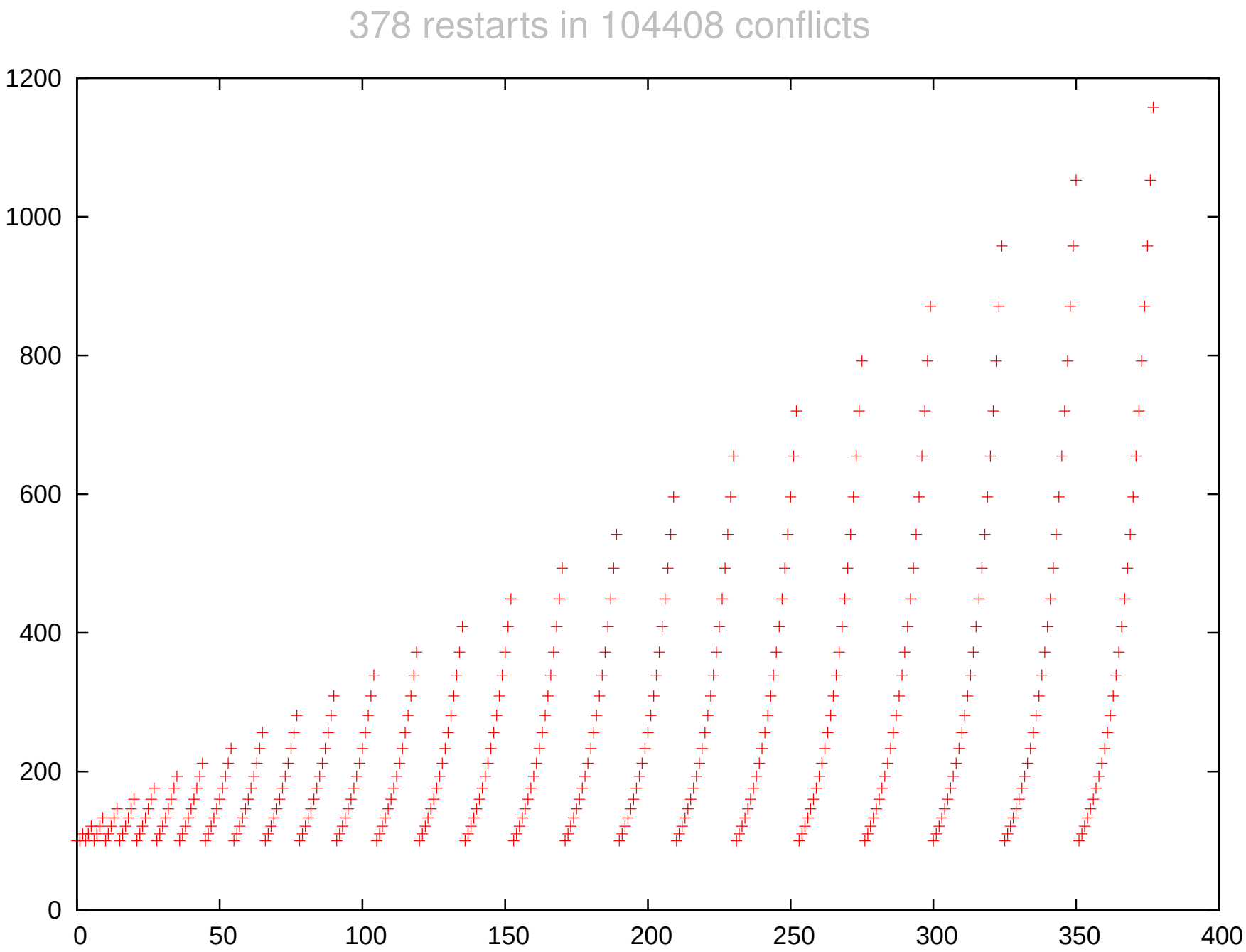
- literature on local search has run-time parameter *max-flipped*
- often number of restarts is limited by *max-tries* too (i.e., outer loop)
- restart scheduling methods:
  - **never** or **always** restart
  - **constant** number of flips (i.e., *max-flipped*)
  - **arithmetically** increase restart interval (increment $i > 0$)
    - $\boxed{\textit{max-flipped} \leftarrow \textit{max-flipped} + i}$
  - **geometrically** increase restart interval (times factor $f > 1$)
    - $\boxed{\textit{max-flipped} \leftarrow f \cdot \textit{max-flipped}}$
  - **inner-outer** geometric increase (as in PicoSAT)
    - use *inner* for *max-flipped* initialized with *initial*
    - $\boxed{\begin{array}{l} \textit{inner} \leftarrow f \cdot \textit{inner} \quad \text{after every restart and} \\ \text{if } \textit{inner} > \textit{outer} \text{ then reset } \textit{inner} \leftarrow \textit{initial} \text{ and } \textit{outer} \leftarrow f \cdot \textit{outer} \end{array}}$
  - **Luby** scheme (also called **reluctant doubling** by Knuth)

DDFW [IshtaiwiThorntonSattarPham-CP'05] [ChowdhuryCodelHeule-NFM'23]

divide-distribute-fixed-weights (CNF F)

initialize clause weigths $\omega : F \to \mathbb{Q}$ with $\omega(C) = \omega_0$
for i = 1 to *max-tries*
    $\alpha \leftarrow$ random total assignment of variables in F
    for j = 1 to *max-flipped*
        if $\alpha(F) = 1$ then return "SATISFIABLE"
        if exists *weight-reducing-variable*
            flip-variable (α, x) with x most reducing one and **continue**
        if exists *sideways-variable* and with propability p
            flip-variable (α, x) with x is *sideways* and **continue**
        for all falsified clauses C
            $D \leftarrow$ maximum-weighted-satisfied clause neighboring C
            if $\omega(D) < \omega_0$ or with probability q
                $D \leftarrow$ random satisfied clause with $\omega(C) \leq \omega_0$
            if $\omega(D) > \omega_0$ **then** *transfer* $\omega_>$ from D to C
            else *transfer* $\omega_=$ from D to C
return "UNKNOWN"

Inner-Outer Restart Intervals



Inner-Outer Restart Scheduling

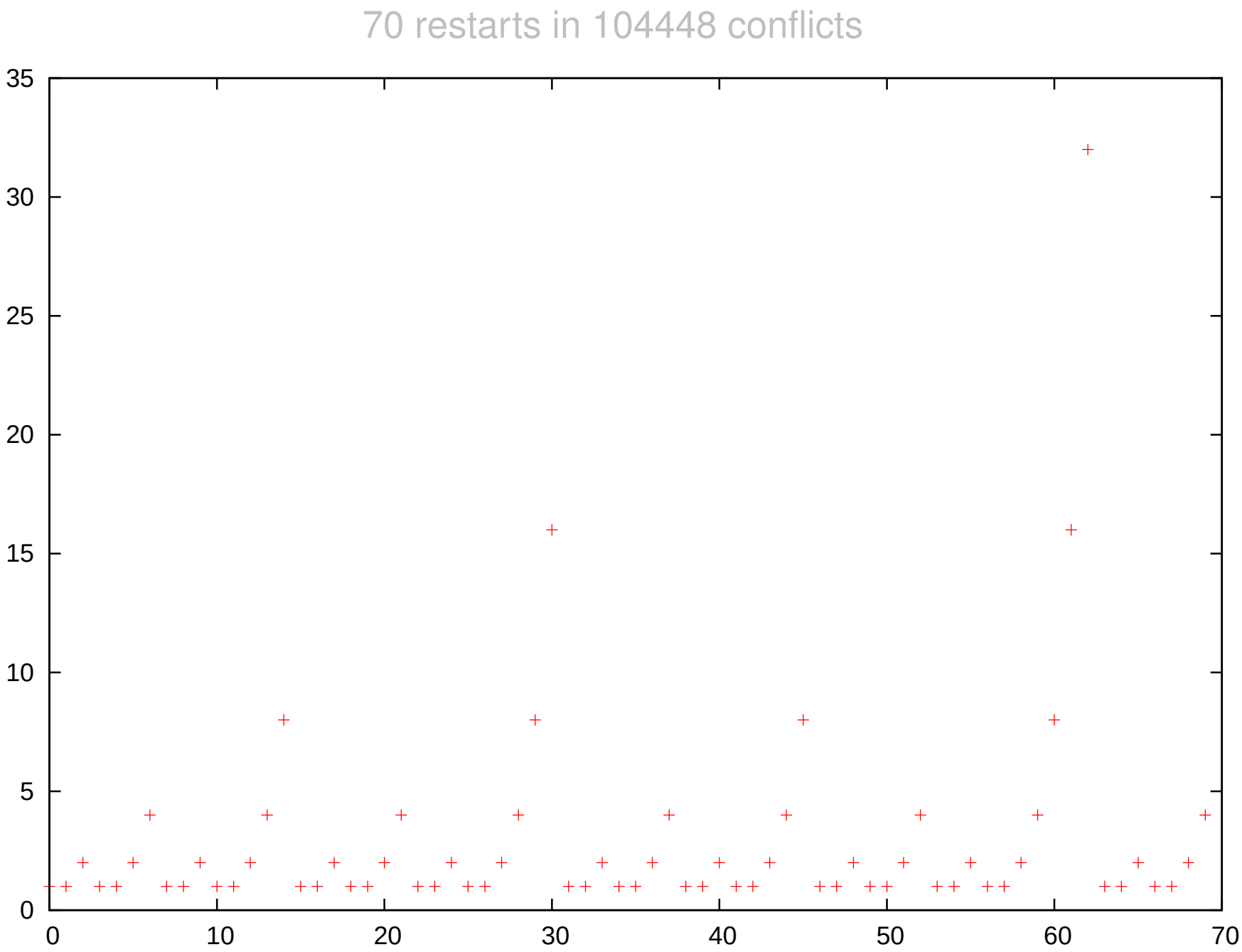
```
int inner = 100, outer = 100;
int restarts = 0, flipped = 0;

for (;;)
  {
    ... // run SAT core loop for 'inner' flips

    restarts++;
    flipped += inner;

    if (inner >= outer)
      {
        outer *= 1.1;
        inner = 100;
      }
    else
      inner *= 1.1;
  }
```

Luby's Restart Intervals



Luby Restart Scheduling

```
unsigned
luby (unsigned i)
{
  unsigned k;

  for (k = 1; k < 32; k++)
    if (i == (1 << k) - 1)
      return 1 << (k - 1);

  for (k = 1;; k++)
    if ((1 << (k - 1)) <= i && i < (1 << k) - 1)
      return luby (i - (1 << (k-1)) + 1);
}

limit = 512 * luby (++restarts);
...  // run SAT core loop for 'limit' flips
```

Reluctant Doubling Sequence

[Knuth'12]

$$(u_1, v_1) := (1,1)$$

$$(u_{n+1}, v_{n+1}) := (u_n \,\&\, -u_n = v_n \,?\, (u_n + 1, 1) : (u_n, 2v_n))$$

$$(1,1),\ (2,1),\ (2,2),\ (3,1),\ (4,1),\ (4,2),\ (4,4),\ (5,1),\ \ldots$$

Focused Random Walk without Restart

focused-random-walk-with-restarts (CNF F)

1 $\alpha \leftarrow$ random total assignment of variables in F
2 **while** $\alpha(F) \neq 1$
3 pick unsatisfied clause $C \in F$ randomly // with $\alpha(C) = 0$
4 pick literal $\ell \in C$ randomly
5 let $x = |\ell|$
6 *flip-variable* (α, x) // flip variable in falsified clause
7 **return** α

Focused Random Walk — [Papadimitrou'91] — [Schöning'99]

focused-random-walk-with-restarts (CNF F)

1 $\alpha \leftarrow$ random total assignment of variables in F
2 **while** $\alpha(F) \neq 1$
3 **if** it is time to restart **then**
4 $\alpha \leftarrow$ random total assignment of variables in F
5 **else**
6 pick unsatisfied clause $C \in F$ randomly // with $\alpha(C) = 0$
7 pick literal $\ell \in C$ randomly
8 let $x = |\ell|$
9 *flip-variable* (α, x) // flip variable in falsified clause
10 **return** α

Make-Value, Critical Clauses, Break-Value

make-value (CNF F, variable x, assignment α)

1 **return** $|\{C \in F \mid \ell \in C,\ |\ell| = x,\ \alpha(C) = 0\}|$ // number of falsified clauses with x

is-critical-clause (clause C, assignment α)

2 **return** $|\{\ell \in C \mid \alpha(\ell) = 1\}| = 1$ // exactly one literal of C set to true

break-value (CNF F, variable x, assignment α)

3 **return** $|\{C \in F \mid \ell \in C,\ |\ell| = x,\ \alpha(\ell) = 1,$ *is-critical-clause* $(C, \alpha)\}|$ // #critical

make-break-value (CNF F, variable x, assignment α)

4 **return** *make-value* (F, x, α) $-$ *break-value* (F, x, α) // made minus broken

Example: Make-Value, Critical Clauses, Break-Value

consider flipping x with $\alpha(x) = 1$ in the following CNF

C_1	$(\neg x \vee x_1 \vee \neg x_2)$	unsatisfied (falsified)
C_2	$(\boxed{x} \vee x_1 \vee x_3)$	critical (single satisfied by x)
C_3	$(\boxed{x} \vee \neg x_2 \vee \boxed{\neg x_3})$	double satisfied
C_3	$(\boxed{x} \vee \boxed{x_2} \vee \boxed{\neg x_3})$	triple satisfied

and after flipping x, so setting $\alpha(x) = 0$, we have

C_1	$(\boxed{\neg x} \vee x_1 \vee \neg x_2)$	**made** (and critical)
C_2	$(x \vee x_1 \vee x_3)$	**broken**
C_3	$(x \vee \neg x_2 \vee \boxed{\neg x_3})$	critical
C_3	$(x \vee \boxed{x_2} \vee \boxed{\neg x_3})$	double satisfied

thus both *make-value* and *break-value* are 1

Greedy Local Search — GSAT — [SelmanLevesqueMitchell-AAAI'92]

greedy-local-search-with-restarts (CNF F)

1 $\alpha \leftarrow$ random total assignment of variables in F

2 **while** $\alpha(F) \neq 1$

3 **if** it is time to restart **then**

4 $\alpha \leftarrow$ random total assignment of variables in F

5 **else**

6 $X \leftarrow \underset{x \text{ variable in } F}{\arg\max}$ *make-break-value* (F, x, α)

7 pick x randomly from X

8 *flip-variable* (α, x)

9 **return** α

GSAT with Random Walk — GWSAT — [SelmanKautz-IJCAI'93]

greedy-local-search-with-random-walk (CNF F)

1 $\alpha \leftarrow$ random total assignment of variables in F
2 **while** $\alpha(F) \neq 1$
3 **if** it is time to restart **then**
4 $\alpha \leftarrow$ random total assignment of variables in F
5 **else**
6 **with** probability p // random walk with probability p
7 pick random falsified clause C // $\alpha(C) = 0$
8 select random literal $\ell \in C$
9 set variable $x \leftarrow |\ell|$
10 **otherwise** // greedy step with probability $1 - p$
11 $X \leftarrow \underset{x \text{ variable in } F}{\arg\max}$ *make-break-value* (F, x, α)
12 pick x randomly from X
13 *flip-variable* (α, x)
14 **return** α

WalkSAT Local Search [SelmanKautzCohen-DIMACS93] [Hoos-AAAI'99]

```
   walksat-local-search (CNF F)
1      α ← random total assignment of variables in F
2      while α(F) ≠ 1
3          if it is time to restart then
4              α ← random total assignment of variables in F
5          else
6              pick random falsified clause C      // α(C) = 0
7              select literal ℓ ← walksat-selection-heuristic (F, C, α)
8              flip-variable (α, |ℓ|)
9      return α
```

WalkSAT Literal Selection Heuristic according to [Hoos-AAAI'99]

walksat-selection-heuristic (CNF F, clause C, assignment α)

1 $L \leftarrow \{\ell \in C \mid$ *break-value* $(F, |\ell|, \alpha) = 0\}$

2 **if** $L \neq \emptyset$

3 **return** $\ell \in L$ randomly // greedily select literal with *break-value* zero

4 **else with** probability p

5 **return** $\ell \in C$ randomly // with probability p select random literal

6 **otherwise**

7 $K \leftarrow \arg\min_{\ell \in C}$ *break-value* $(F, |\ell|, \alpha)$

8 **return** $\ell \in K$ randomly // with probability $1 - p$ minimize *break-value*

Other WalkSAT Heuristics

see also text-book [StützleHoos'04]

- Tabu Search
  - maintain a "tabu" list of recently flipped variables
  - those are not allowed to be flipped
- Novelty [McAllesterSelmanKautz-AAAI'97]
  - use variable with best *make-break-value*
  - select best variable if not most-recently-flipped (*age*)
  - otherwise with probability p choose second best
  - finally (with probability $1 - p$) still pick best
  - ties broken by *age*
- Novelty+ [Hoos-AAAI'99]
  - adds random walk step to Novelty
  - as otherwise Novelty is not PAC
- AdoptNovelty+ [Hoos-AAAI'02]
  - self-adapting noise mechanism

ProbSAT Local Search Algorithm — [BalintSchöning-SAT'12]

probsat-score (CNF F, literal ℓ, assignment α)

1 $b \leftarrow$ *break-value* (F, $|\ell|$, α)

2 **return** 2^{-b} // or other monotonically decreasing function $f(b)$ instead of b

probsat-local-search (CNF F)

3 $\alpha \leftarrow$ random total assignment of variables in F

4 **while** $\alpha(F) \neq 1$

5 pick random falsified clause C // $\alpha(C) = 0$

6 $S \leftarrow \sum_{\ell \in C}$ *probsat-score* (F, ℓ, α)

7 sample $\ell \in C$ over probability distribution $\ell \mapsto \frac{\textit{probsat-score}\,(F,\,\ell,\,\alpha)}{S}$

8 *flip-variable* (α, $|\ell|$)

9 **return** α

Clause Weights, Make-Score, Break-Score

$\boxed{\omega : F \to \mathbb{Q}}$

make-score (CNF F, variable x, assignment α, clause weights ω)

1 $G \leftarrow \{C \in F \mid \ell \in C,\ |\ell| = x,\ \alpha(C) = 0\}|$

2 **return** $\sum_{C \in G} \omega(C)$ // sum of weights of falsified clauses with x

break-score (CNF F, variable x, assignment α, clause weights ω)

3 $G \leftarrow |\{C \in F \mid \ell \in C,\ |\ell| = x,\ \alpha(\ell) = 1,$ *is-critical-clause* $(C, \alpha)\}|$

4 **return** $\sum_{C \in G} \omega(C)$ // sum of weights clauses where x is critical

make-break-score (CNF F, variable x, assignment α, clause weights ω)

5 **return** *make-score* (F, x, α) $-$ *break-score* (F, x, α) // improved score

Clause Weighting in Dynamic Local Search Algorithms

vast literature

dynamic-local-search (CNF F)

1 $\alpha \leftarrow$ random total assignment of variables in F
2 initialize clause weigths $\omega : F \to \mathbb{Q}$ with $\omega(C) = 1$
3 **while** $\alpha(F) \neq 1$
4 use *make*/*break* scores instead of values
5 same principles: “falsified clause”, “random walks”, “restarts”
6 additionally *scale* (increase) weights ω of unsatisfied clauses
7 optionally *smooth* (move to average) weights ω of all clauses
8 **return** α

- scaling and smoothing global
- slower (but better?) flips per second than ProbSAT
- much more complex algorithmics
- CCASat [CaiSu-AAAI’12] successful over the years
- TaSSAT [ChowdhuryCodelHeule-NFM’23,ChowdhuryCodelHeule-TACAS’24]
  extension of YalSAT [Biere’14]

DDFW [IshtaiwiThorntonSattarPham-CP'05] [ChowdhuryCodelHeule-NFM'23]

divide-distribute-fixed-weights (CNF F)

initialize clause weigths $\omega : F \to \mathbb{Q}$ with $\omega(C) = \omega_0$
for i = 1 to *max-tries*
    $\alpha \leftarrow$ random total assignment of variables in F
    for j = 1 to *max-flipped*
        if $\alpha(F) = 1$ then return "SATISFIABLE"
        if exists *weight-reducing-variable*
            flip-variable (α, x) with x most reducing one and **continue**
        if exists *sideways-variable* and with propability p
            flip-variable (α, x) with x is *sideways* and **continue**
        for all falsified clauses C
            $D \leftarrow$ maximum-weighted-satisfied clause neighboring C
            if $\omega(D) < \omega_0$ or with probability q
                $D \leftarrow$ random satisfied clause with $\omega(C) \leq \omega_0$
            if $\omega(D) > \omega_0$ **then** *transfer* $\omega_>$ from D to C
            else *transfer* $\omega_=$ from D to C
return "UNKNOWN"

TaSSAT: Transfer and Share SAT*

Md Solimul Chowdhury( ), Cayden R. Codel, and Marijn J. H. Heule

Carnegie Mellon University, Pittsburgh, PA, USA
{mdsolimc,ccodel,mheule}@cs.cmu.edu

Abstract. We present TaSSAT, a powerful local search SAT solver that effectively solves hard combinatorial problems. Its unique approach of transferring clause weights in local minima enhances its efficiency in solving problem instances. Since it is implemented on top of YalSAT, TaSSAT benefits from practical techniques such as restart strategies and thread parallelization. Our implementation includes a parallel version that shares data structures across threads, leading to a significant reduction in memory usage. Our experiments demonstrate that TaSSAT outperforms similar solvers on a vast set of SAT competition benchmarks. Notably, with the parallel configuration of TaSSAT, we improve lower bounds for several van der Waerden numbers.

Keywords: Local Search for SAT · Weight Transfer · Memory Efficiency

[ChowdhuryCodelHeule-TACAS'24]

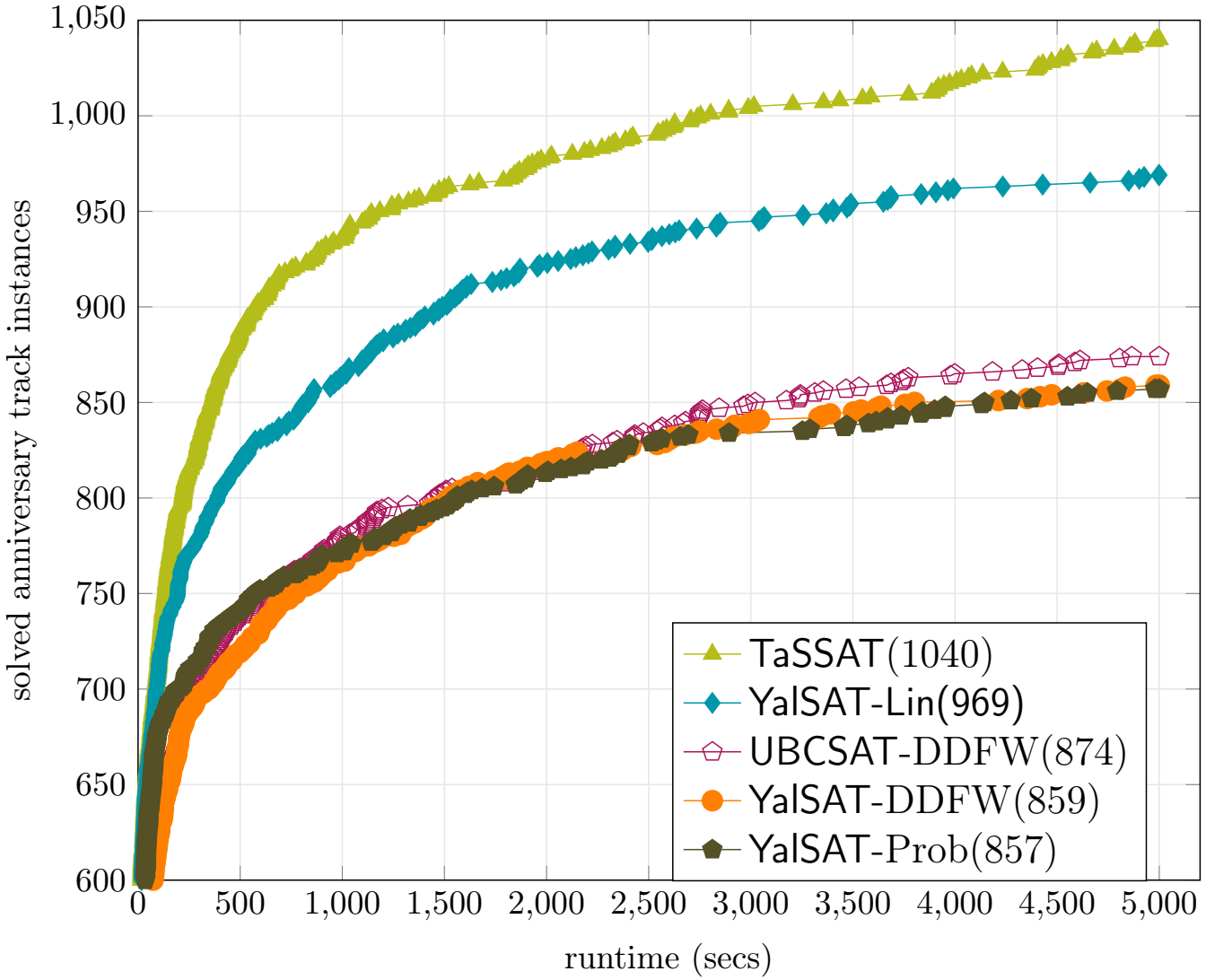
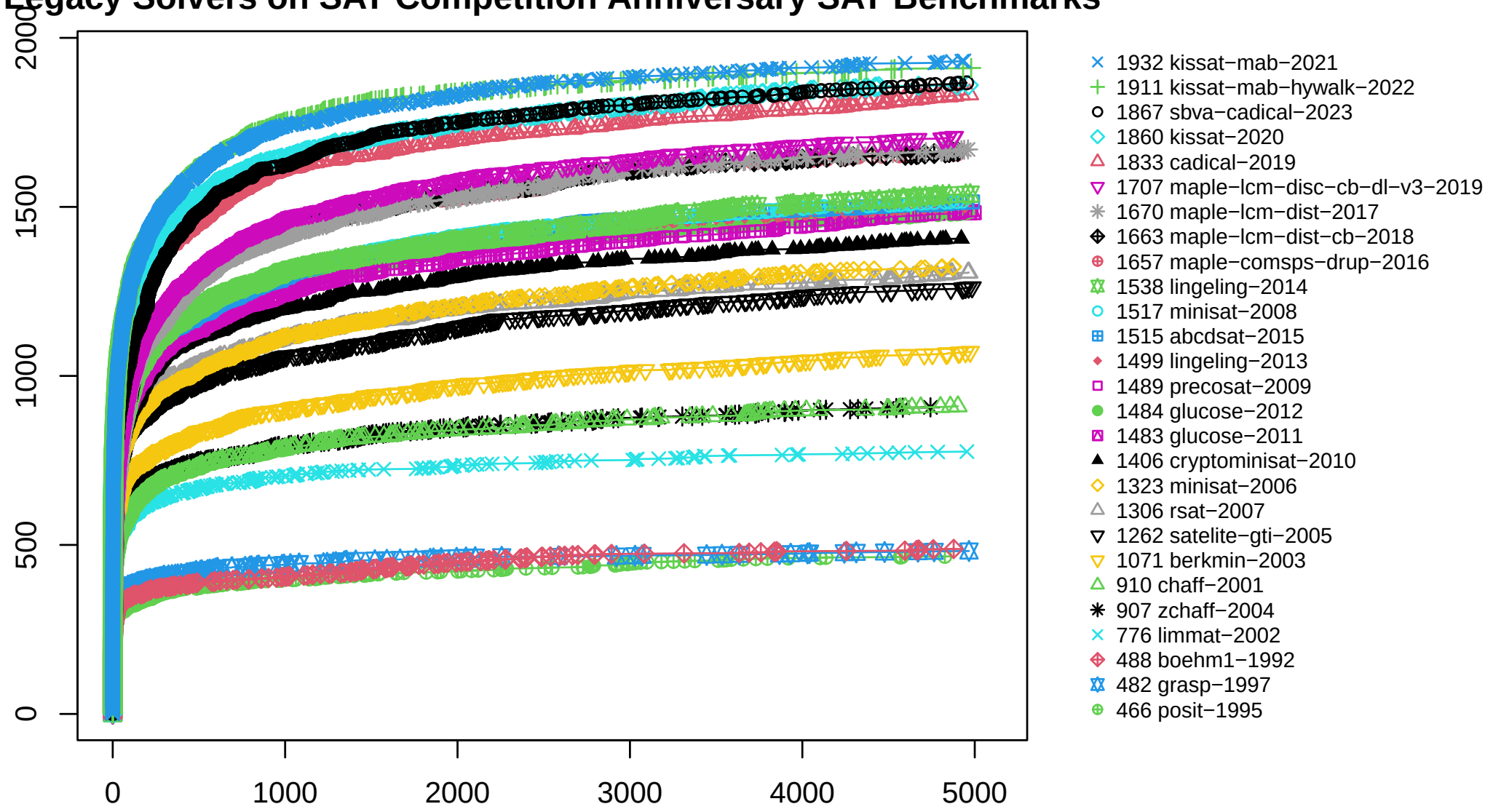


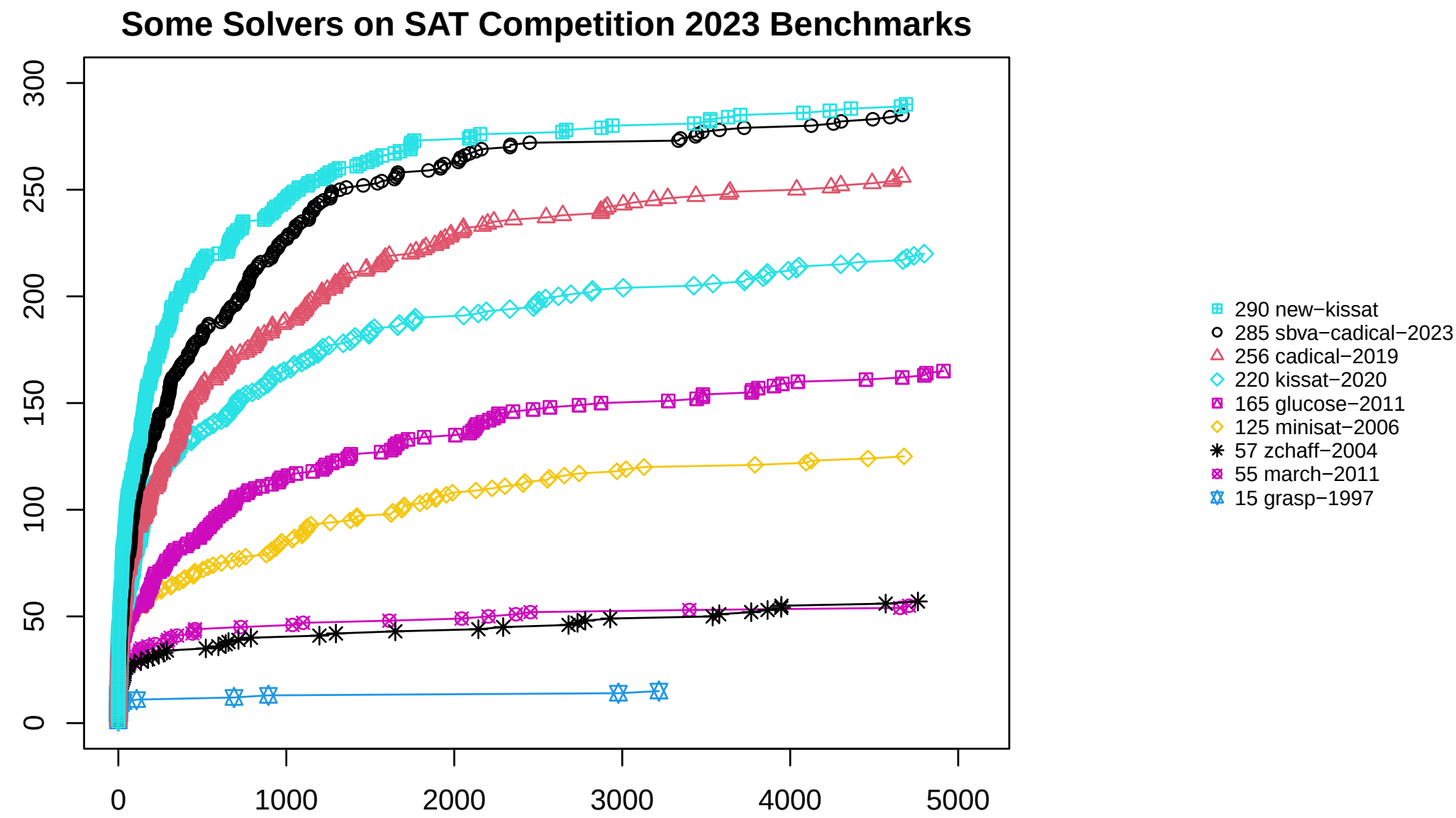
Fig. 2: Performance profiles for solver modifications on the `anni` benchmark set show that TaSSAT significantly outperforms the others. Since all solvers can quickly solve 600 instances, we start the y-axis at 600 to improve readability.

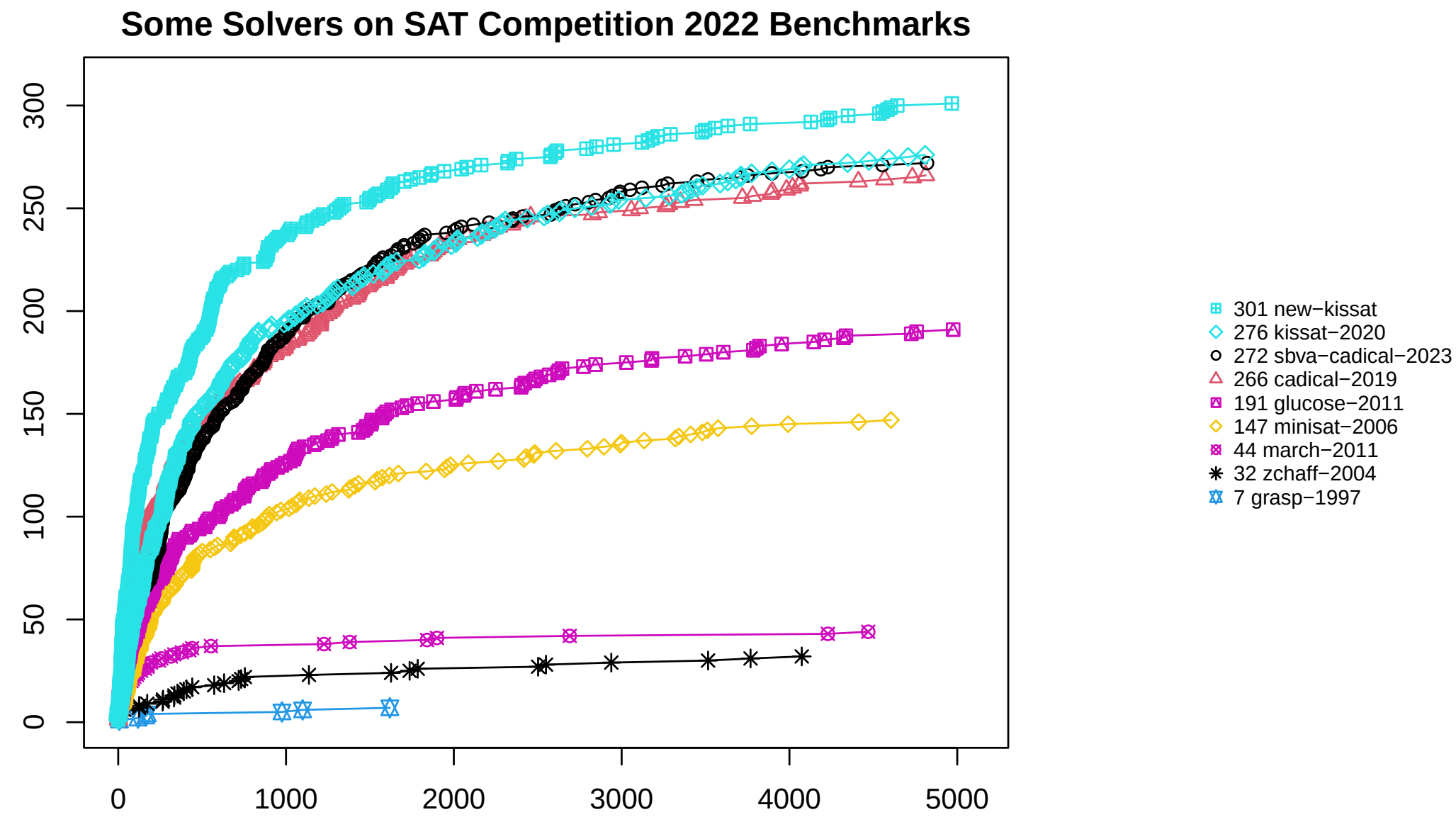
Legacy Solvers on SAT Competition Anniversary SAT Benchmarks



Conclusion

- SAT solvers get faster and faster
  - SAT museum shows steady progress in the last 30 years
- every 3-5 years quantum leaps in performance improvement
  - due to clever new algorithms and search heuristics
  - pushed forward by the yearly SAT competition and
  - by a steady increase of applications
- old and new ideas have to be revisited
  - check-out 2nd edition of the “Handbook of Satisfiability” [BiereHeuleMaarenWalsh’21]
  - intuitions need to be checked empirically
  - we also need to explain the progress better
- local-search reappeared as interesting research topic


Some Solvers on SAT Competition 2023 Benchmarks
0
50
100
150
200
250
300
0
1000
2000
3000
4000
5000
290 new−kissat
285 sbva−cadical−2023
256 cadical−2019
220 kissat−2020
165 glucose−2011
125 minisat−2006
57 zchaff−2004
55 march−2011
15 grasp−1997


Some Solvers on SAT Competition 2022 Benchmarks
0
50
100
150
200
250
300
0
1000
2000
3000
4000
5000
301 new−kissat
276 kissat−2020
272 sbva−cadical−2023
266 cadical−2019
191 glucose−2011
147 minisat−2006
44 march−2011
32 zchaff−2004
7 grasp−1997