Factoring Learned Clauses

Florian Pollitt[1] Zach Battleman[2] Mathias Fleury[1] Yakir Vizel[3]

Marijn Heule[2] Armin Biere[1] Randy Bryant[2]

[1] universität freiburg [2] Carnegie Mellon University

[3] TECHNION Israel Institute of Technology

SAT 2026

July 20.-23., 2026, Lisbon, Portugal

29th International Conference on
Theory and Applications of Satisfiability Testing

supported by Co-funded by the European Union DFG Intel

Factoring Learned Clauses — Story

- beyond resolution: extended resolution (ER) [CDPC'83]
- blocked clauses [DAM'99]
- BVA [HVC'12]
- only resolution in SAT competition solvers until 2022
- BVA winning the competition since 2023:
  - SBVA-CADICAL SC2023, KISSAT SC2024, AE KISSAT MAB SC2025
- ER works with incremental SAT [POS'25]
- ER techniques in PDR (for proofs) [CAV'25]
- BVA officially in cadical [SAT'26] (Tuesday)
- logical step: revisit ER techniques

Factoring Learned Clauses — Contributions

- revisited GLUCOSER [AAAI'10]:
  - efficient reimplementation: CADICAL-ER
- extending BVA: CADICAL-FX
  - Advanced Gate Factorization: XORs/ITEs
  - inprocessing with learned clauses
- common strength:
  - Tseitin formulas
- polynomial proofs: CADICAL-PP

CaDiCaL-ER

- definition $x \leftrightarrow (a \wedge d)$
  - CNF: $(a \vee b \vee c) \wedge (b \vee c \vee d)$
  - NNF: $(a \wedge d) \vee (b \vee c)$
  - substitute $x \rightarrow (a \wedge d)$: $(\overline{x} \vee a) \wedge (\overline{x} \vee d) \wedge (x \vee b \vee c)$
  - $x \leftarrow (a \wedge d) = (x \vee \overline{a} \vee \overline{d})$ is redundant (blocked)
- detect and apply during conflict analysis
  - detect on consecutive clauses
  - apply to recently learned clauses

CaDiCaL-ER — Discussion

- challenges:
  - mangage active definitions
- strengths:
  - reactive
  - can be very effective (mutilated chessboard)
- weaknesses:
  - overhead during clause learning
  - does not pay off in general

Advanced Gate Factorization — CaDiCaL-FX

- extension of BVA
- factor $x \leftrightarrow XOR(a,b)$ and $x \leftrightarrow ITE(c,t,f)$
- detect and apply during inprocessing
- crucial: consider learned clauses

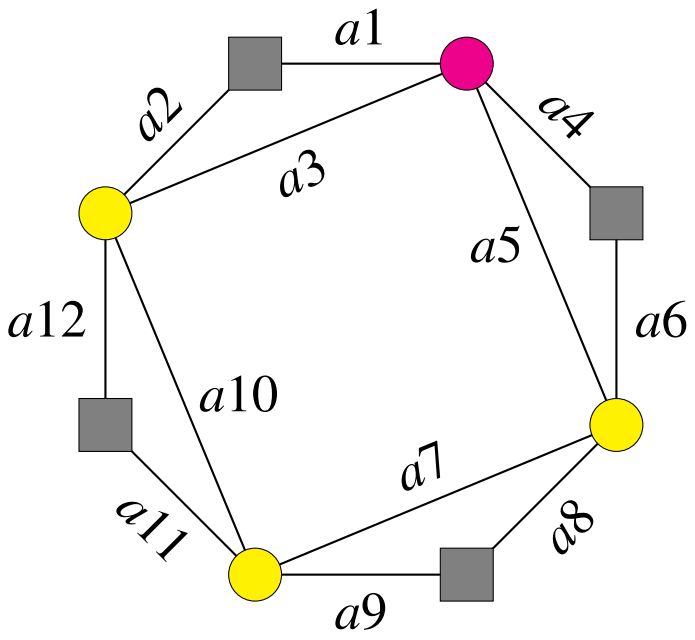
Advanced Gate Factorization — Extraction

- $x \leftrightarrow ITE(c,t,f)$: $(c \rightarrow (x \leftrightarrow t)) \wedge (\overline{c} \rightarrow (x \leftrightarrow f))$
- pick c, then for all t, f:
- match $(C \vee \overline{c} \vee t)$ with $(C \vee c \vee f)$
- choose best pattern to reencode (smallest result)
- introduce definition: $(x \vee \overline{c} \vee \overline{t}), (\overline{x} \vee \overline{c} \vee t), (\overline{x} \vee c \vee f), (x \vee c \vee \overline{f})$
- substitute into pairs: $C \vee x$
- special case: $ITE(c,t,\overline{t}) = XOR(c,t)$

Advanced Gate Factorization — Discussion

- challenges:
  - efficient gate extraction
- strengths:
  - less intrusive
  - little-to-no overhead in general
- weaknesses:
  - relies on gate structures

Tseitin Formula — Graph



$XOR(a1, a3, a4, a5)$

$XOR(a2, a3, a10, a12)$

$XOR(a5, a6, a7, a8)$

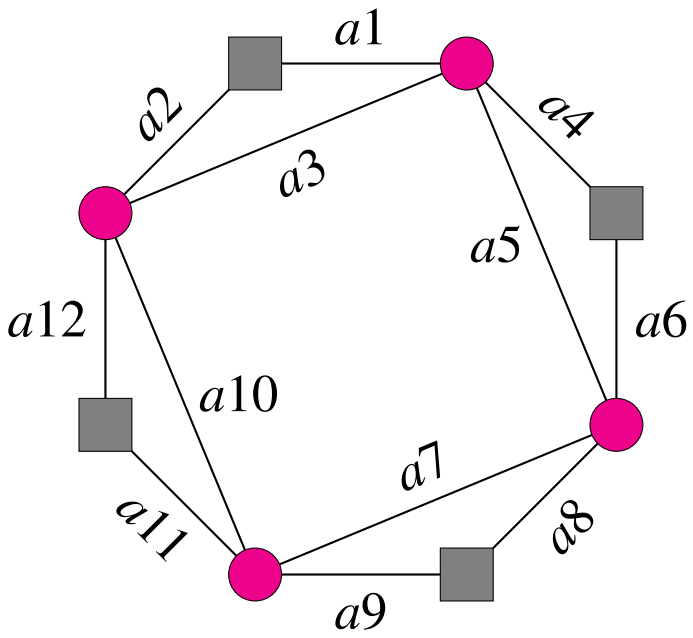
$XOR(a7, a9, a10, a11)$

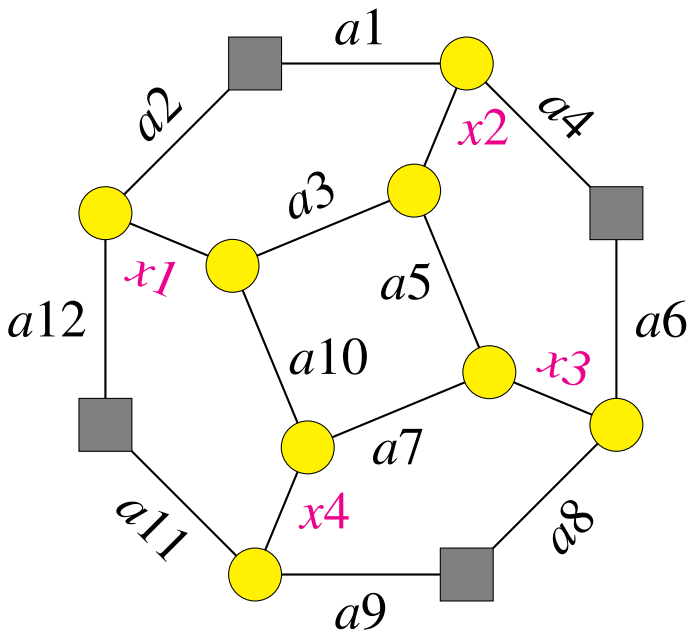
$4 * 8 = 32$ Clauses

Polynomial Tseitin Proofs

- $XOR(a1,a2,a3,a4) \rightarrow \exists x.(XOR(a1,a2,x) \wedge XOR(\bar{x},a3,a4))$

Split $x1,x2,x3$ and $x4$





$XOR(a1,a4,x2)$
$XOR(a3,a5,\overline{x2})$

$XOR(a2,a12,x1)$
$XOR(a3,a10,\overline{x1})$

$XOR(a6,a8,x3)$
$XOR(a5,a7,\overline{x3})$

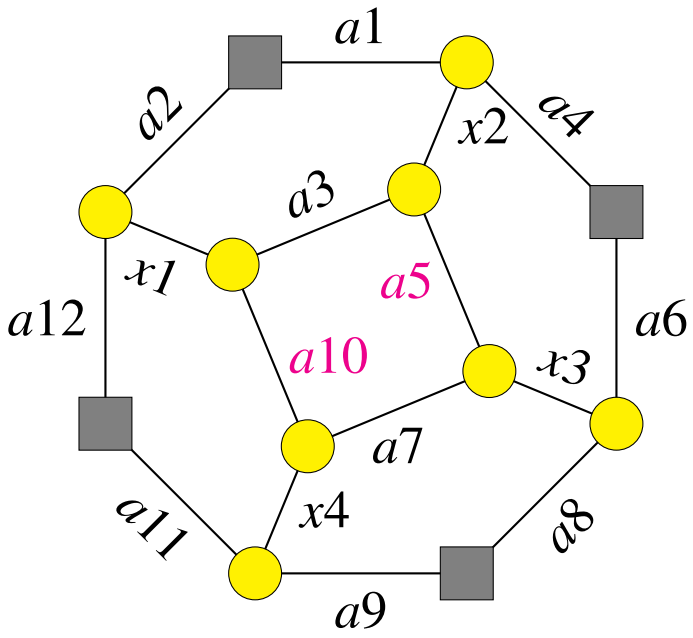
$XOR(a9,a11,x4)$
$XOR(a7,a10,\overline{x4})$

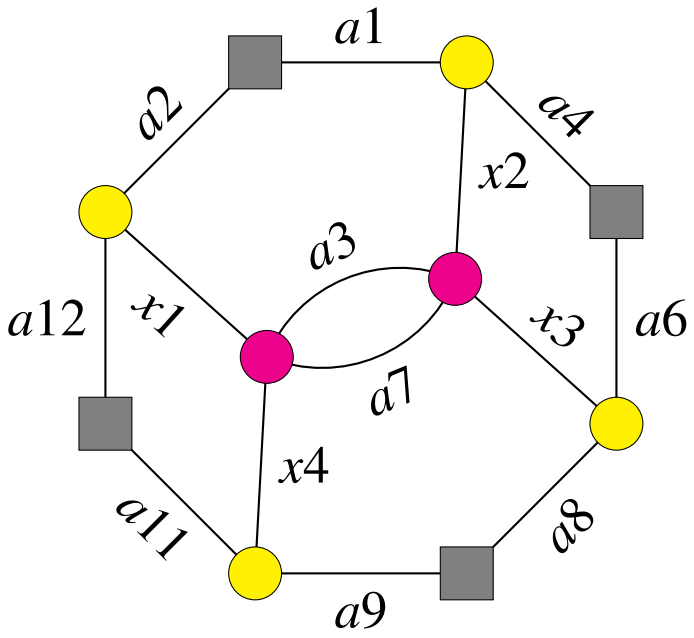
$8*4 = 32$ Clauses

Polynomial Tseitin Proofs

- $XOR(a1,a2,a3,a4) \leftarrow \exists x.(XOR(a1,a2,x) \wedge XOR(\bar{x},a3,a4))$

Merge $a10$ and $a5$





$XOR(a1,a4,x2)$

$XOR(a2,a12,x1)$

$XOR(a6,a8,x3)$

$XOR(a9,a11,x4)$

$XOR(x1,x4,a3,a7)$

$XOR(x2,x3,a3,a7)$

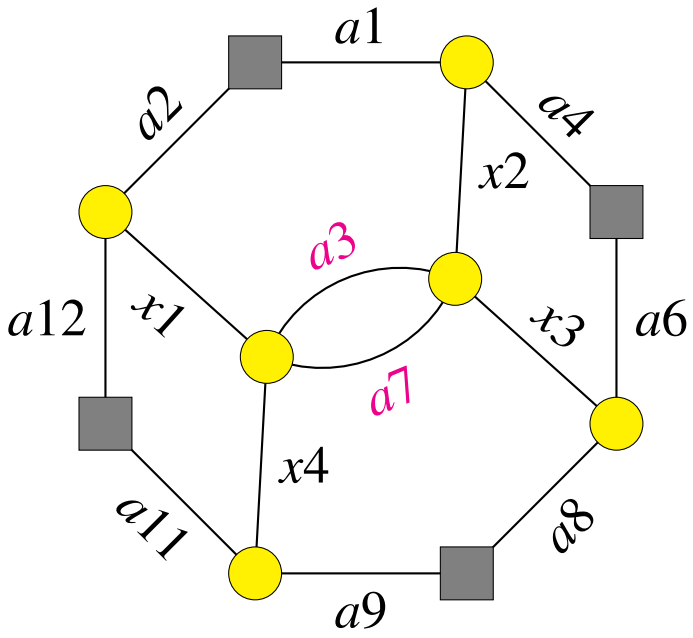
$4*4+2*8=32$

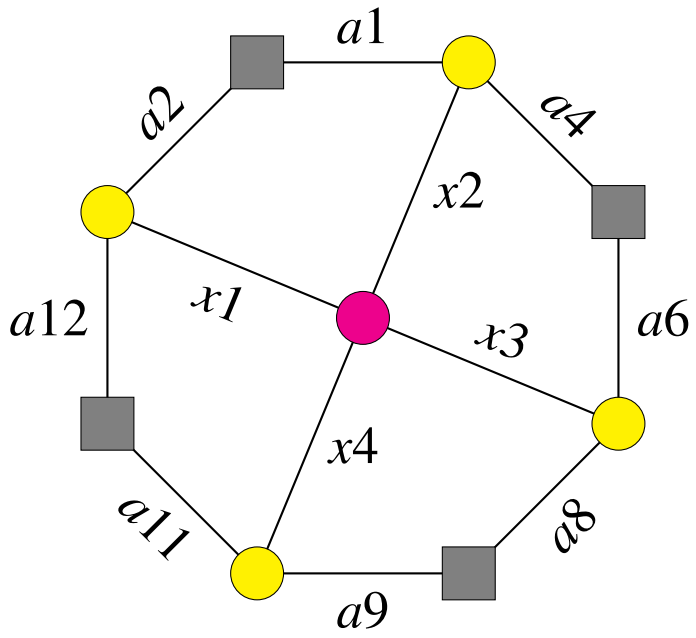
Clauses

Polynomial Tseitin Proofs

- $\exists x, y.(XOR(a1, a2, x, y) \wedge XOR(x, y, a3, a4)) \leftrightarrow XOR(a1, a2, a3, a4)$

Collapse





$XOR(a1, a4, x2)$

$XOR(a2, a12, x1)$

$XOR(a6, a8, x3)$

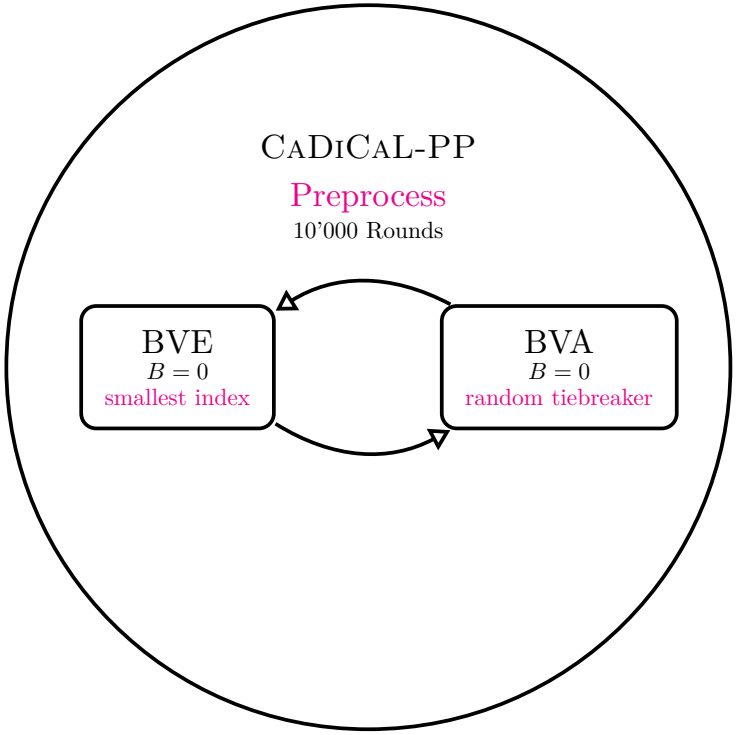
$XOR(a9, a11, x4)$

$XOR(x1, x2, x3, x4)$

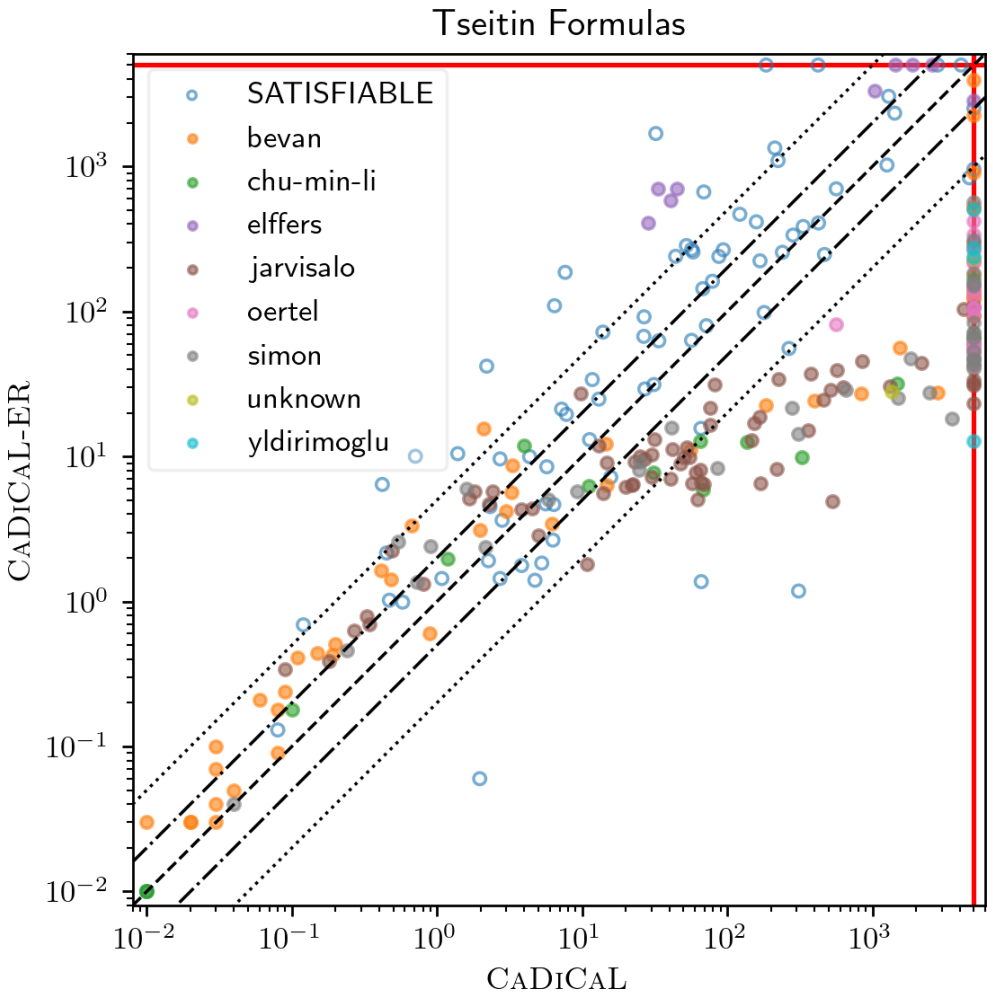
$4 * 4 + 8 = 24$ Clauses

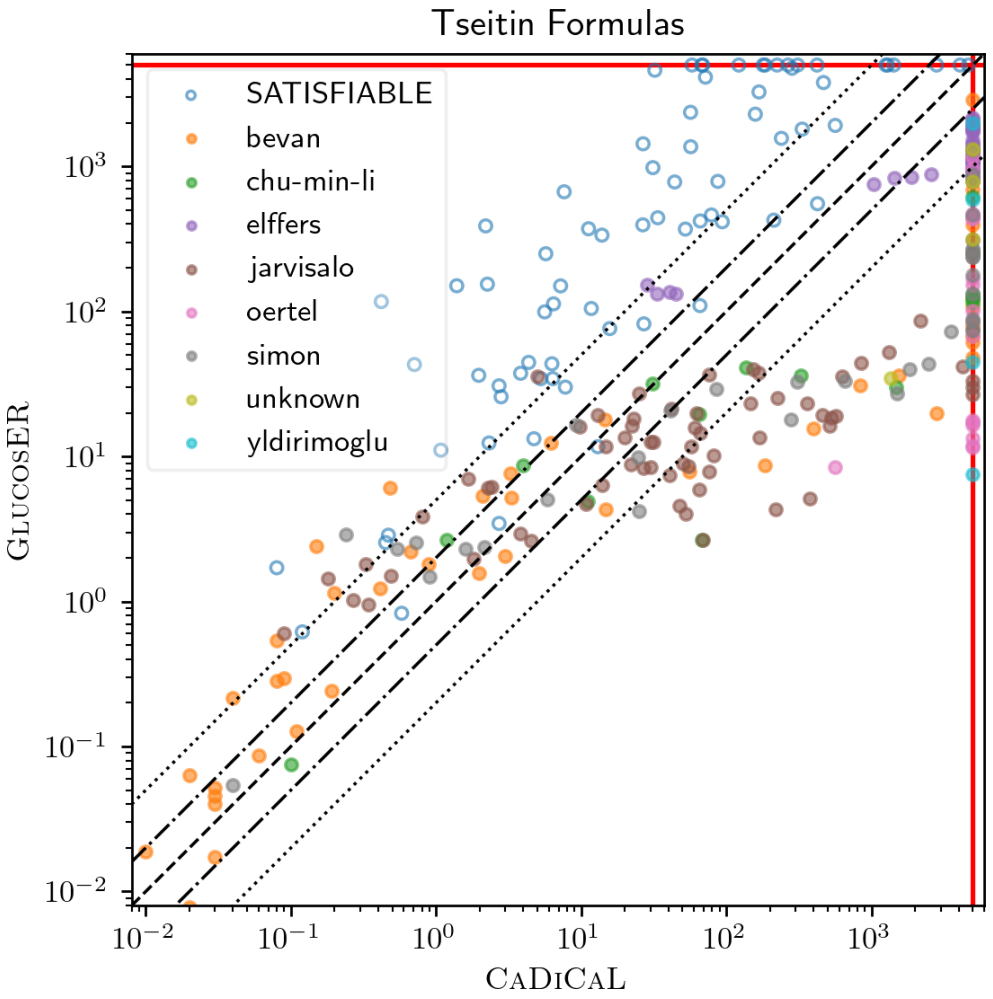
Polynomial Tseitin Proofs — CaDiCaL-PP

- CaDiCaL-PP
  - $B_{BVA} = B_{BVE} = 0$
  - no CDCL
- CaDiCaL-FX:
  - $B_{BVA} = -B_{BVE} - 1$
  - deterministic tiebreaker

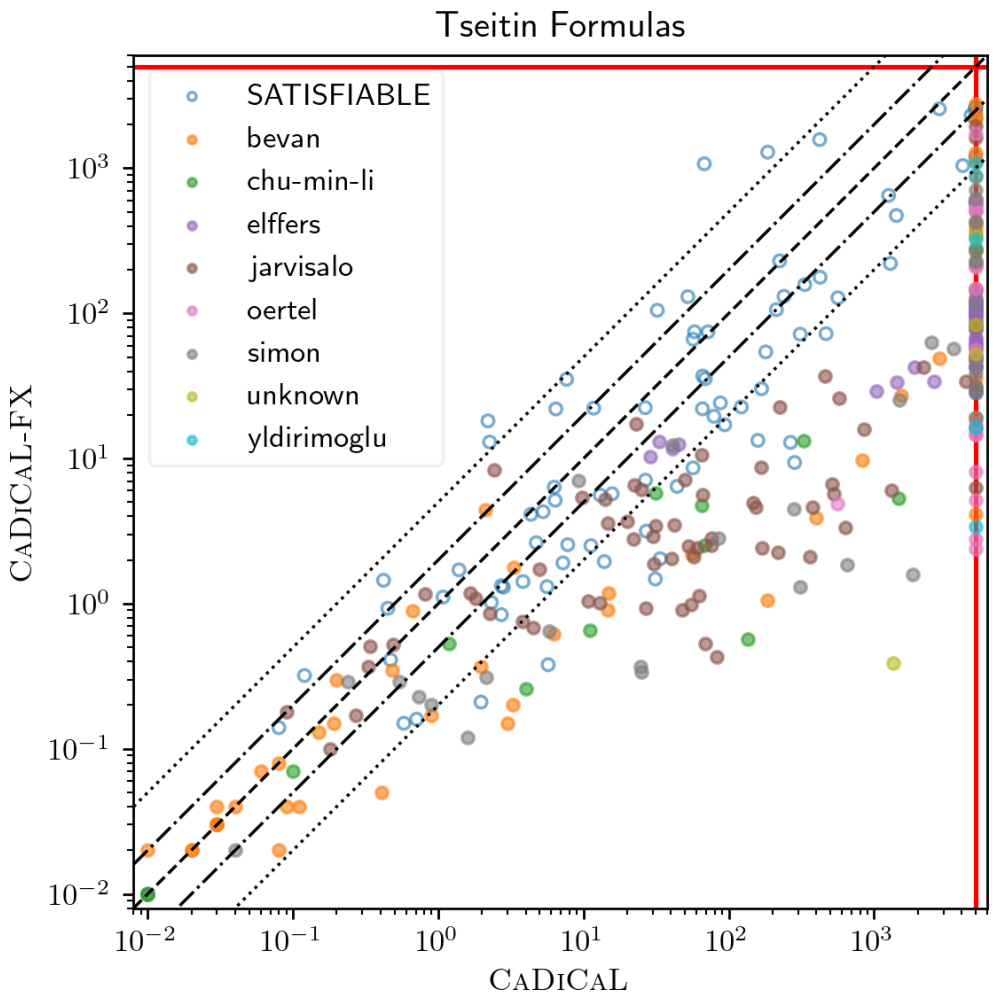


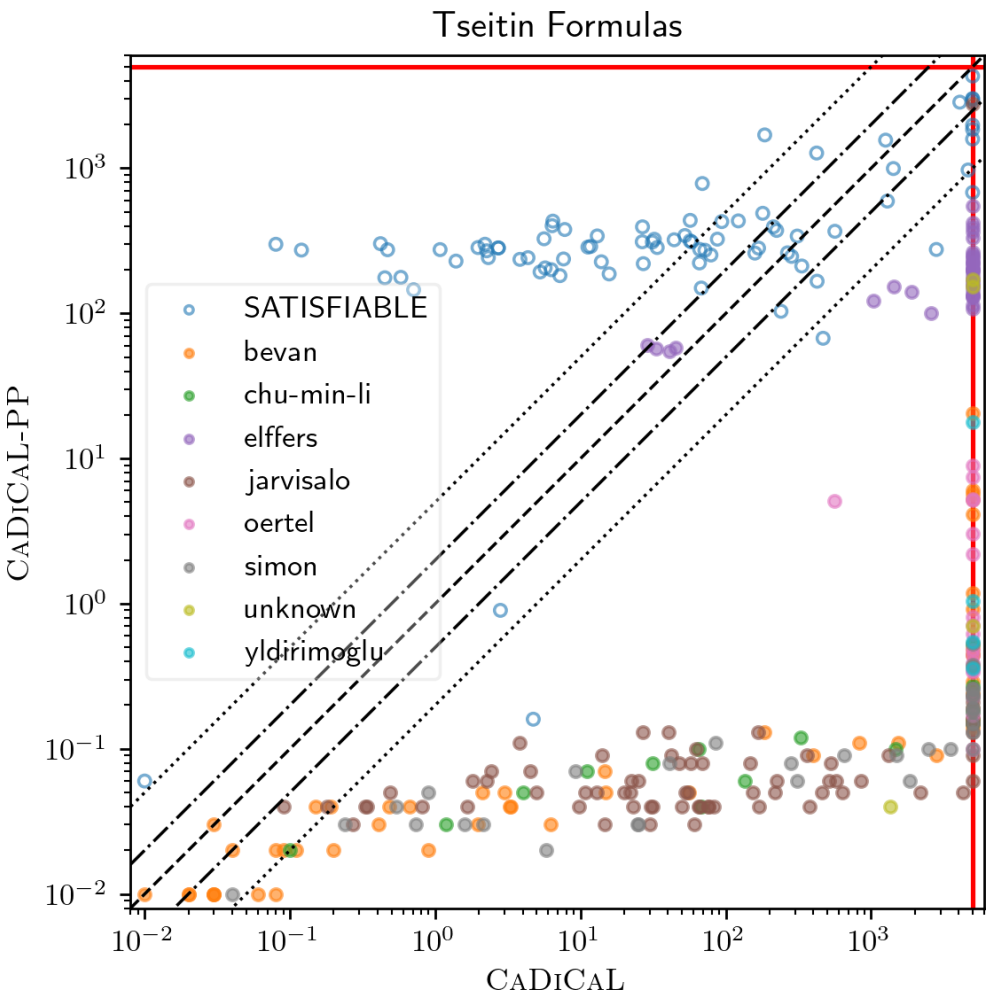
Results — Tseitin Families GBD



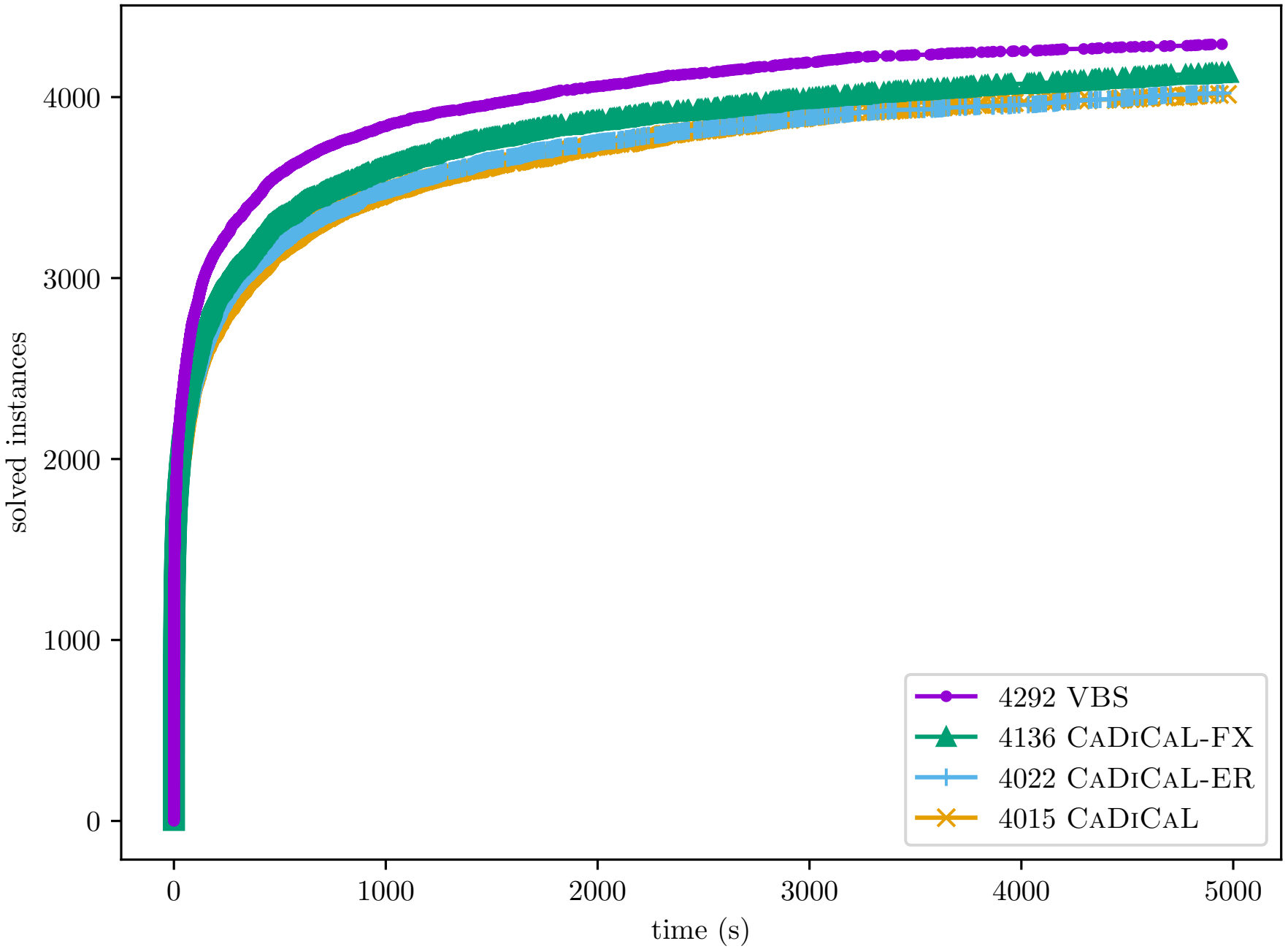


Results — Tseitin Families GBD

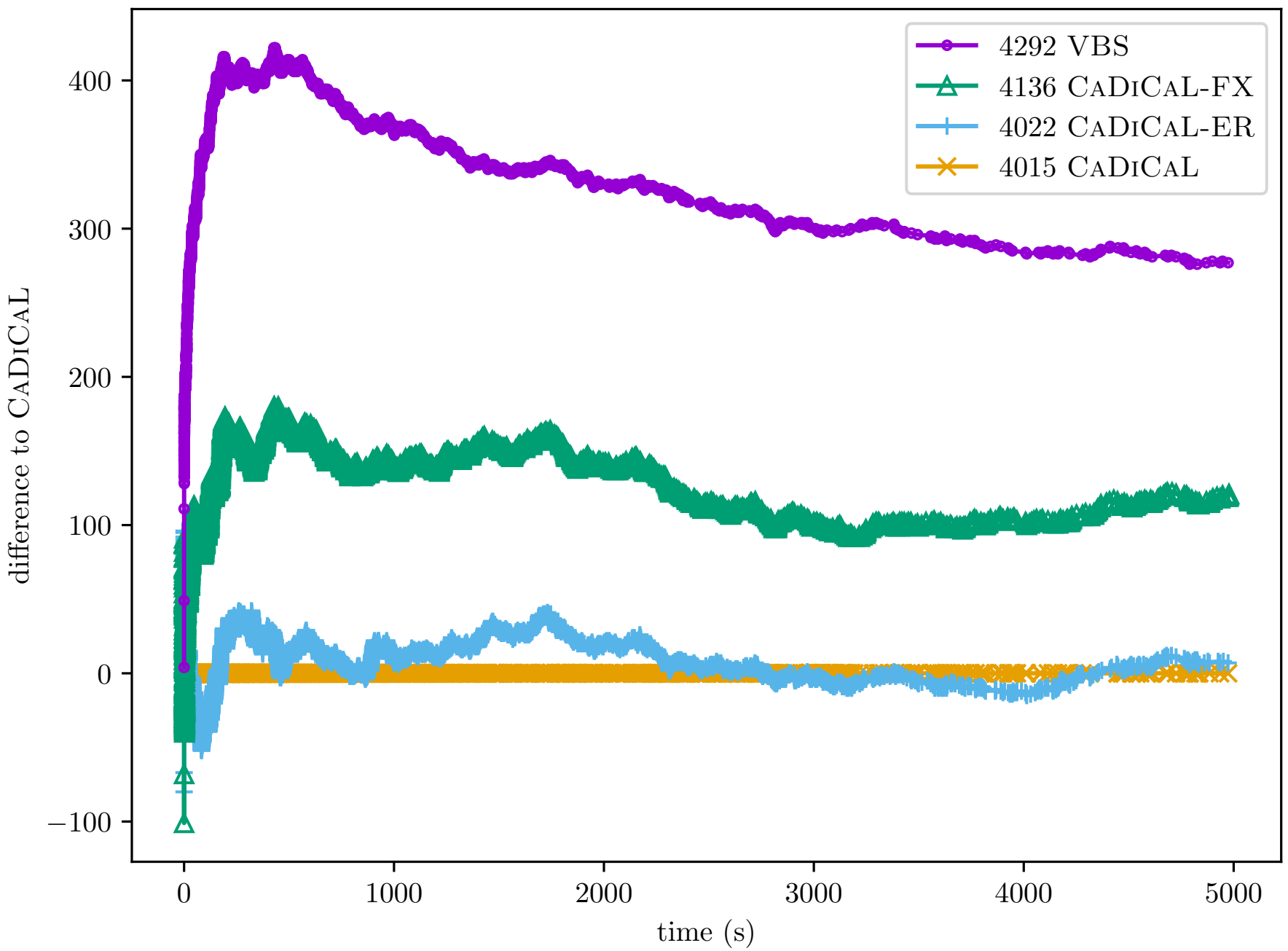




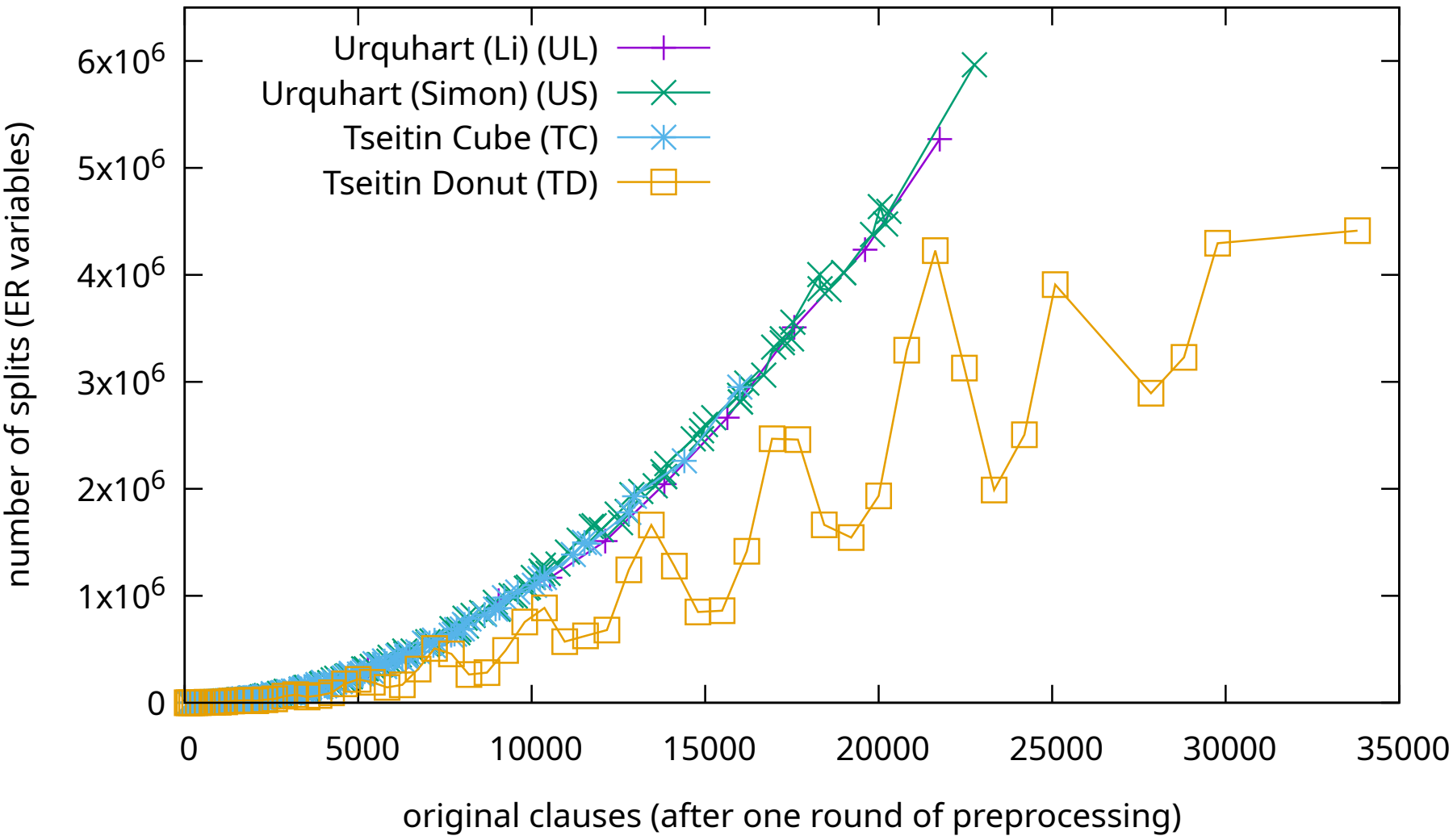
Results — SAT Competition 2022 Anniversary Track (5356)



Results — SAT Competition 2022 Anniversary Track (5356)



Results — Polynomial Proofs



Conclusion

- CaDiCaL-ER:
  - efficient reimplementation of GlucosER
- CaDiCaL-FX:
  - effective application of ER
  - gate factorization in CaDiCaL (got disqualified)
- CaDiCaL-PP:
  - polynomial proofs for Tseitin formulas
- future work:
  - explain difference in CaDiCaL-ER vs GlucosER
  - other gates (ITE, Half-Adders, ...)
  - interaction between variable addition and elimination

Bibliography

[CDPC'83] *G. S. Tseitin*. "On the Complexity of Derivation in Propositional Calculus"

[DAM'99] *Oliver Kullmann*. "On a generalization of extended resolution"

[AAAI'10] *Gilles Audemard, George Katsirelos, and Laurent Simon*. "A restriction of extended resolution for clause learning SAT solvers"

[HVC'12] *Norbert Manthey, Marijn Heule and Armin Biere*. "Automated Reencoding of Boolean Formulas"

[POS'25] *Katalin Fazekas, Florian Pollitt, Mathias Fleury, and Armin Biere*. "Incremental inprocessing rules beyond resolution"

[CAV'25] *Andrew Luka and Yakir Vizel*. "Property directed reachability with extended resolution"

[SAT'26] *Florian Pollitt, Mathias Fleury, Katalin Fazekas, Nils Froleyks, André Schidler, Dominik Schreiber and Armin Biere*. "CaDiCaL 3.0"