Iterative Symmetry Handling in the SAT Competition 2026

Markus Anders
RPTU Kaiserslautern-Landau
Kaiserslautern, Germany
0009-0004-5992-8433

Cayden Codel
Carnegie Mellon University
Pittsburgh, United States
0000-0003-3588-4873

***Abstract*—We submit KISSAT together with a new version of the SATSUMA symmetry preprocessor. This new version of SATSUMA applies iterative symmetry breaking using lightweight unit and binary clauses while producing SR proofs. Our goal is to demonstrate that symmetry handling is a viable default preprocessing technique for modern proof-producing SAT solvers.**

I. INTRODUCTION

We submit KISSAT [1] together with the SATSUMA [2] symmetry-breaking tool, used here as a preprocessor. This version of SATSUMA uses our new symmetry-handling techniques and algorithms [3], [4], and, as such, is an extension of the original SATSUMA algorithm [2]. To justify the clauses it adds, SATSUMA produces proofs in the substitution redundancy (SR) proof system [5], which is a generalization of DRAT [6].

Intuitively, our symmetry-handling algorithm *iteratively* applies *lightweight* symmetry breaking and CNF preprocessing techniques. Our approach is heavily inspired by similar work in optimization [7], [8].

Our submission consists of two solvers, where each solver combines SATSUMA with a different version of KISSAT: one with the latest public version of KISSAT as of this writing (v4.0.4),[1] and the other with the KISSAT modification used by the winner of the SAT Competition 2025 [9].[2]

II. NEW TECHNIQUES IN SATSUMA

Given a CNF formula, SATSUMA's new high-level routine for symmetry handling works as follows:

1) Simplify the CNF formula,
2) Detect symmetries in the simplified formula. If no non-trivial symmetries remain or if computational limits are exceeded, terminate.
3) Break symmetries by introducing unit and binary clauses.
4) Repeat from Step 1.

Here are some notes about various aspects of the algorithm.

CNF Simplification. Simplification includes the propagation of unit and pure literals, the removal of duplicate literals and clauses, and the strengthening of so-called unique literal clauses [10] into at-most-one constraints. While SATSUMA can also merge equivalent literals (as described here [3]), we disable this option in the competition submission.

Detection. The algorithm models the CNF formula as a graph and then uses DEJAVU [11], [12] to detect symmetries. It also makes use of some of DEJAVU's data structures and its implementation of the Schreier-Sims algorithm [13].

Fix and Cut. Symmetry handling combines orbitopal fixing, negation fixing, clausal fixing, and Schreier-Sims table cuts. We provide brief descriptions below. More comprehensive descriptions can be found in our work [3], [4].

Orbitopal fixing is a technique that can be applied on particular matrix models, i.e., matrices of literals extracted from the formula. The technique applies when the matrix model has row symmetry, meaning that rows can be exchanged by symmetries of the formula, and when the columns contain additional cardinality constraints. This allows the algorithm to fix literals corresponding to the lower-triangular part of the matrix using symmetry arguments.

Negation fixing, clausal fixing, and Schreier-Sims table cuts form a family of symmetry-breaking techniques based on pointwise stabilizer chains. The preprocessor iteratively stabilizes literals and updates the remaining symmetry group using the Schreier-Sims algorithm. Depending on the orbit structure, the algorithm either fixes literals or adds binary clauses (Schreier-Sims table cuts) over literal orbits.

Iteration. A key feature of the algorithm is its *iterative* preprocessing. Since all of the introduced symmetry-breaking constraints are unit or binary clauses, the added constraints often trigger further propagation and simplification. These simplifications can expose additional symmetries, allowing for multiple rounds of symmetry breaking.

Ordering. The order of symmetry breaking is guided by structural connectivity in the formula. For clausal fixing and Schreier-Sims table cuts, literals are prioritized according to the number of clauses shared with previously stabilized literals. For orbitopal fixing, columns of the involved matrix model are reordered using clique information extracted from a graph modeling the rows of the matrix. We use the maximum clique solver CLIQUER [14] to compute cliques. If the cliques are too large, we use a lightweight greedy approximation instead.

[1] https://github.com/arminbiere/kissat/releases/tag/rel-4.0.4
[2] https://satcompetition.github.io/2025/downloads/solvers/main/AE_kissat2025_MAB.tar.xz

III. LOC Changes and AI Use

Compared to last year's version of SATSUMA, roughly 5 000 lines of code have changed, with another 250 lines changed in DEJAVU. Note that while the addition of CLIQUER is also new, we don't count CLIQUER's code base in our LoC count, instead treating it like a black box solver.

We only made very minor changes to all other included tools. In KISSAT (v4.0.4) and AE-KISSAT2025-MAB, we made it so that the solvers *append* to an existing proof file, i.e., the one produced by SATSUMA, instead of *overwriting* it. We also added computational limits to CLIQUER. These changes amounted to fewer than 10 lines changed in each tool.

No AI was used to make any of these modifications, and no heuristics were tuned by AI by the present authors. However, the base solver AE-KISSAT2025-MAB we use from last year's competition was, according to its authors, modified and tuned by AI [9].

IV. Availability

SATSUMA is open-source and freely available.[3]

V. Acknowledgement

We thank Sofia Brenner, Marijn Heule, and Pascal Schweitzer for their help in creating the algorithms that make this submission possible.

References

[1] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, and F. Pollitt, "CaDiCaL, Gimsatul, IsaSAT and Kissat entering the SAT Competition 2024," in *Proc. of SAT Competition 2024 – Solver, Benchmark and Proof Checker Descriptions*, ser. Department of Computer Science Report Series B, M. Heule, M. Iser, M. Järvisalo, and M. Suda, Eds., vol. B-2024-1. University of Helsinki, 2024, pp. 8–10.

[2] M. Anders, S. Brenner, and G. Rattan, "Satsuma: Structure-based symmetry breaking in SAT," *J. Artif. Intell. Res.*, vol. 85, 2026. [Online]. Available: https://doi.org/10.1613/jair.1.18744

[3] M. Anders, C. Codel, and M. J. H. Heule, "Simplify, Order, Break, Repeat," in *29th International Conference on Theory and Applications of Satisfiability Testing, SAT 2026*, 2026, to appear.

[4] ——, "Orbitopal fixing in SAT," in *Tools and Algorithms for the Construction and Analysis of Systems - 32nd International Conference, TACAS 2026, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2026, Turin, Italy, April 11-16, 2026, Proceedings, Part I*, ser. Lecture Notes in Computer Science, S. Junges and G. Katz, Eds. Springer, 2026, pp. 89–109.

[5] C. R. Codel, J. Avigad, and M. J. H. Heule, "Verified substitution redundancy checking," in *Formal Methods in Computer-Aided Design, FMCAD 2024, Prague, Czech Republic, October 15-18, 2024*. IEEE, 2024, pp. 186–196.

[6] N. Wetzler, M. J. H. Heule, and W. A. Hunt, "DRAT-trim: Efficient checking and trimming using expressive clausal proofs," in *Theory and Applications of Satisfiability Testing - SAT 2014*, C. Sinz and U. Egly, Eds. Springer International Publishing, 2014, vol. 8561, pp. 422–429.

[7] V. Kaibel, M. Peinhardt, and M. E. Pfetsch, "Orbitopal fixing," *Discrete Optimization*, vol. 8, no. 4, pp. 595–610, 2011.

[8] A. Jäger and M. E. Pfetsch, "Structure-preserving symmetry presolving for mixed-binary linear problems," in *Integer Programming and Combinatorial Optimization*, 2026, to appear.

[9] H. Ding, M. Luo, C.-M. Li, S. Li, R. Chen, C. Xiong, and X. Wu, "A self-optimizing framework for sat solvers via population evolution and large language model collaboration," in *Proceedings of SAT Competition 2025: Solver and Benchmark Descriptions*, ser. Proceedings of SAT Competitions, C. Codel, K. Fazekas, M. J. H. Heule, and M. Iser, Eds., Vienna, 2025, pp. 15–17. [Online]. Available: https://doi.org/10.34726/10379

[10] A. Sheng, J. E. Reeves, and M. J. H. Heule, "Reencoding unique literal clauses," in *28th International Conference on Theory and Applications of Satisfiability Testing, SAT 2025, August 12-15, 2025, Glasgow, Scotland*, ser. LIPIcs, vol. 341. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, pp. 29:1–29:21.

[11] M. Anders and P. Schweitzer, "Parallel computation of combinatorial symmetries," in *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, ser. LIPIcs, vol. 204. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 6:1–6:18.

[12] ——, "Search problems in trees with symmetries: Near optimal traversal strategies for individualization-refinement algorithms," in *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, ser. LIPIcs, vol. 198. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 16:1–16:21.

[13] Á. Seress, *Permutation Group Algorithms*, ser. Cambridge Tracts in Mathematics. Cambridge University Press, 2003.

[14] P. R. Östergård, "A fast algorithm for the maximum clique problem," *Discrete Applied Mathematics*, vol. 120, no. 1, pp. 197–207, 2002, special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization.

[3] https://github.com/markusa4/satsuma