Learned Clause Minimization in Parallel SAT Solvers

Pragmatics of SAT 2019

Marc Hartung, Florian Schintke

ZUSE INSTITUTE BERLIN

SPONSORED BY THE

Federal Ministry of Education and Research

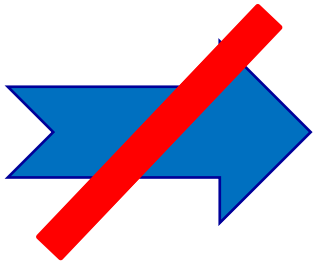
1. Background
2. Parallel Clause Minimization
3. Experiments
4. Conclusion

Background

(Learned) Clause Minimization in SC18

Solver	Author	CM/ LMC
MapleLCMDistChronoBT	Ryvchin et al.	✓
Maple_LCM_Scavel_fix2	Xu et al.	✓
Maple_CM	Luo et al.	✓
cms55-main-all4fixed	M. Soos	✓
Maple_CM_ordUIP	Luo et al.	✓
Maple_CM_Dist	Luo et al.	✓
cms55-main-all4fixed	M. Soos	✓
Maple_CM_ordUIP+	Luo et al.	✓
Maple_LCM_Scavel_200_fix2	Xu et al.	✓
cms55-main-all4fixed	M. Soos	✓

Top10 Main Track



Success was not transferred to parallel

Solver	Author	CM/ LMC
painless	Le Frioux et al.	✗
plingeling	A. Biere	✗
abcdsat	J. Chen	✓
cms55-parallel, 12 core	M. Soos	✓
cbpenelope	T. Sonobe	✗
ccspenelope	T. Sonobe	✗
syrup, 24 threads	Audemard et al.	✓
penelope_MDLC	Konan Tchinda et al.	✗
treengeling	A. Biere	✗
scalope	Konan Tchinda et al.	✗

Top10 Parallel Track

(Learned) Clause Minimization (LCM)

- Clause Minimization using Distillation[1] / Vivification[2]
- Applied at decision level zero

Clause $C = l_1 \lor l_2 \lor \ldots \lor l_i \lor \ldots \lor l_j \lor \ldots$

Iteratively propagate negations $\rightarrow$

<u>After</u> <u>propagating</u> $\neg l_1, \neg l_2, \ldots, \neg l_i$:

Case 1:
- l_j propagated to true
- C replaced by $l_1 \lor l_2 \lor \ldots \lor l_i \lor l_j$

Case 2:
- l_i propagated to false
- l_i removed from C

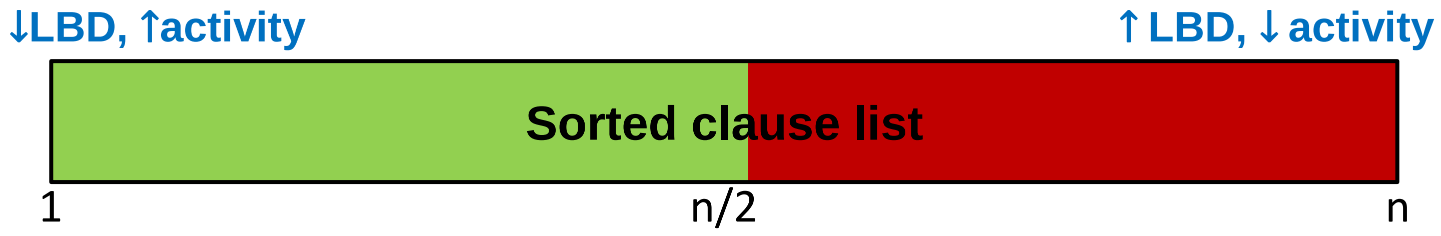
Case 3:
- Conflict detected
- C replaced by $l_1 \lor l_2 \lor \ldots \lor l_i$

- In this presentation: Minimization $\equiv$ Distillation/Vivification

LMC Approach [3]

Only apply CM to (in future) kept learned clauses

- Each clause minimized only once
- Reduction heuristic specifies which are kept
- Reduction example Glucose:



- low LBD, higher activity
→ keep and minimize

- high LBD, lower activity
→ remove

- Minimization triggered after a restart or decision tree is stashed

Parallel Clause Minimization

Heterogeneous vs. Homogenous

Heterogeneous minimization approach

Dedicate individual threads to minimization

- Examples:
    - CDCL solvers + One minimization thread [4][5]
    - Only part of solvers use minimization [6]
- Problems:
    - Not trivial for many cores
    - Introduces load balancing problem
    - Adds more magic parameters
- Finding good parameters expensive

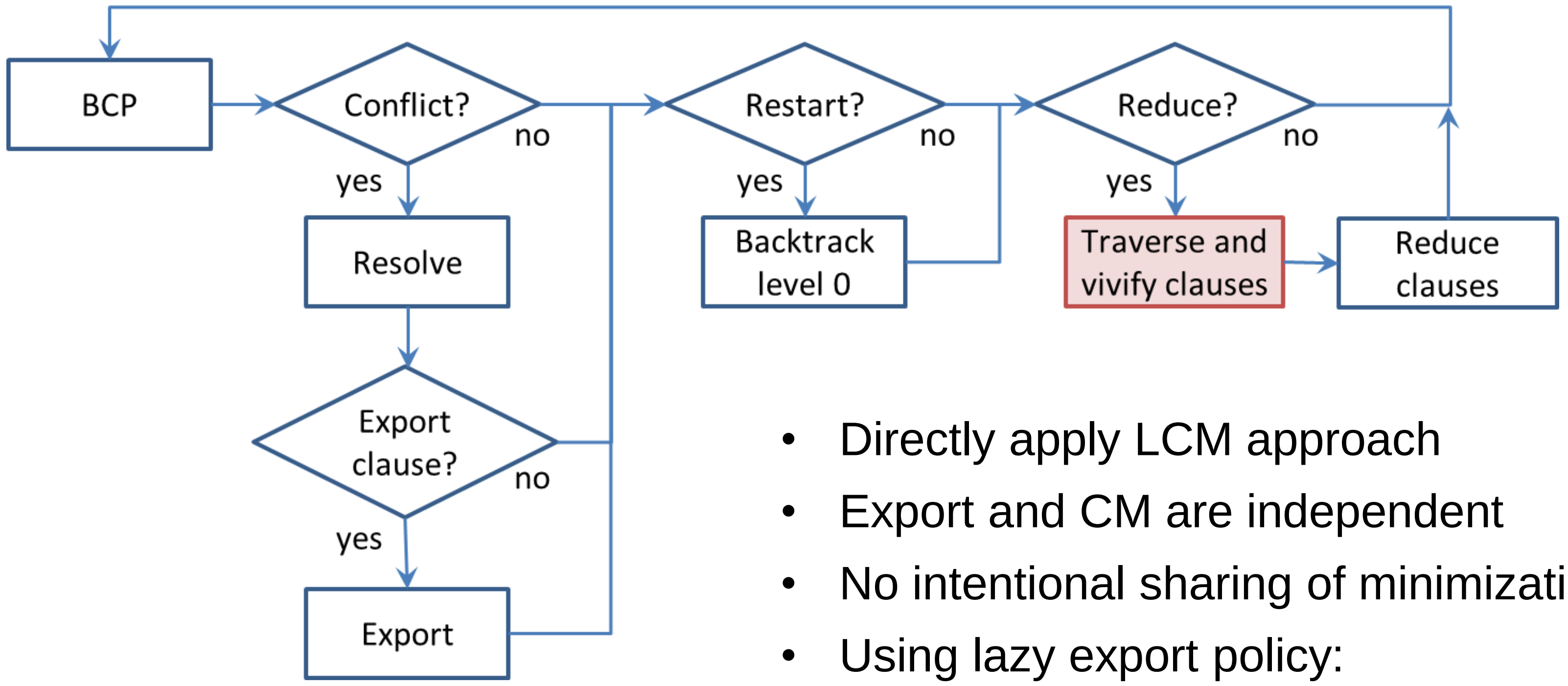
→ Discarded for future work

Heterogeneous vs. Homogenous

Homogenous minimization approach

- All solvers use same minimization approach
- Example: Minimize export clauses [7]
- Problems:
    - Balance minimization and BCP
    - How and if minimizations should be shared

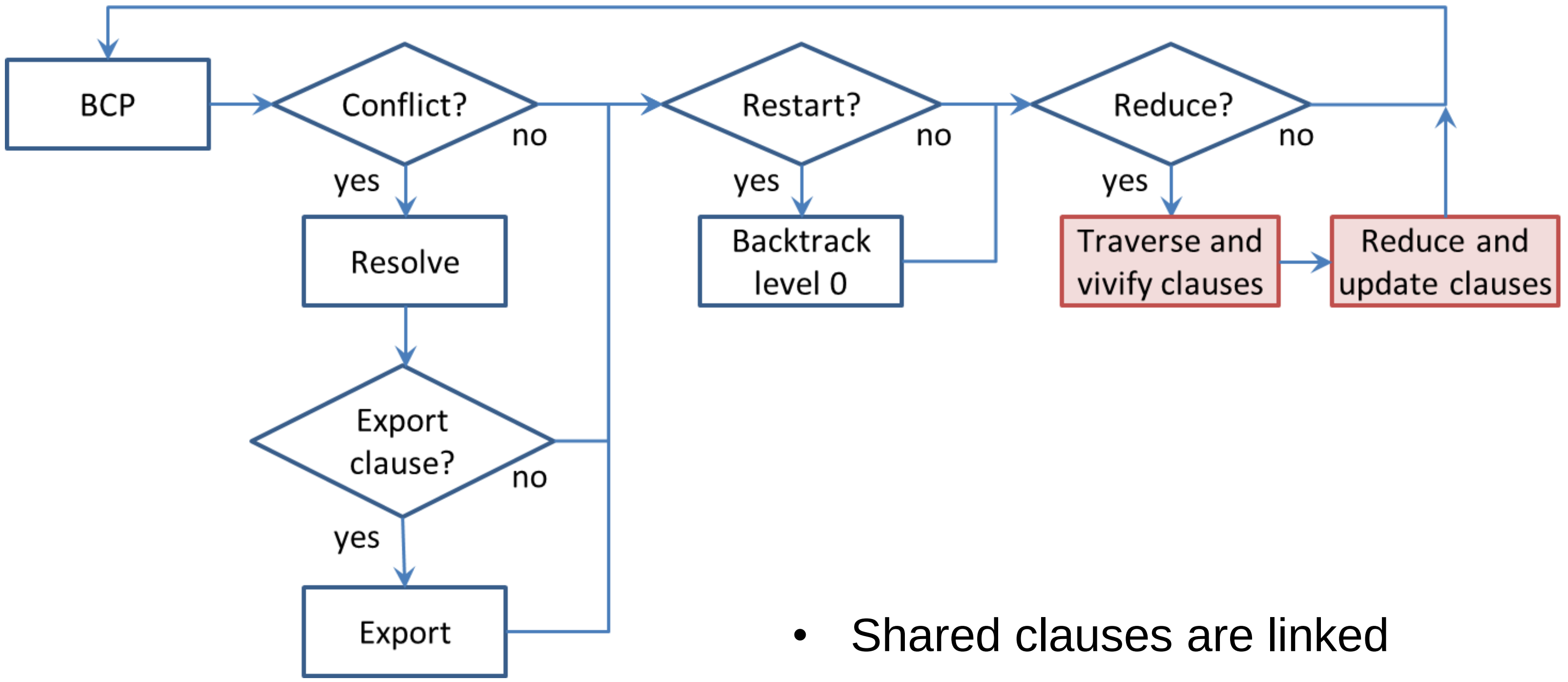
PCM – Private Clause Minimization



- Directly apply LCM approach
- Export and CM are independent
- No intentional sharing of minimizations
- Using lazy export policy:
  Minimized clauses might be shared

Implementation:
- LBD (≤ 5) cut
- Original version (no LBD cut) decreased performance
- Lazy export policy (two times used)

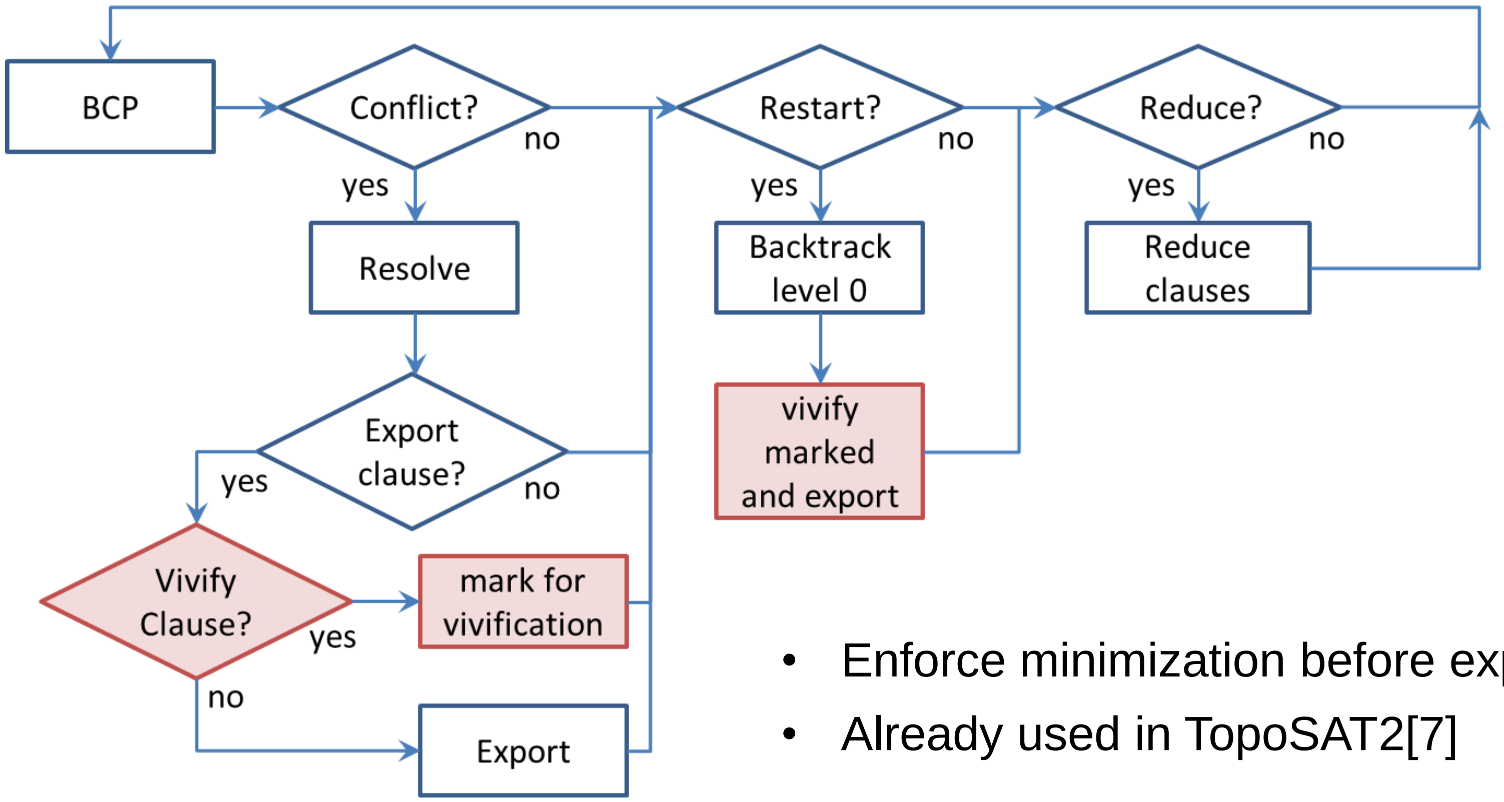
LPCM – Linked Private Clause Minimization



- Shared clauses are linked
- Minimizations shared via link

Implementation:
- LBD (≤ 5) cut
- Clause header contains pointer to memory chunk
- If minimized, chunk contains new clause

ECM – Export Clause Minimization



- Enforce minimization before export
- Already used in TopoSAT2[7]

Implementation:
- Lazy export policy (two times used)
- LBD (≤ 3 or ≤ 4) and length (≤ 30) cut
- Marked clauses are protected during reduction

Experiments

Test Set and Environment

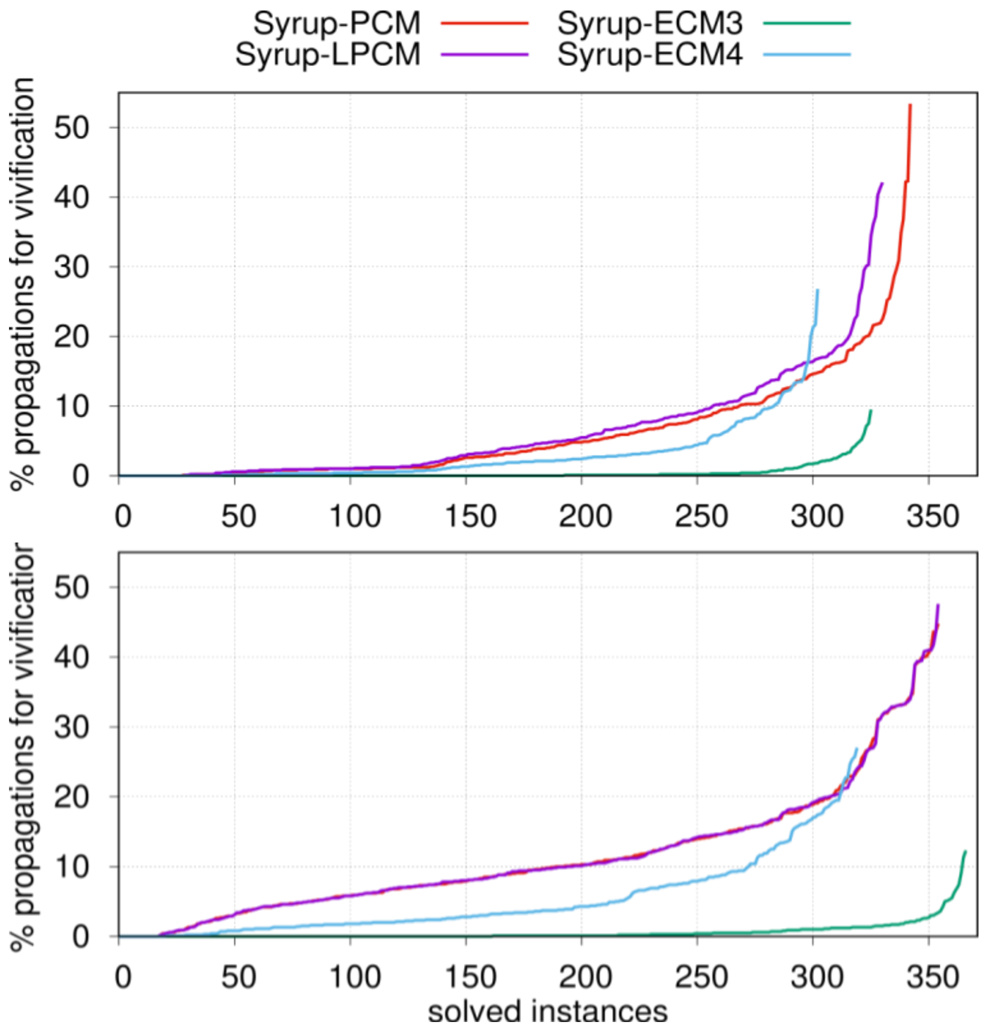
- SAT competition ‘16 application track, ’17 and ‘ 18 main track
- On Intel Xeon Phi 7250, 68 cores at 1.4 GHz with 96 GB RAM

- Maximum walltime of 15000 seconds
- Maximal 34 threads per solver
- Restrictions due to CPU frequency, cache and main memory

Vivification Overhead

SAT

UNSAT



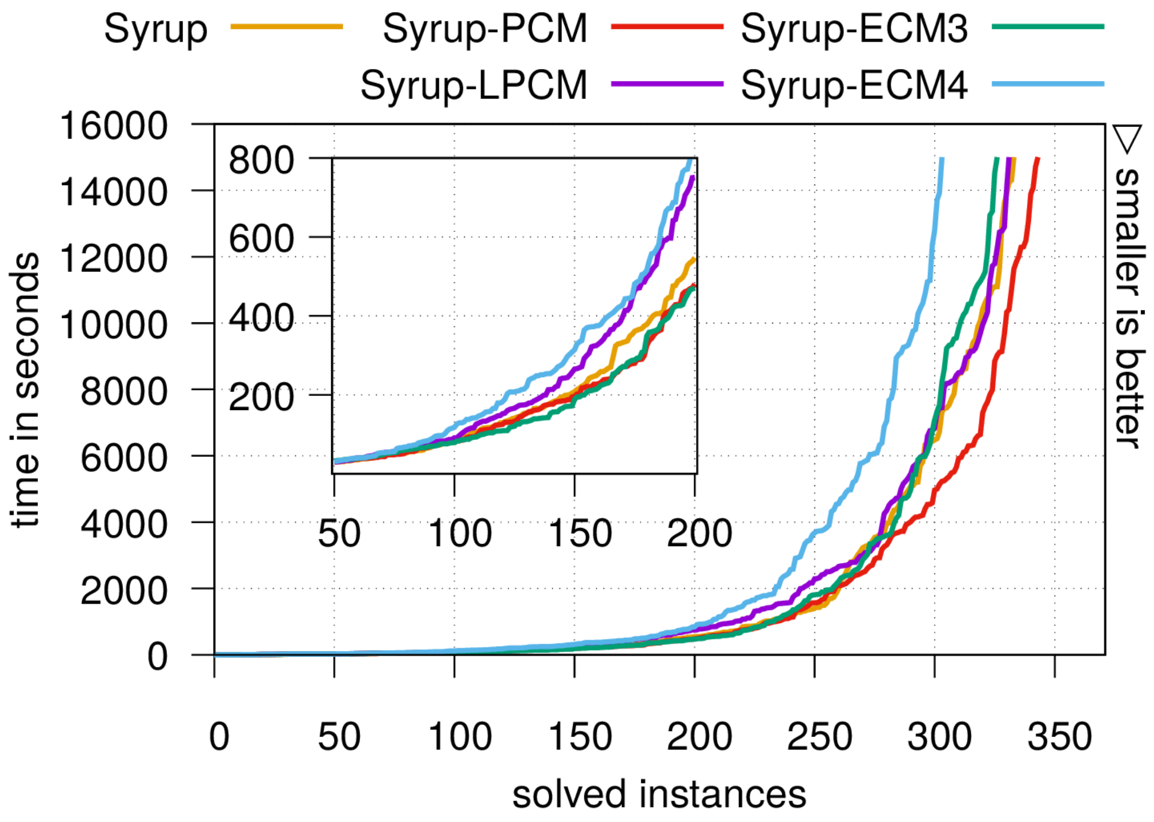
Propagation Overhead:

(L)PCM ≈ 10%
ECM3 ≈ 1%
ECM4 ≈ 4%
on average

Minimization success correlates with overhead:

(L)PCM ≈ 40%
ECM3 ≈ 6%
ECM4 ≈ 32%
on average

Syrup Runtime SAT



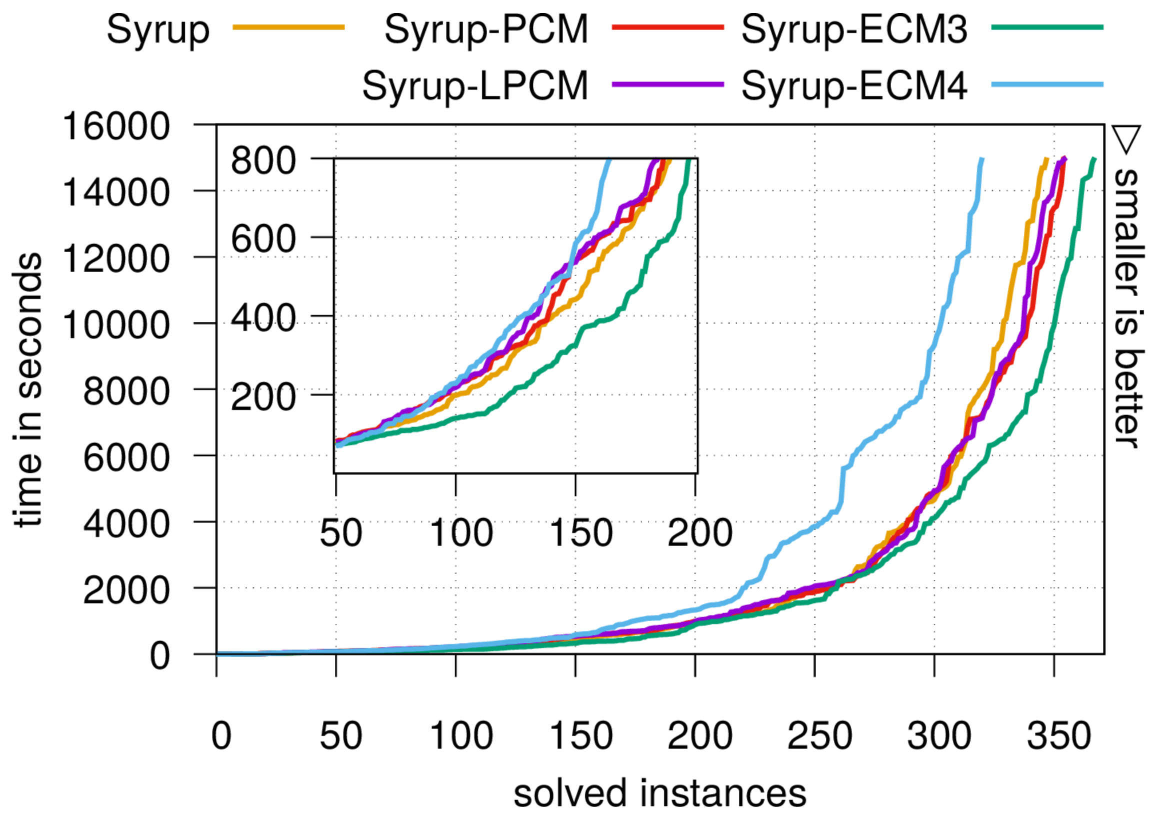
PCM increases SAT performance

Improvement to Syrup small for easy instances

Solved instances:

Syrup:	333
PCM:	**343**
LPCM:	331
ECM3:	326
ECM4:	302

Syrup Runtime UNSAT



Solved instances:

Syrup:	347
PCM:	355
LPCM:	355
ECM3:	**367**
ECM4:	321

- ECM increases overall UNSAT performance
- PCM, LPCM and ECM3 improve performance

Parallel CM Solver

TopoSAT2 – ECM

- Glucose 3.0 based ECM solver
- Direct clause export
- Copies of clauses are minimized and exported

→ Minimizations are not used by minimizing solver

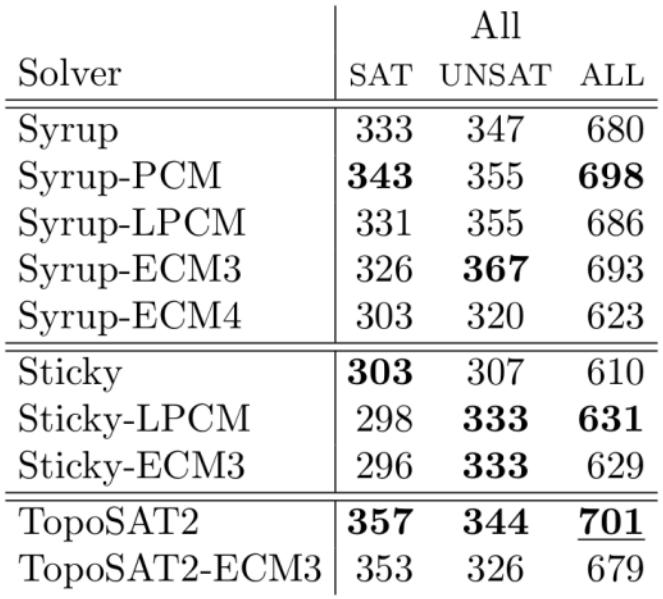
Sticky – LPCM, ECM

- Glucose 4.0 based solver with physical clause sharing
- No copy-sharing of clauses, only references are shared
- Adapted lazy clause sharing heuristic

SAT Competition Results

Results SC'16 (application track), SC'17, SC'18 (main track)

- Overall increase through nearly every CM approach
- Syrup-PCM nearly closed gap to Toposat2
- LPCM and ECM3 decrease SAT but increase solved UNSAT instances more



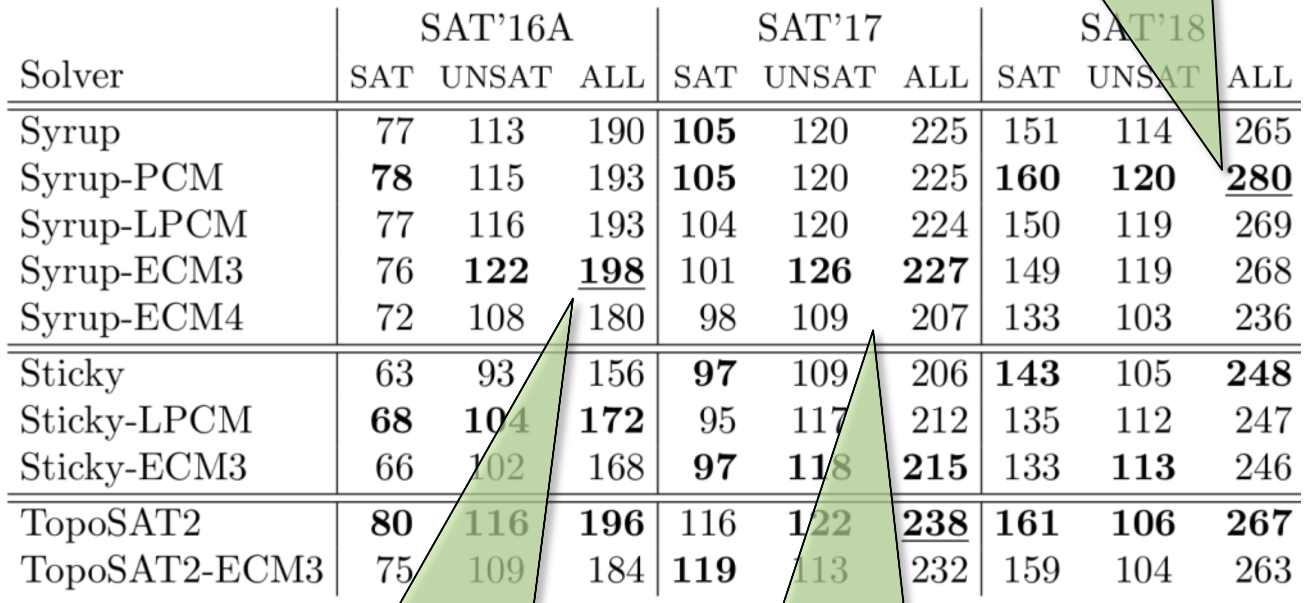
	All		
Solver	SAT	UNSAT	ALL
Syrup	333	347	680
Syrup-PCM	**343**	355	**698**
Syrup-LPCM	331	355	686
Syrup-ECM3	326	**367**	693
Syrup-ECM4	303	320	623
Sticky	**303**	307	610
Sticky-LPCM	298	**333**	**631**
Sticky-ECM3	296	**333**	629
TopoSAT2	**357**	**344**	**701**
TopoSAT2-ECM3	353	326	679

- TopoSAT2-ECM3 decrease:
  - No lazy export → missing activity filter for export → higher overhead
  - Minimizations not inserted in minimizing solver

SAT Competition Results

Single Competition Results

Syrup-PCM wins on SC’18 application track benchmarks



Solver	SAT'16A SAT	SAT'16A UNSAT	SAT'16A ALL	SAT'17 SAT	SAT'17 UNSAT	SAT'17 ALL	SAT'18 SAT	SAT'18 UNSAT	SAT'18 ALL
Syrup	77	113	190	**105**	120	225	151	114	265
Syrup-PCM	**78**	115	193	**105**	120	225	**160**	**120**	**280**
Syrup-LPCM	77	116	193	104	120	224	150	119	269
Syrup-ECM3	76	**122**	**198**	101	**126**	**227**	149	119	268
Syrup-ECM4	72	108	180	98	109	207	133	103	236
Sticky	63	93	156	**97**	109	206	**143**	105	**248**
Sticky-LPCM	**68**	**104**	**172**	95	117	212	135	112	247
Sticky-ECM3	66	102	168	**97**	**118**	**215**	133	**113**	246
TopoSAT2	**80**	**116**	**196**	116	**122**	**238**	**161**	**106**	**267**
TopoSAT2-ECM3	75	109	184	**119**	113	232	159	104	263

Syrup-ECM3 wins on SC’16 application track benchmarks

No real improvement on SC’17 benchmarks

Conclusion

- Homogeneous CM applicable for parallel solvers

  → Approaches solved 6 – 21 additional instances

- Sharing minimizations via link has no advantage

  → LPCM fewer solved instances than PCM

- More restrictive clause selection than in serial

  → ECM4 and TopoSAT2-ECM slow down
  → PCM/LPCM only succeed with LBD cut

- Prioritize:
  - Activity-based selection for SAT (PCM)
  - LBD-based selection for UNSAT (ECM)

References

[1] Hyojung Han and Fabio Somenzi. Alembic: An ecient algorithm for CNF preprocessing, *in DAC'07*

[2] Cdric Piette, Youssef Hamadi, and Lakhdar Sais. Vivifying propositional clausal formulae, *in ECAI'08*

[3] Mao Luo, Chu-Min Li, Fan Xiao, Felip Many, and Zhipeng L. An effective learnt clause minimization approach for CDCL SAT solvers*, in IJCAI'17*

[4] Siert Wieringa and Keijo Heljanko. Concurrent clause strengthening, *in SAT'13*

[5] Michael Kaufmann, Stephan Kottler, Michael Kaufmann, and Stephan Kottler. SArTagnan – a parallel portfolio SAT solver with lockless physical clause sharing*, in POS'11*

[6] Gilles Audemard and Laurent Simon. Glucose and Syrup: Nine years in the SAT competitions, *in Proceedings of SAT Competition 2018*

[7] Thorsten Ehlers and Dirk Nowotka. Glucose hacks and TOPOSAT2 description, *in Proceedings of SAT Competition 2018*

Questions?

Marc Hartung
Parallel and Distributed Computing
Zuse Institute Berlin (ZIB)
hartung@zib.de

ZUSE INSTITUTE BERLIN

HPS

SPONSORED BY THE
Federal Ministry of Education and Research