CADIBACK

Extracting Backbones with CaDiCaL

Armin Biere, Nils Froleyks, Wenxi Wang

2023-07-07

Backbone

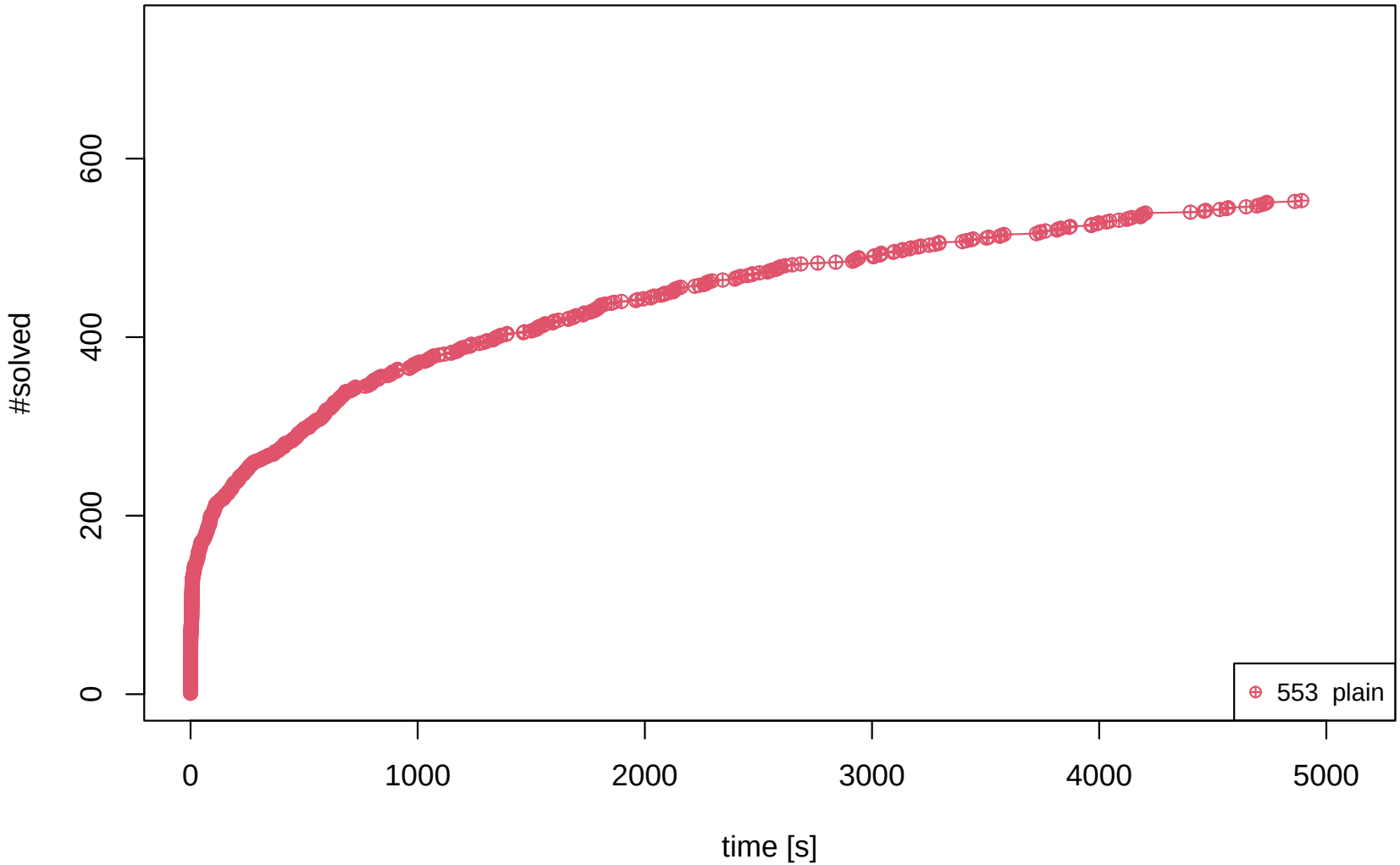
The backbone of a satisfiable formula is the set of literals that are true in all satisfying assignments.

$$a \quad \wedge \quad \overline{a} \vee b \quad \wedge \quad \overline{a} \vee \overline{b} \vee c \quad \wedge$$
$$d \vee e \vee f \quad \wedge \quad \overline{d} \vee e \vee f \quad \wedge \quad d \vee \overline{e} \vee f \quad \wedge \quad \overline{d} \vee \overline{e} \vee f \quad \wedge$$
$$\mathrm{PHP} \vee g$$

Backbone Extraction

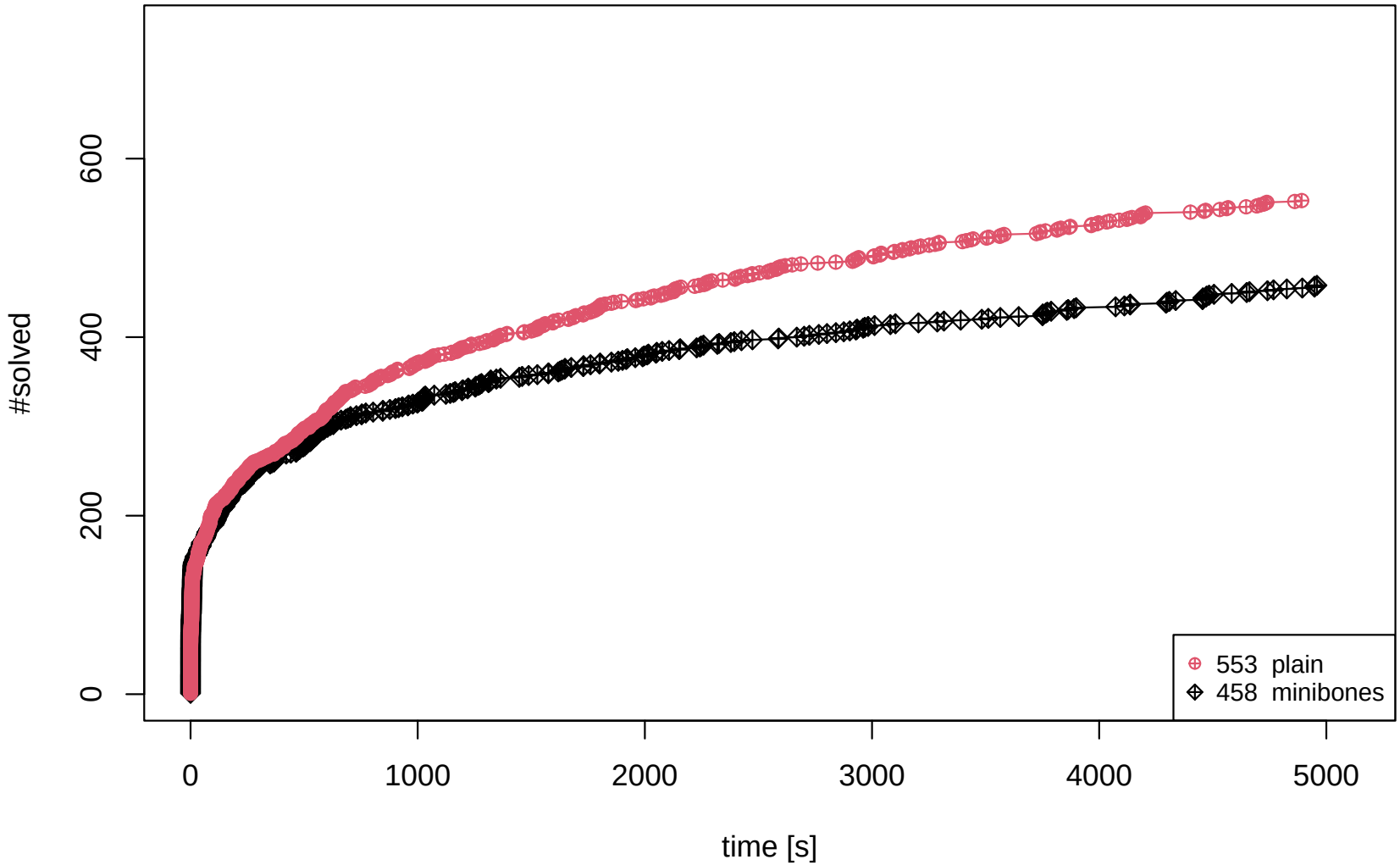
backbone (CNF φ)

1 $(res, \sigma) \leftarrow \mathsf{SAT}(\varphi)$
2 $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \sigma$
3 **while** $\Lambda \neq \emptyset$ **do**
4 $\quad \ell \leftarrow$ pick literal from Λ
5 $\quad (res, \sigma) \leftarrow \mathsf{SAT}(\varphi \mid \neg\ell)$
6 $\quad$ **if** res **then** // satisfiable
7 $\quad\quad \Lambda \leftarrow \Lambda \setminus \{\ell\}$
8 $\quad$ **else** // unsatisfiable
9 $\quad\quad \mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$
10 $\quad\quad \Lambda \leftarrow \Lambda \setminus \{\ell\}$
11 **return** $\mathcal{B}$


#solved
0
200
400
600
0
1000
2000
3000
4000
5000
time [s]
553 plain

Evaluation

- Benchmarks
  - SAT competition 2004 – 2022
  - remove comments, pretty print, compute hash, remove duplicates
  - 5,000 second of Kissat 3.0.0 [Biere and Fleury, 2022]
  - total 1798 benchmarks (6 GB)
  - `https://cca.informatik.uni-freiburg.de/sc04to22sat.zip`
- Baseline
  - MiniBones: Mikolás Janota, Inês Lynce, João Marques-Silva [Janota et al., 2015]
  - `-u -c 100 -i` slightly outperformed default configuration
  - ported to newer C++ compiler
  - `https://github.com/arminbiere/minibones`


#solved
0
200
400
600
0
1000
2000
3000
4000
5000
time [s]
553 plain
458 minibones

Antithesis 1: Incremental SAT solving did not improve

"Our results, obtained on a number of well-known practically significant applications, suggest that most improvements made to SAT solvers in recent years have no positive impact on the overall performance when solvers are used incrementally."

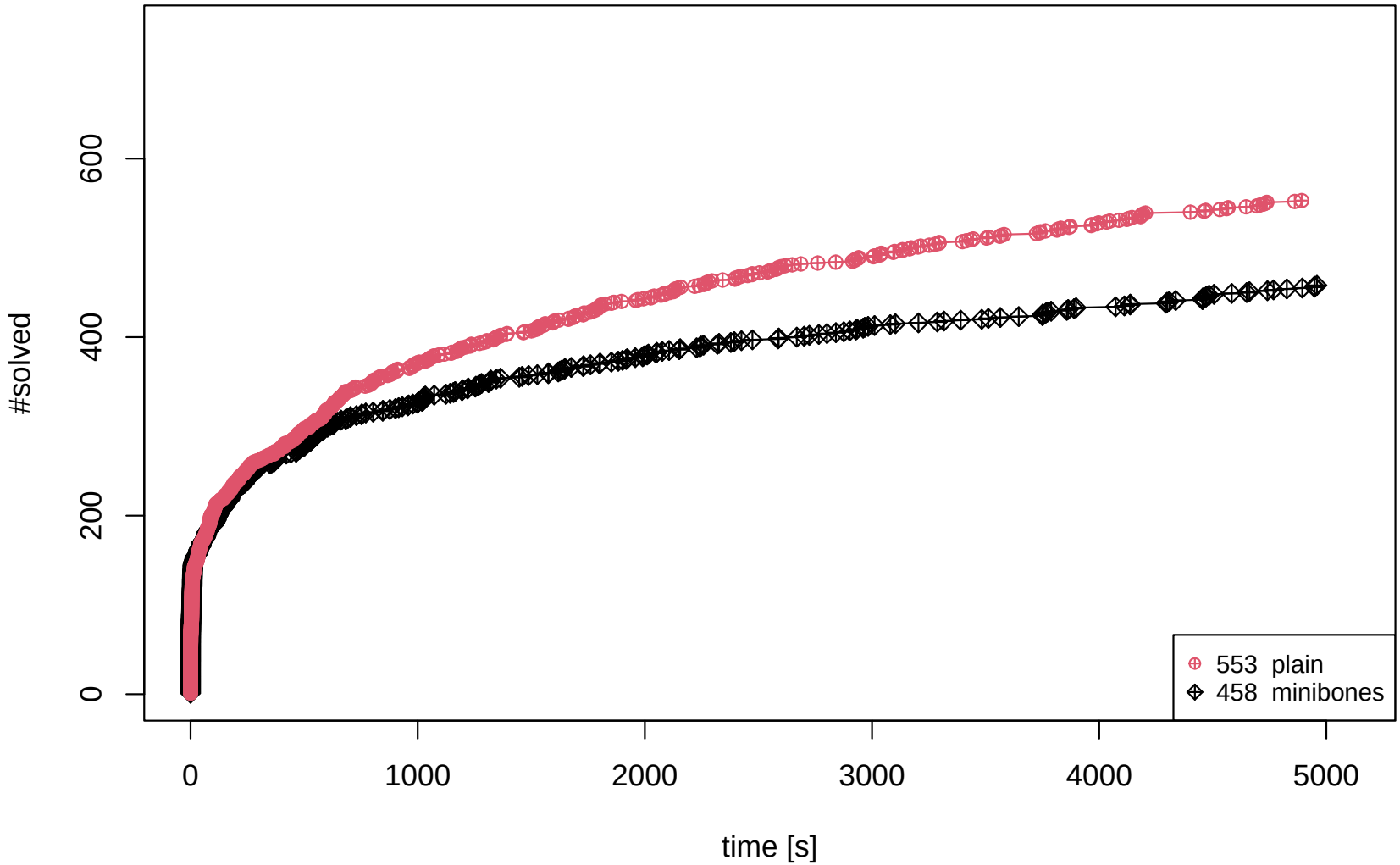
Assessing Progress in SAT Solvers Through the Lens of Incremental SAT
– Stepan Kochemazov, Alexey Ignatiev, João Marques-Silva
[Kochemazov et al., 2021]

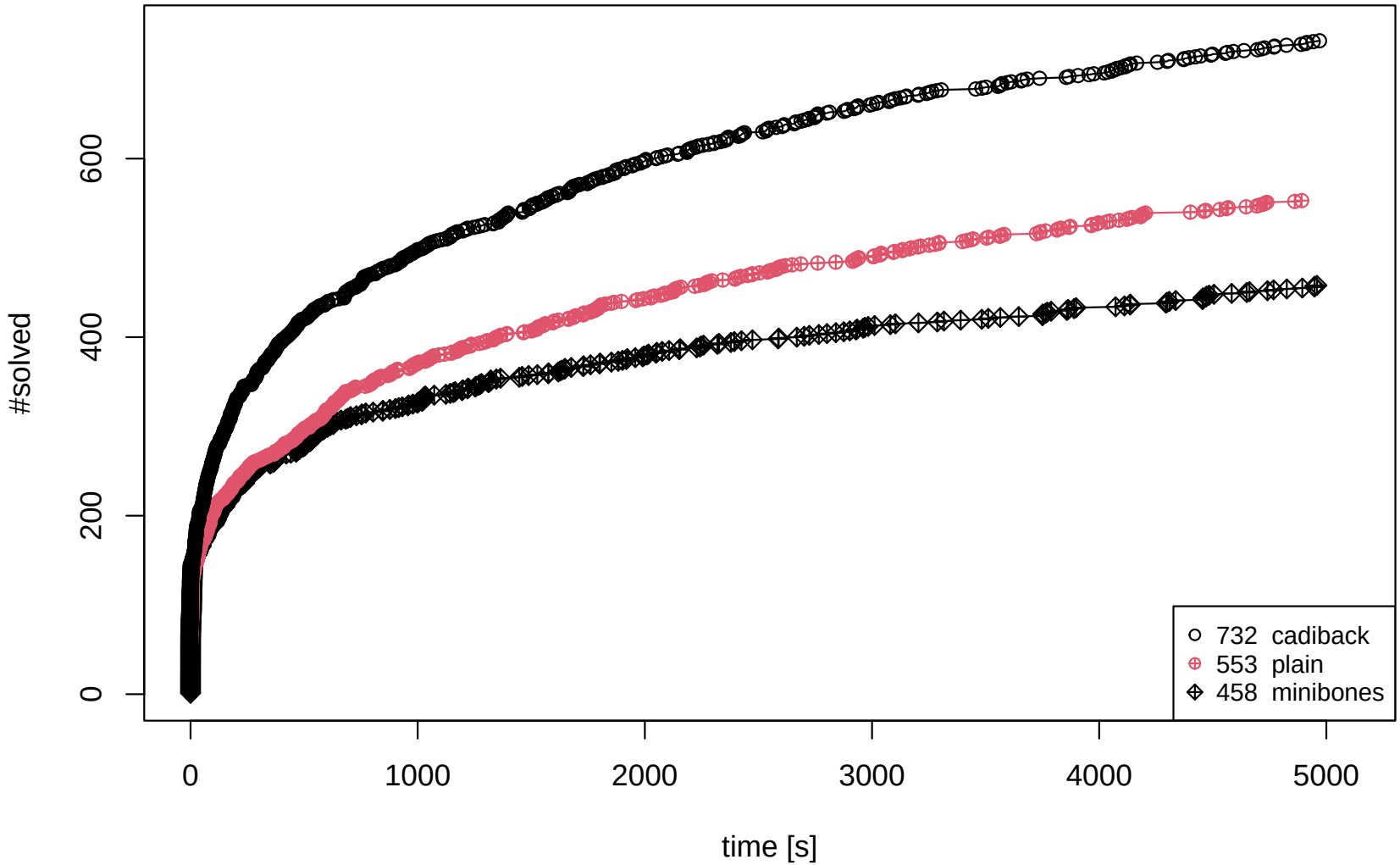
"My experience at Intel: no progress on industrial optimization problems"

Introducing Intel® SAT solver Talk at SAT'22
– Alexander Nadel
[Nadel, 2022]

"In my world there has been no progress since MiniSAT 2.2"

Discussion of the SAT Museum PoS'23
– Oliver Kullmann


#solved
0
200
400
600
0
1000
2000
3000
4000
5000
time [s]
553 plain
458 minibones


#solved
0
200
400
600
0
1000
2000
3000
4000
5000
time [s]
732 cadiback
553 plain
458 minibones

Backbone Extraction

backbone (CNF φ)

1 $(res, \sigma) \leftarrow \mathsf{SAT}(\varphi)$

2 $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \sigma$

3 **while** $\Lambda \neq \emptyset$ **do**

4 $\quad \ell \leftarrow$ pick literal from Λ

5 $\quad (res, \sigma) \leftarrow \mathsf{SAT}(\varphi \mid \neg\ell)$

6 $\quad$ **if** res **then** // satisfiable

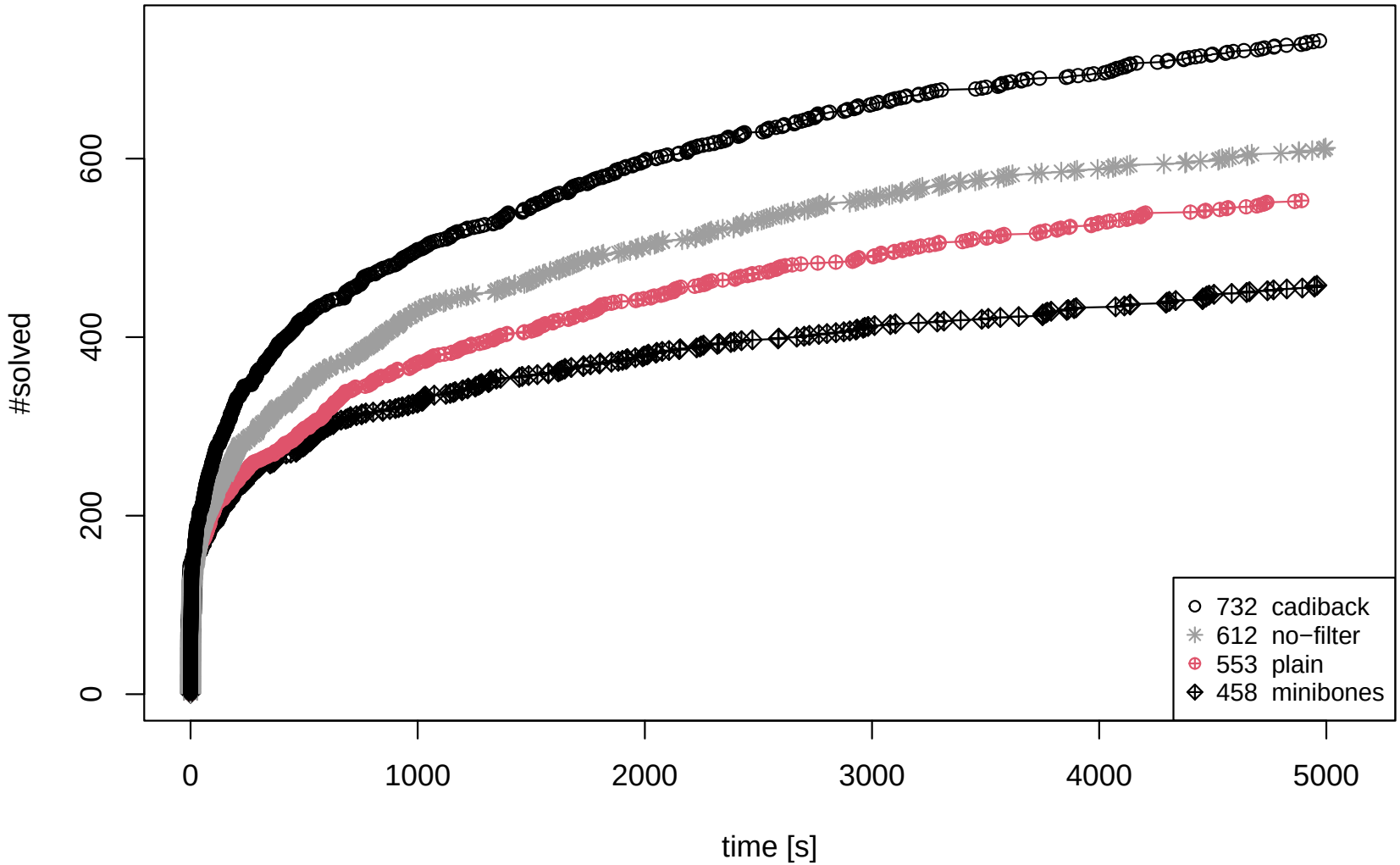
7 $\quad\quad \Lambda \leftarrow \Lambda \setminus \{\ell\}$ $\Lambda \cap \sigma$

8 $\quad$ **else** // unsatisfiable

9 $\quad\quad \mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$

10 $\quad\quad \Lambda \leftarrow \Lambda \setminus \{\ell\}$

11 **return** $\mathcal{B}$


#solved
0
200
400
600
0
1000
2000
3000
4000
5000
time [s]
732 cadiback
612 no−filter
553 plain
458 minibones

Backbone Extraction

backbone (CNF φ)

1 $(res, \sigma) \leftarrow \mathsf{SAT}(\varphi)$

2 $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \sigma, \quad k \leftarrow 1$

3 **while** $\Lambda \neq \emptyset$ **do**

4 $\quad \Gamma \leftarrow$ pick k literals from Λ // chunk

5 $\quad (res, \sigma) \leftarrow \mathsf{SAT}(\varphi \mid \cancel{\neg\ell} \bigvee_{\ell \in \Gamma} \neg\ell)$

6 $\quad$ **if** res **then** // satisfiable

7 $\quad\quad \Lambda \leftarrow \Lambda \cap \sigma, \quad k \leftarrow 1$

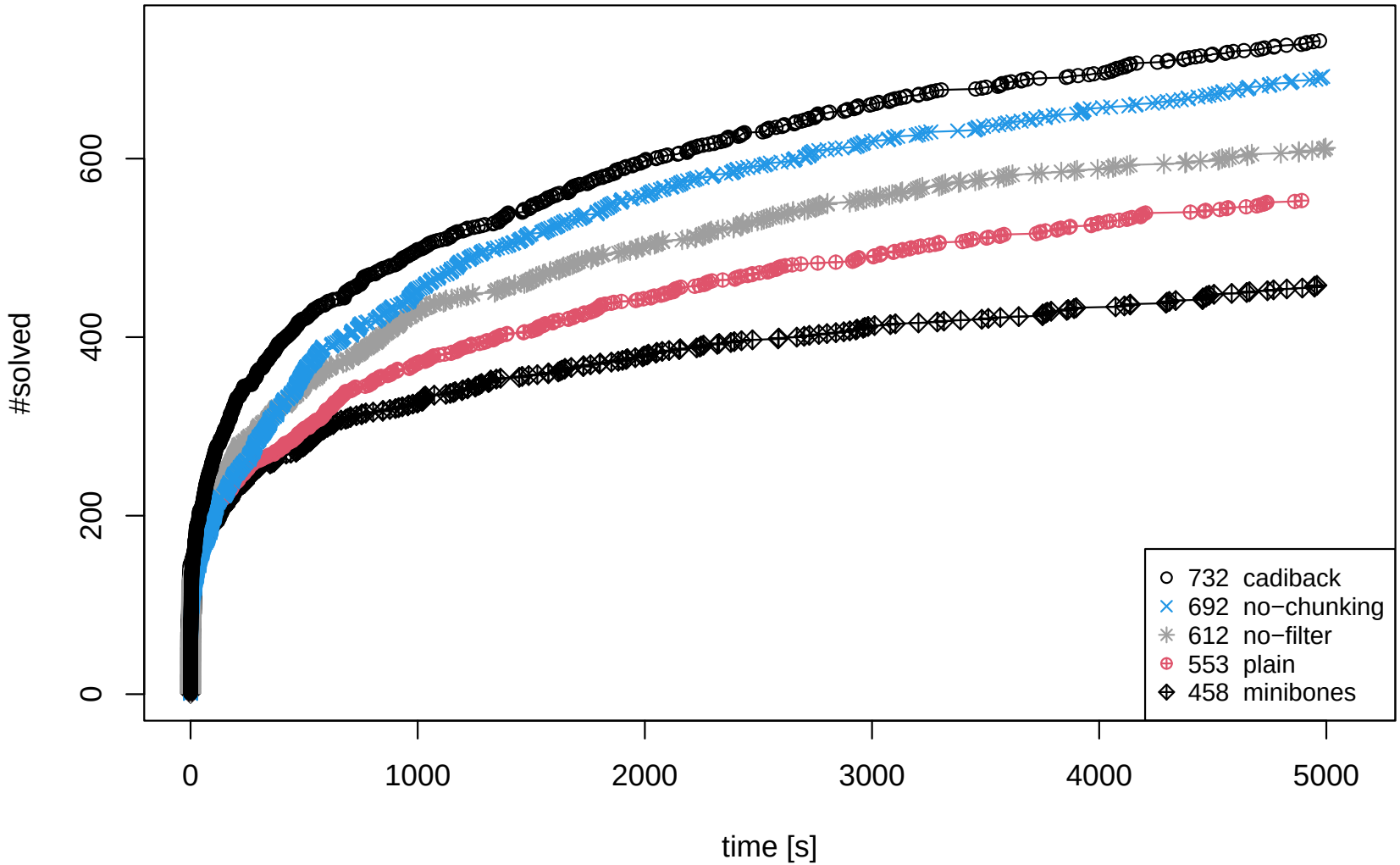
8 $\quad$ **else** // unsatisfiable

9 $\quad\quad \mathcal{B} \leftarrow \mathcal{B} \cup \cancel{\{\ell\}}\,\Gamma$

10 $\quad\quad \Lambda \leftarrow \Lambda \setminus \cancel{\{\ell\}}\,\Gamma$

11 $\quad\quad k \leftarrow |\Lambda|$

12 **return** $\mathcal{B}$


#solved
0
200
400
600
0
1000
2000
3000
4000
5000
time [s]
732 cadiback
692 no−chunking
612 no−filter
553 plain
458 minibones

Constraint

$$\mathsf{SAT}(\varphi \mid \neg\ell)$$

$$\mathsf{SAT}(\varphi \mid \bigvee_{\ell\in\Gamma} \neg\ell)$$

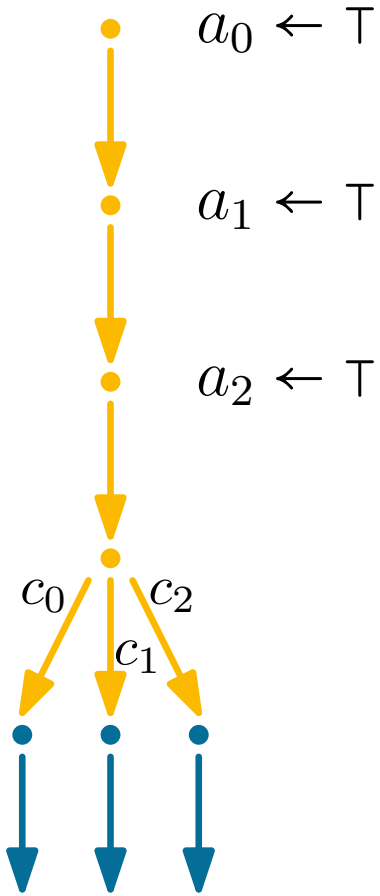
$$\underbrace{\underbrace{(\bigvee_{\ell\in\Gamma} \neg\ell \vee a)}_{\text{Clause}} \wedge \underbrace{\varphi}_{\text{Clauses}}}_{\text{Clauses}} \wedge \underbrace{\overline{a}}_{\text{Assumptions}}$$

MiniSAT [Eén and Sörensson, 2003]

$$\underbrace{(\bigvee_{\ell\in\Gamma} \neg\ell)}_{\text{Constraint}} \wedge \underbrace{\varphi}_{\text{Clauses}}$$

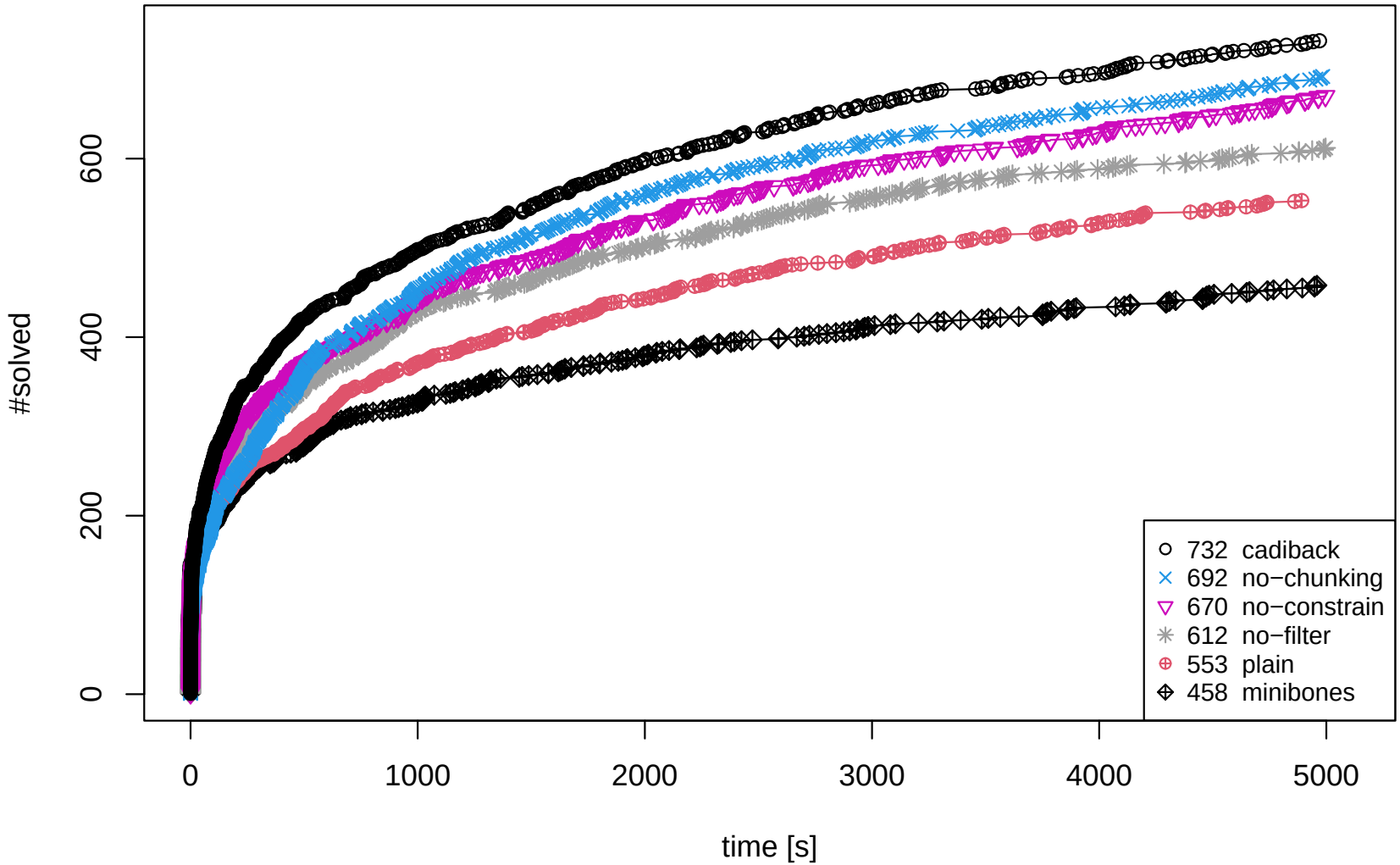
CaDiCaL [Froleyks and Biere, 2021]

Constraint



decide (constraint ρ, partial model σ)

```
1  ... // handle literal assumptions
2  if σ(ρ) = 1 then //constraint true
3      ℓ ← "first" literal in ρ with σ(ℓ)
       // speed-up future search for ℓ
4      move ℓ to the front of ρ
5  elif σ(ρ) = 0 then//constraint false
6      ... // handle conflicting constraint
7  else    // constraint undetermined
8      ℓ ← highest scored literal in σ(ρ)
9      pick ℓ as new decision and return
10 ... // fall back to default decisions
```


#solved
0
200
400
600
0
1000
2000
3000
4000
5000
time [s]
732 cadiback
692 no−chunking
670 no−constrain
612 no−filter
553 plain
458 minibones

Antithesis 2: Constraints do not work

"[. . .]improvements are not very impressive"

doctored quote from a review of our rejected
SAT'21 paper on constraints
[Froleyks and Biere, 2021]

"Initially I was not sure about the argument about *clogging* with the extra clause & assumption that required constrain()"

review of this paper

Inprocessing

Incremental Inprocessing in SAT Solving [Fazekas et al., 2019]

Best student paper award SAT'19

$$\frac{\varphi\,[\rho]\,\sigma}{\varphi\,[\rho \wedge C]\,\sigma}\ \sharp$$

LEARN$^-$

$$\frac{\varphi\,[\rho \wedge C]\,\sigma}{\varphi\,[\rho]\,\sigma}$$

FORGET

$$\frac{\varphi\,[\rho \wedge C]\,\sigma}{\varphi \wedge C\,[\rho]\,\sigma}$$

STRENGTHEN

$$\frac{\varphi\,[\rho]\,\sigma}{\varphi \wedge \Delta\,[\rho]\,\sigma}\ \mathcal{I}$$

ADDCLAUSES

$$\frac{\varphi \wedge C\,[\rho]\,\sigma}{\varphi\,[\rho]\,\sigma \cdot (\omega : C)}\ \flat$$

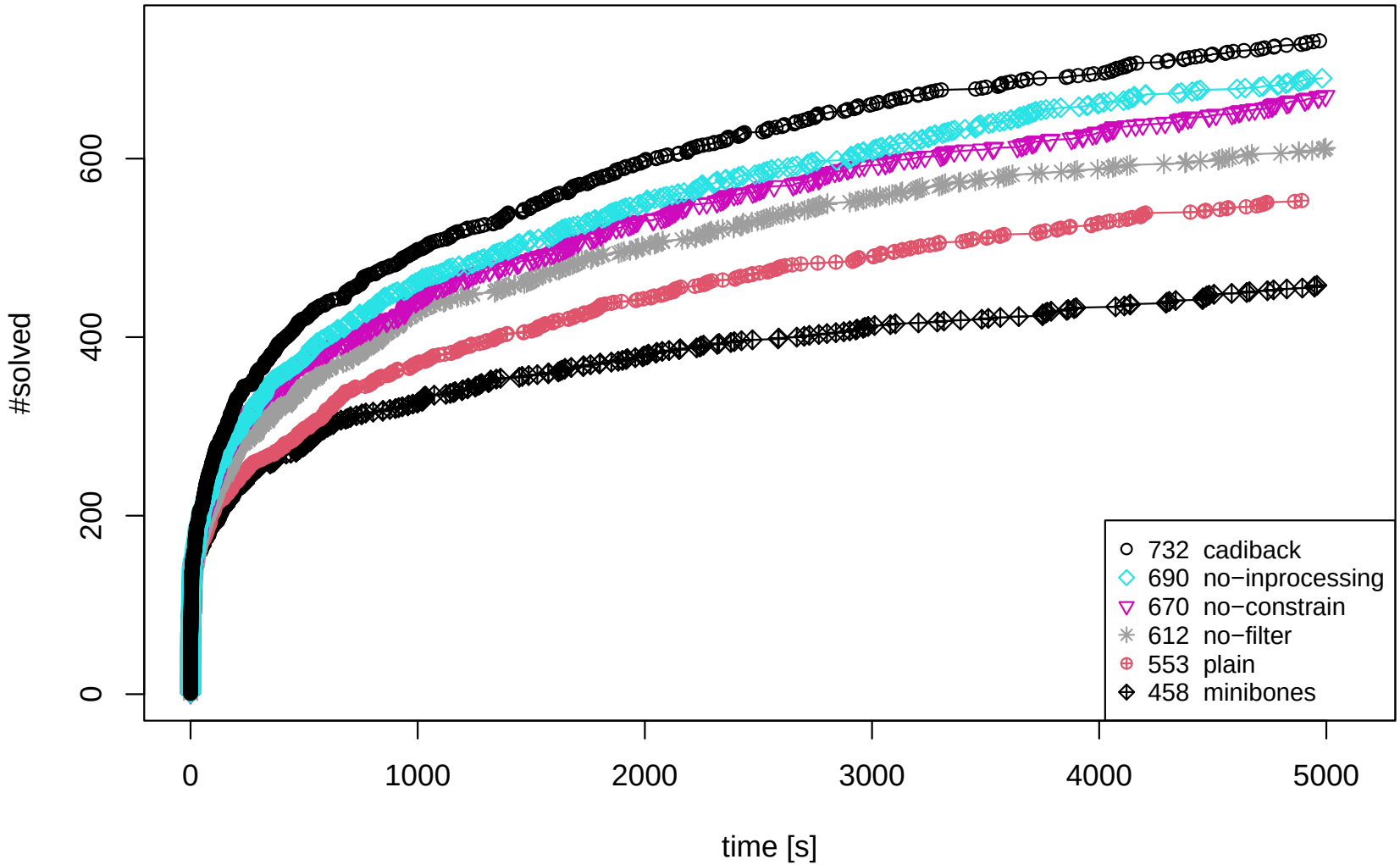
WEAKEN$^+$

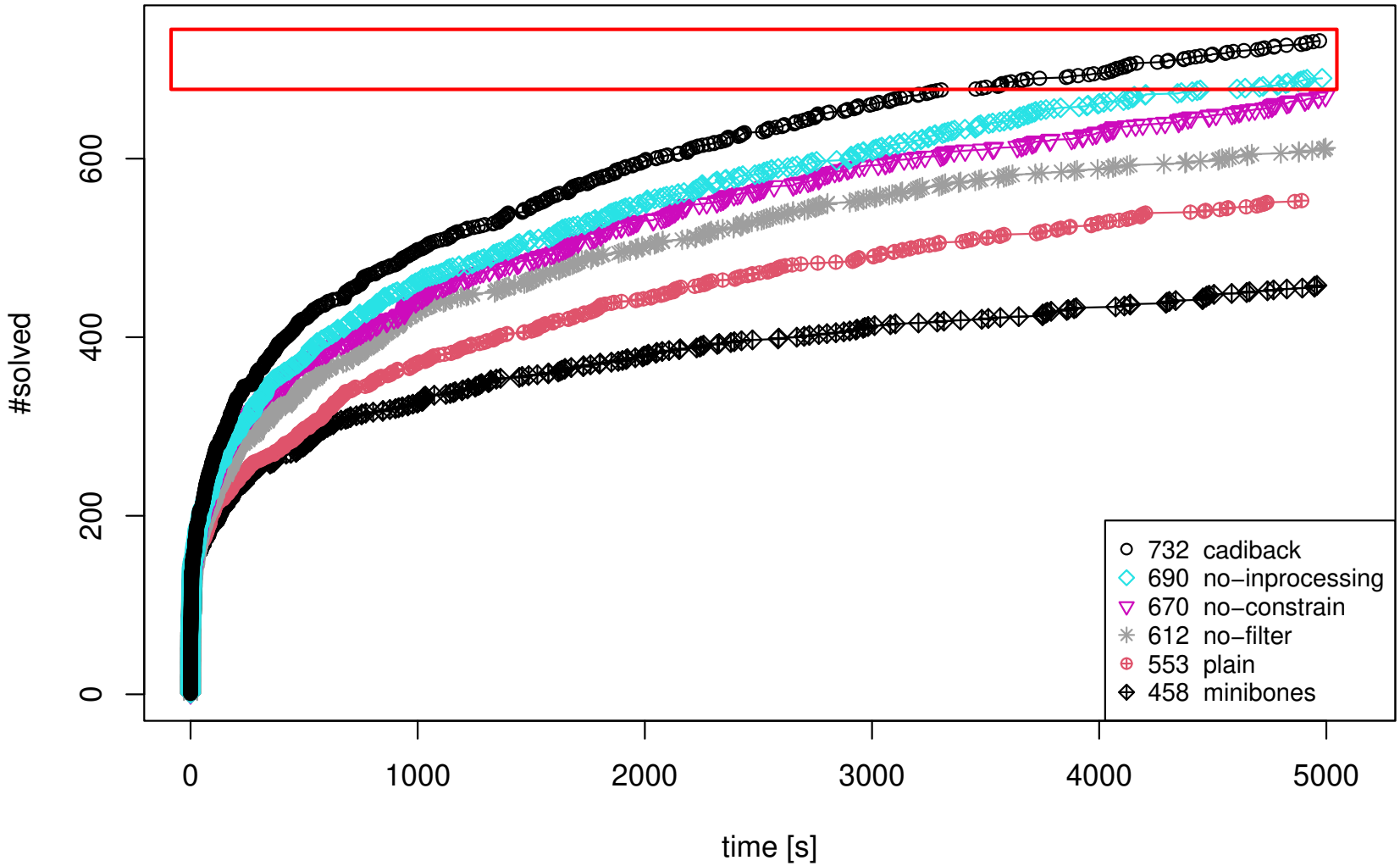
$$\frac{\varphi \wedge C\,[\rho]\,\sigma}{\varphi\,[\rho]\,\sigma}\ \text{ø}$$

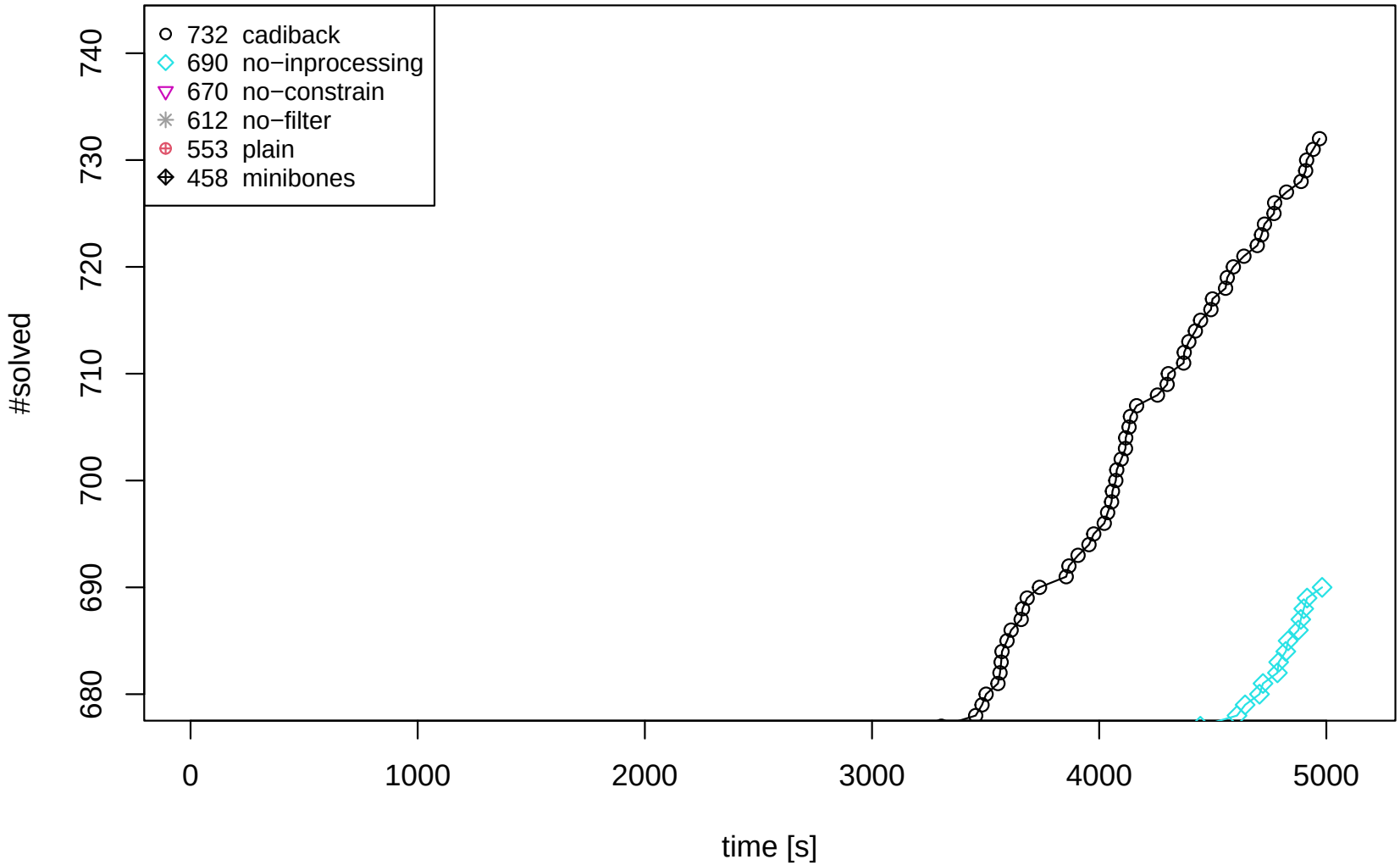
DROP

$$\frac{\varphi\,[\rho]\,\sigma \cdot (\omega : C) \cdot \sigma'}{\varphi \wedge C\,[\rho]\,\sigma \cdot \sigma'}\ \partial$$

RESTORE


#solved
0
200
400
600
0
1000
2000
3000
4000
5000
time [s]
732 cadiback
690 no−inprocessing
670 no−constrain
612 no−filter
553 plain
458 minibones


#solved
0
200
400
600
0
1000
2000
3000
4000
5000
time [s]
732 cadiback
690 no−inprocessing
670 no−constrain
612 no−filter
553 plain
458 minibones


732 cadiback
690 no−inprocessing
670 no−constrain
612 no−filter
553 plain
458 minibones
#solved
680
690
700
710
720
730
740
0
1000
2000
3000
4000
5000
time [s]

Flippable Literals via Watch Lists

Literal ℓ is flippable in model σ `iff` $\sigma \setminus \{\ell\} \cup \{\neg\ell\}$ is a model

flippable (CNF φ, literal ℓ, model σ)

1 **for** all clauses C watched by ℓ in φ

2 **if** $\sigma(C \setminus \{\ell\}) = 0$ **then return** 0

3 **return** 1

Kissat [Fleury and Biere, 2022]

Backbone Extraction

backbone (CNF φ)

1 $(res, \sigma) \leftarrow \mathsf{SAT}(\varphi)$

2 $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \sigma\{\ell \in \sigma \mid \neg\mathsf{flippable}(\ell, \sigma)\}, \quad k \leftarrow 1$

3 **while** $\Lambda \neq \emptyset$ **do**

4 $\quad \Gamma \leftarrow$ pick k literals from Λ // chunk

5 $\quad (res, \sigma) \leftarrow \mathsf{SAT}(\varphi \mid \bigvee_{\ell \in \Gamma} \neg\ell)$

6 $\quad$ **if** res **then** // satisfiable

7 $\quad\quad \Lambda \leftarrow \Lambda \cap \sigma\{\ell \in \sigma \mid \neg\mathsf{flippable}(\ell, \sigma)\}, \quad k \leftarrow 1$

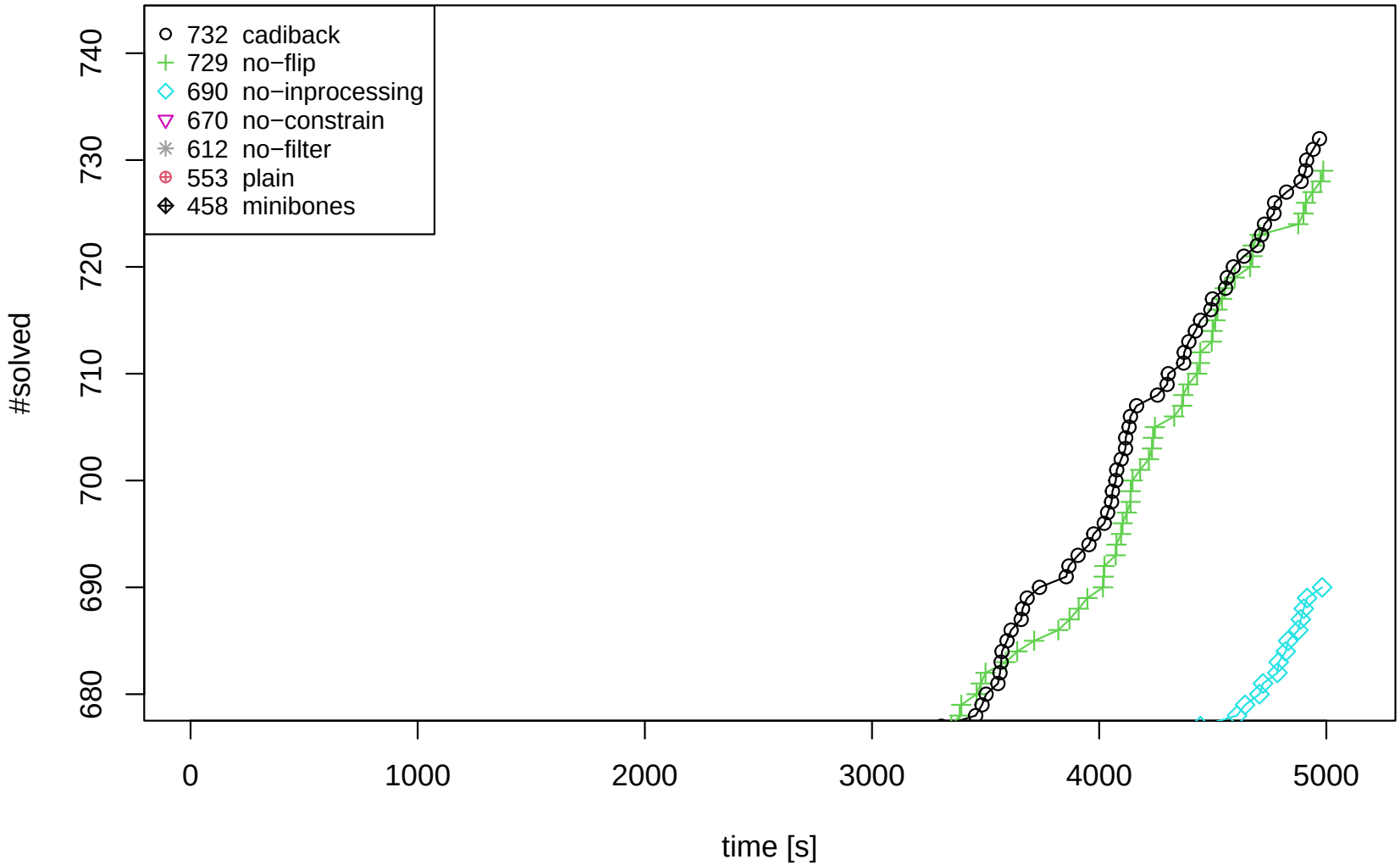
8 $\quad$ **else** // unsatisfiable

9 $\quad\quad \mathcal{B} \leftarrow \mathcal{B} \cup \Gamma$

10 $\quad\quad \Lambda \leftarrow \Lambda \setminus \Gamma$

11 $\quad\quad k \leftarrow |\Lambda|$

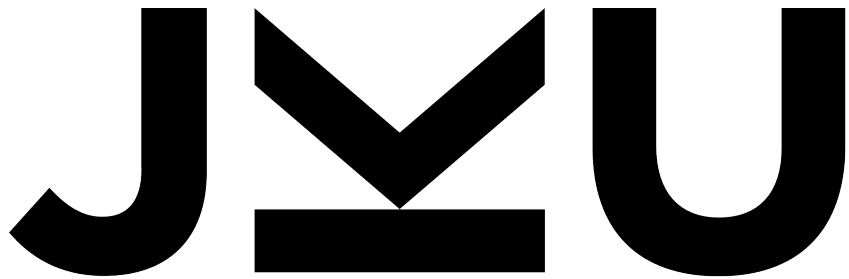
12 **return** $\mathcal{B}$


732 cadiback
729 no−flip
690 no−inprocessing
670 no−constrain
612 no−filter
553 plain
458 minibones
#solved
680
690
700
710
720
730
740
0
1000
2000
3000
4000
5000
time [s]

Antithesis 3: Flippable literals do not work

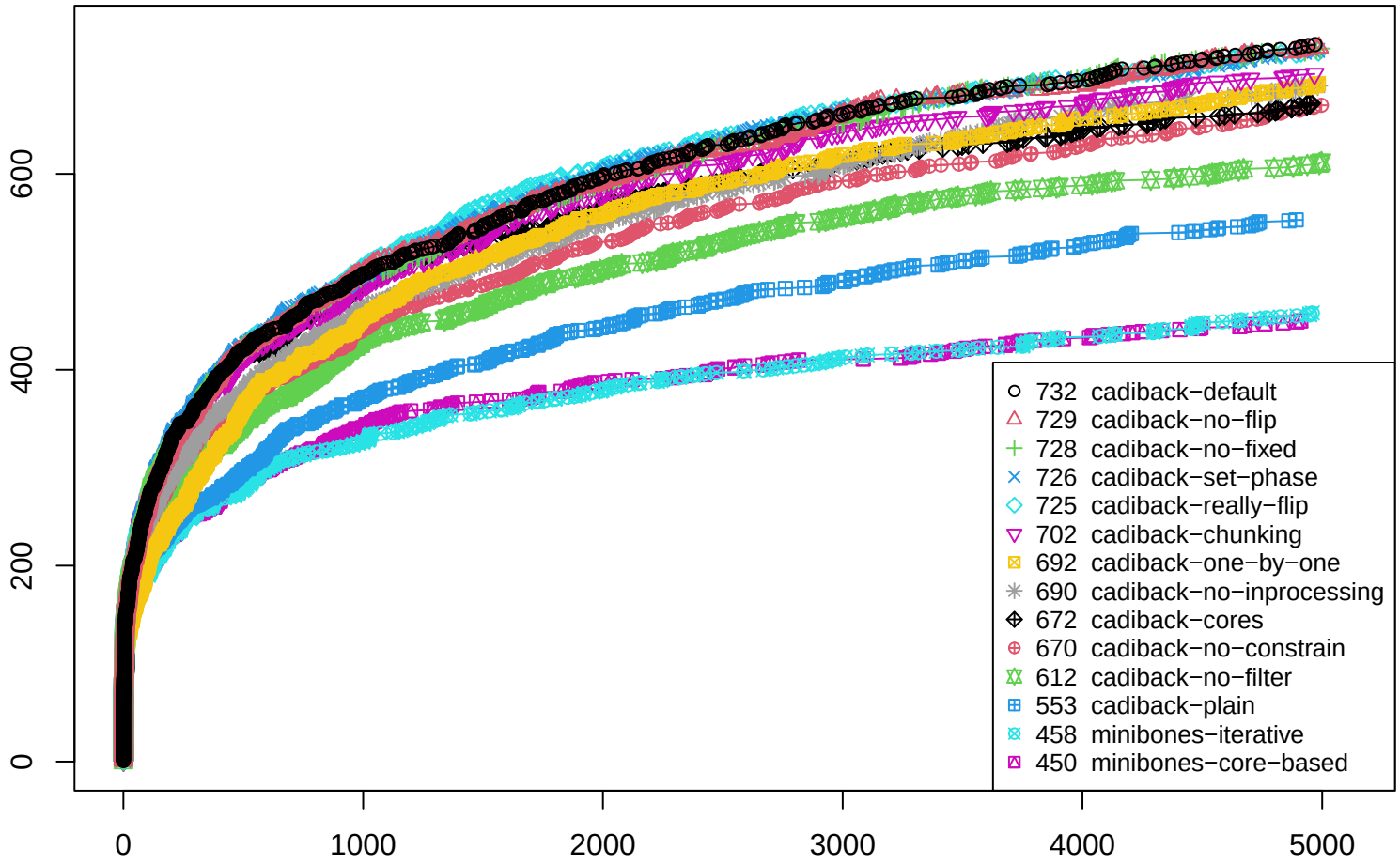
"However, the rotatable [flippable] literals technique yields generally a slower algorithm than the one without it."

Algorithms for Computing Backbones of Propositional Formulae
– Mikolás Janota, Inês Lynce, João Marques-Silva
[Janota et al., 2015]

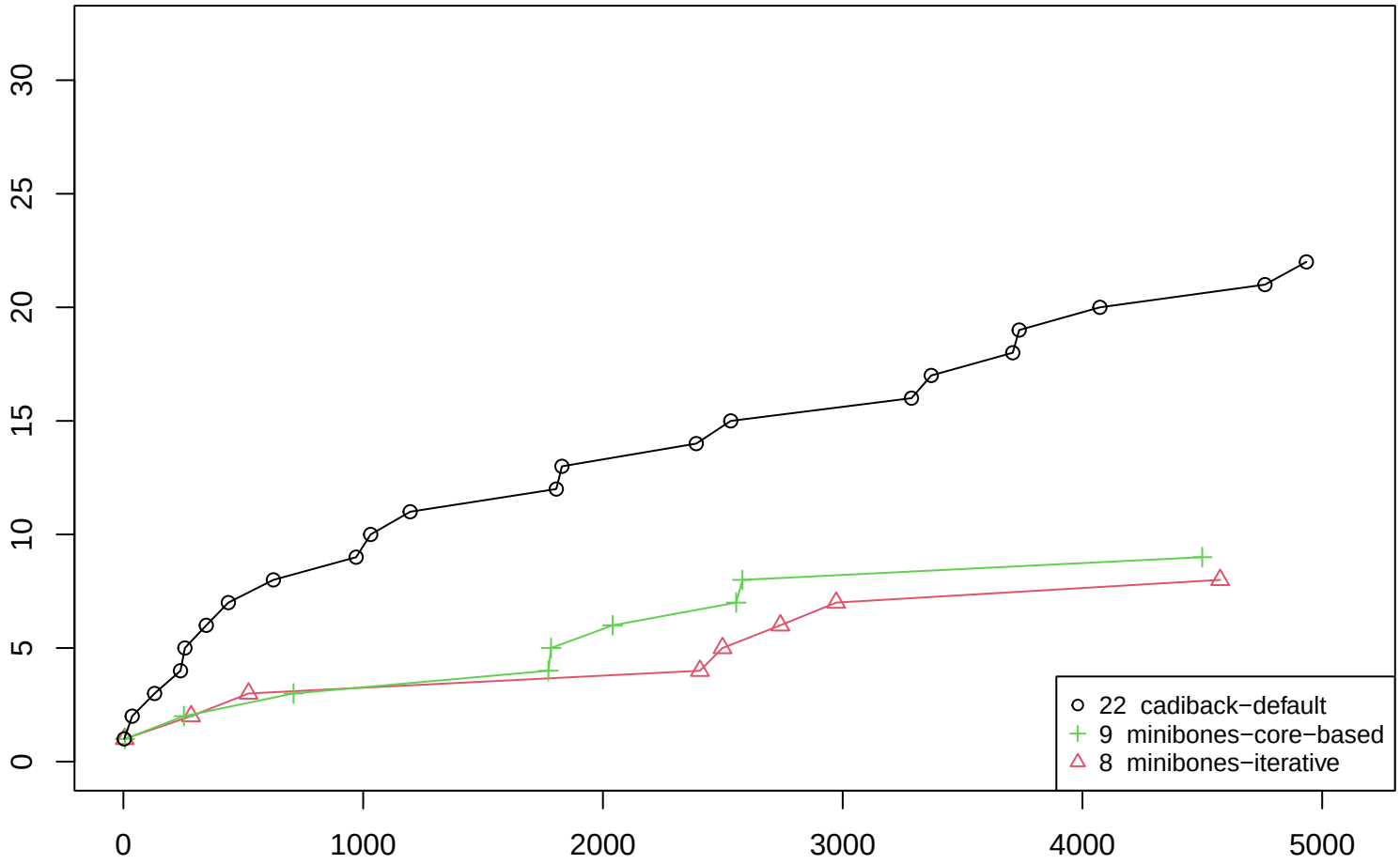


JOHANNES KEPLER
UNIVERSITY LINZ

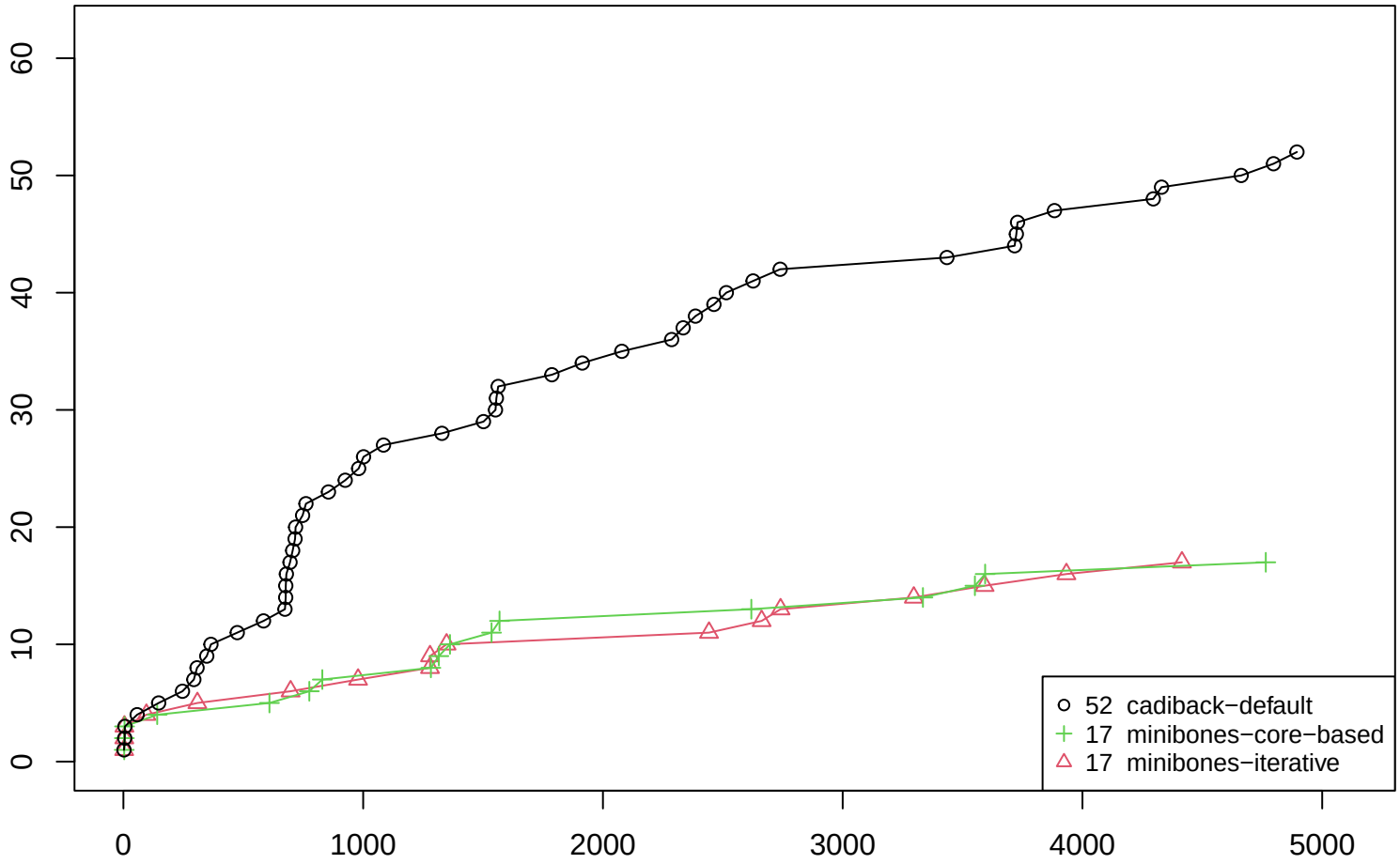
Full CDF


0
200
400
600
0
1000
2000
3000
4000
5000
732 cadiback−default
729 cadiback−no−flip
728 cadiback−no−fixed
726 cadiback−set−phase
725 cadiback−really−flip
702 cadiback−chunking
692 cadiback−one−by−one
690 cadiback−no−inprocessing
672 cadiback−cores
670 cadiback−no−constrain
612 cadiback−no−filter
553 cadiback−plain
458 minibones−iterative
450 minibones−core−based

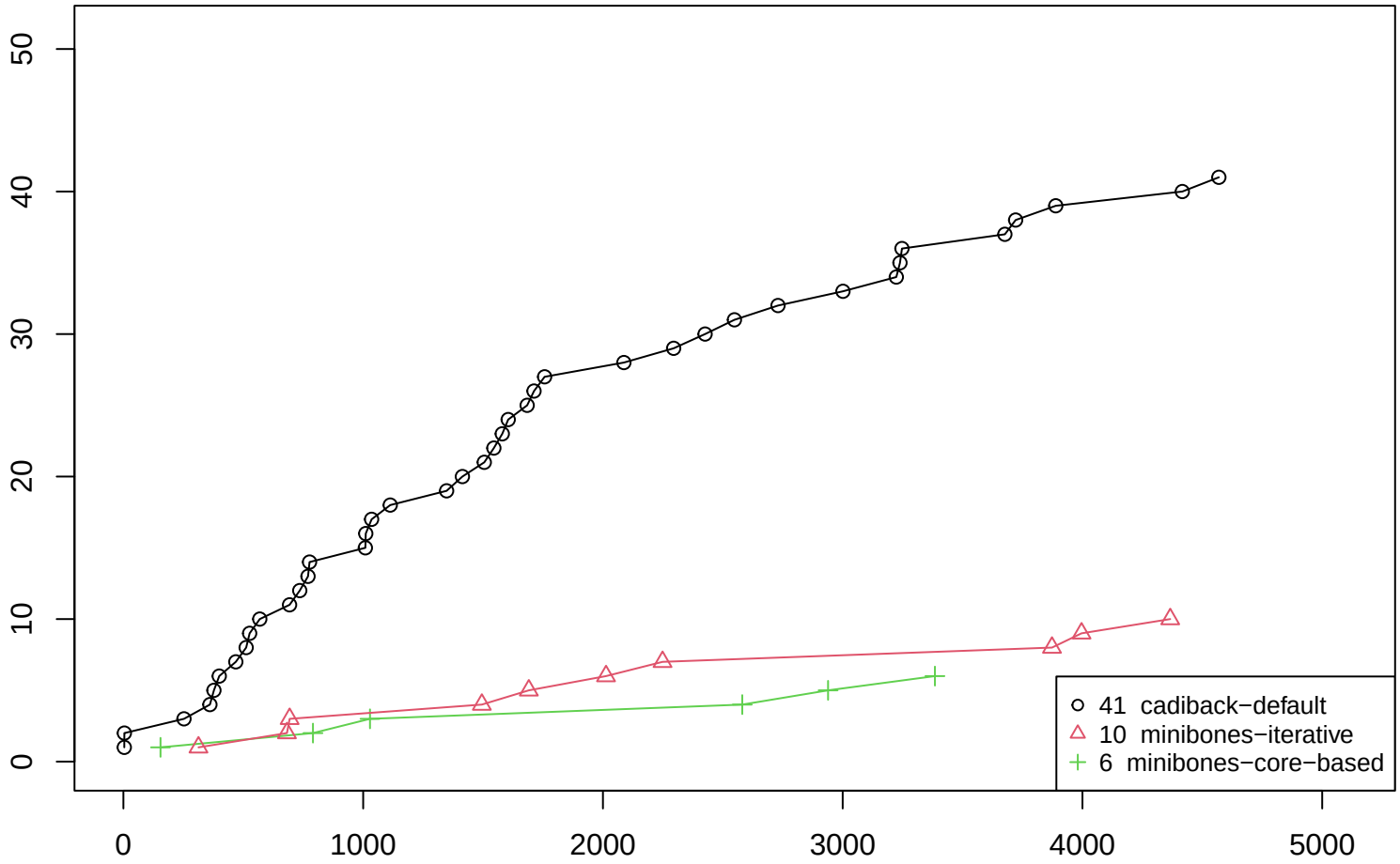
SC2020


0
5
10
15
20
25
30
0
1000
2000
3000
4000
5000
22 cadiback−default
9 minibones−core−based
8 minibones−iterative

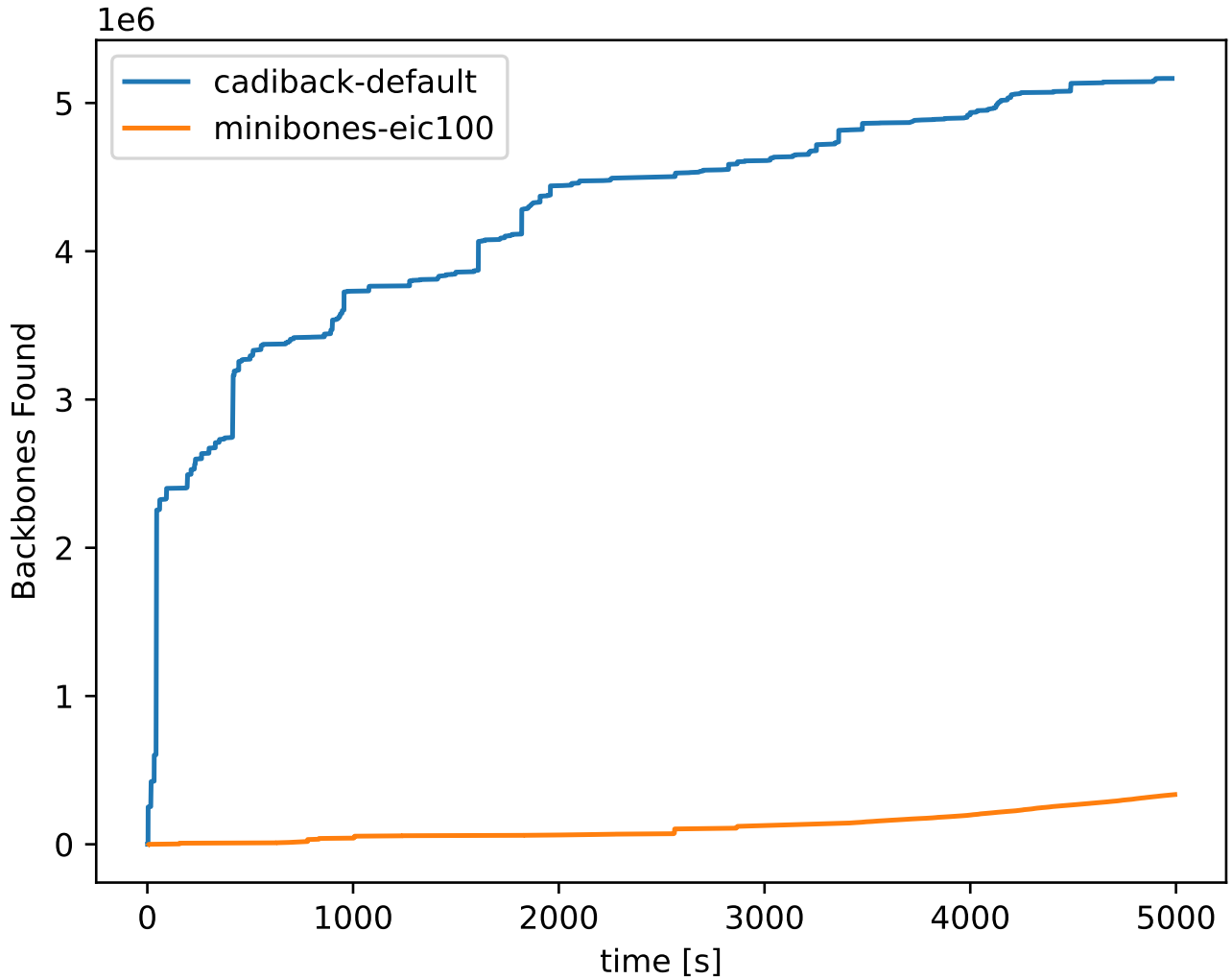
SC2021


0
10
20
30
40
50
60
0
1000
2000
3000
4000
5000
52 cadiback−default
17 minibones−core−based
17 minibones−iterative

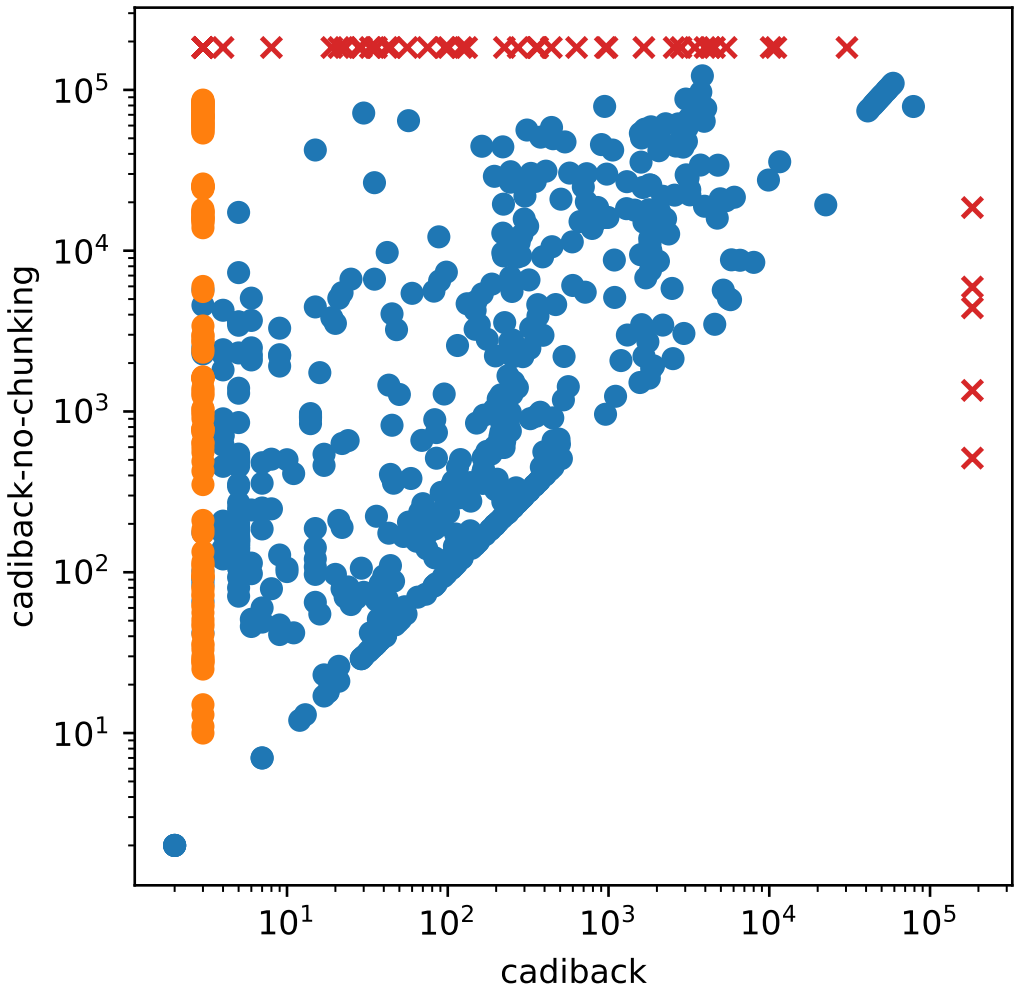
SC2022


0
10
20
30
40
50
0
1000
2000
3000
4000
5000
41 cadiback−default
10 minibones−iterative
6 minibones−core−based

Identified Backbones in SC2022



Number of SAT Calls



References I

[Biere and Fleury, 2022] Biere, A. and Fleury, M. (2022).
Gimsatul, IsaSAT and Kissat entering the SAT Competition 2022.
In Balyo, T., Heule, M., Iser, M., Järvisalo, M., and Suda, M., editors, *Proc. of SAT Competition 2022 – Solver and Benchmark Descriptions*, volume B-2022-1 of *Department of Computer Science Series of Publications B*, pages 10–11. University of Helsinki.

[Eén and Sörensson, 2003] Eén, N. and Sörensson, N. (2003).
Temporal Induction by Incremental SAT Solving.
Electronic Notes in Theoretical Computer Science, 89(4):543–560.

[Fazekas et al., 2019] Fazekas, K., Biere, A., and Scholl, C. (2019).
Incremental Inprocessing in SAT Solving.
In Janota, M. and Lynce, I., editors, *Theory and Applications of Satisfiability Testing – SAT 2019*, Lecture Notes in Computer Science, pages 136–154, Cham. Springer International Publishing.

References II

[Fleury and Biere, 2022] Fleury, M. and Biere, A. (2022).
GIMSATUL, ISASAT, KISSAT.
SAT COMPETITION 2022, page 10.

[Froleyks and Biere, 2021] Froleyks, N. and Biere, A. (2021).
Single Clause Assumption without Activation Literals to Speed-up IC3.
In *2021 Formal Methods in Computer Aided Design (FMCAD)*, pages 72–76.

[Janota et al., 2015] Janota, M., Lynce, I., and Marques-Silva, J. (2015).
Algorithms for computing backbones of propositional formulae.
AI Commun., 28(2):161–177.

References III

[Kochemazov et al., 2021] Kochemazov, S., Ignatiev, A., and Marques-Silva, J. (2021).
Assessing Progress in SAT Solvers Through the Lens of Incremental SAT.
In Li, C.-M. and Manyà, F., editors, *Theory and Applications of Satisfiability Testing - SAT 2021 - 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings*, volume 12831 of *Lecture Notes in Computer Science*, pages 280–298. Springer.

[Nadel, 2022] Nadel, A. (2022).
Introducing Intel(R) SAT Solver.
In Meel, K. S. and Strichman, O., editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, pages 8:1–8:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.