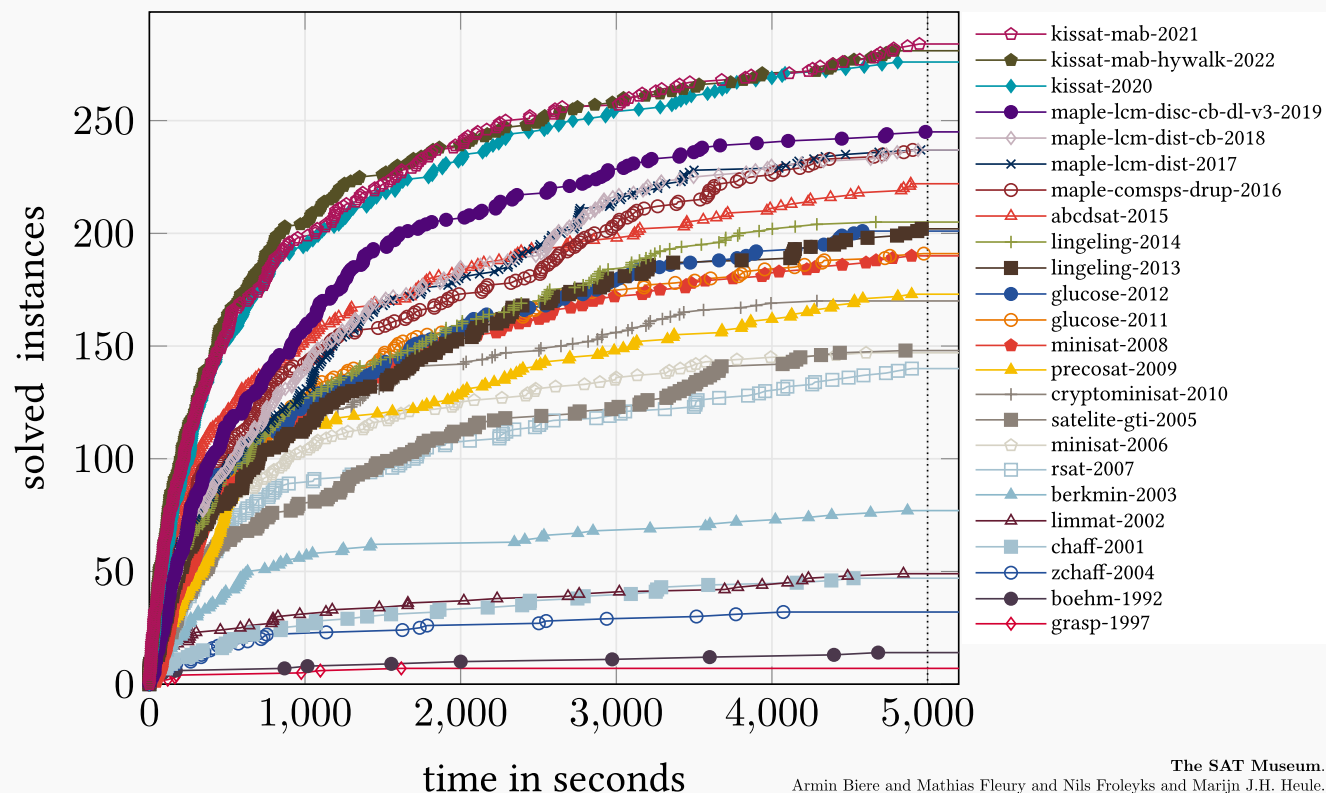


Solvers based on SAT are...

- Improving each year
- Competition winning
 - MiniZinc, XCSP3
- Versatile
 - Industrial applications
 - Incremental solving
 - Model counting
 - Explainability
 - Proof-logging
- Hard to use

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 Froyleys 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]

Pindakaas - A SAT encoding library

$$\begin{aligned} p, q, r &\in \mathbb{B} \\ p &\oplus q \\ 2p + 3q + 5r &\leq 6 \end{aligned}$$

model

```
import pindakaas as pdk
cadical = pdk.CaDiCaL()
p, q, r = cadical.new_vars(3)
cadical += p ^ q
cadical += 2*p + 3*q + 5*r <= 6
```

encode

$$(p \vee q) \wedge (\neg p \vee \neg q) \wedge \dots$$

solve

$$p \wedge \neg q \wedge \neg r$$

When solving high-level problems with low-level SAT solvers, the quality of the encoding is **critical** for solver performance

Features

- Encode high-level constraints
- Solve incrementally
- Integration with **CPMpy** and **Huub**



Rust implementation



Python interface