



Streamlining Distributed SAT Solver Design (Abstract)

Dominik Schreiber ✉ 🏠 

Karlsruhe Institute of Technology, Germany

Niccolò Rigi-Luperti ✉ 🏠 

Karlsruhe Institute of Technology, Germany

Armin Biere ✉ 🏠 

University of Freiburg, Germany

Abstract

We briefly describe our work on streamlining distributed SAT solver design, accepted at SAT 2025.

2012 ACM Subject Classification Hardware → Theorem proving and SAT solving; Theory of computation → Distributed algorithms

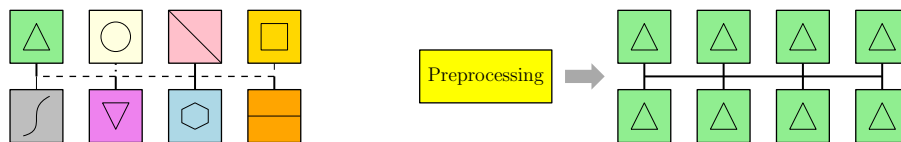
Keywords and phrases Satisfiability, parallel SAT solving, distributed computing, preprocessing

1 Overview

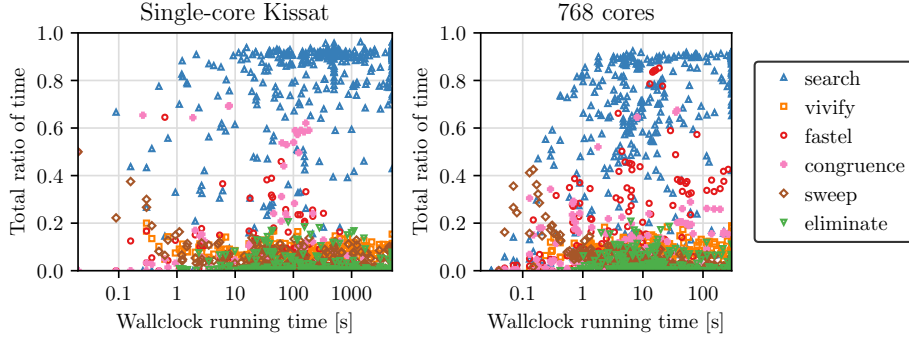
Over the last years, parallel and distributed clause-sharing SAT solvers have been established as powerful automated reasoning tools that can conquer previously infeasible instances [4, 8]. A common design of distributed SAT solvers is to run many off-the-shelf sequential solvers in parallel, employ some diversification (e.g., restart intervals or decision orders), and share conflict clauses among the solver threads [6]. This approach, naïvely, adopts all best practices of sequential solver design, which, however, may be less useful or even actively detrimental for distributed solving.

In particular, solvers employ multiple simplification techniques such as *preprocessing* and *inprocessing* [2]. In a distributed setting, these techniques are typically performed by all solver threads in a fully redundant fashion. These tasks are not parallelized and can therefore present a scalability bottleneck. Furthermore, some simplification techniques also heavily modify the original formula. When two solvers operate on different representations of the problem, efficient information exchange can be obstructed. Schreiber and Sanders observed this phenomenon for example with bounded variable elimination [3].

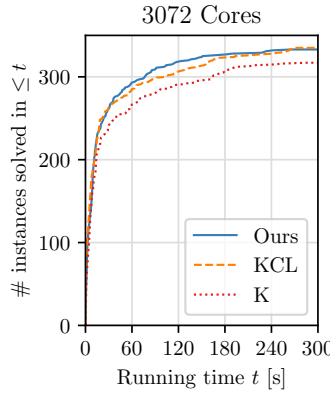
In this work we diagnose such shortcomings in the state-of-the-art system MallobSat and propose first effective mitigations. In particular, we replace the redundant pre- and inprocessing at all threads with single-core preprocessing that runs next to the parallel search, remove LBD values from the clause-sharing operation, and slim down solver diversification to very few lightweight and uniform methods. Experimental evaluations on up to 3072 cores (64 nodes) confirm that our measures improve performance while also drastically simplifying



■ **Figure 1** Left: Common distributed solver design, where a portfolio of solvers (colored squares) with diverse strategies (symbols in the squares) periodically exchange insights (connections between squares) with some “language barriers”. Right: Streamlined design, where parallel solver threads are *uniform* CDCL searchers and operate on the results of preprocessing that is logically *separate*.



■ **Figure 2** By-input distribution over the total solver time ratio spent in actual (CDCL) search and in selected pre-/inprocessing tasks, for single-threaded Kissat (left) and at 768 cores (right), relative to overall (wallclock) running time.



■ **Figure 3** Comparison of three tested MallobSat configurations at 3072 cores.

the SAT solving program to a pure “search-only” approach. This challenges the predominant view that solver threads need to either explicitly operate on different sub-spaces (*search space splitting*) or be explicitly *diversified* in terms of their approaches (*portfolio*) [8].

2 Highlighted Results

We profile the cutting-edge distributed SAT solver MallobSat [7, 8] in terms of the amount of time individual solver threads spend in different tasks. Figure 2 shows these ratios for Kissat, both in a single-threaded run as well as in a 768-core distributed setting. Pre- and inprocessing techniques take up relatively more time in the distributed setting, presenting an increased danger for redundant computations. We also investigate the impact of LBD values and find that in a distributed setting disabling them does not impact performance, marking another deviation from sequential SAT solving [1, 5]. Our benchmark set consists of 200 instances taken from both the ’23 and ’24 SAT Competitions. In the distributed setting a time-limit of 300s is set, to keep the computational demand manageable.

We then propose a new “search-only” setup for MallobSat: We use Kissat as the only solver backend, drop all pre- and inprocessing techniques and disable all diversification strategies, i.e. solvers no longer received diversified hyper-parameters. This results in a “pure” clause-sharing solver with uniform worker threads (sketched in Fig. 1 right) and next to no characteristics of a *portfolio* (Fig. 1 left). We then configure a single thread to perform a single run of a number of crucial simplification techniques, making use of all out-of-the-box preprocessing capabilities of Kissat.

Figure 3 shows the performance of this new setup compared to two previously used MallobSat setups. One makes use of Kissat, Cadical and Lingeling (denoted KCL, winner of the cloud track of the SAT 2024 competition), the other one uses only a Kissat backend (denoted K). Our new setup beats the previous Kissat-only setup and matches the KCL-setup in performance, while using a much smaller codebase with a highly simplified configuration. This streamlined approach now also simplifies the integration of any future extensions.

References

- 1 Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 399–404, 2009.
- 2 Armin Biere, Matti Järvisalo, and Benjamin Kiesl. Preprocessing in SAT solving. In *Handbook of Satisfiability*, pages 391–435. IOS Press, 2021. doi:10.3233/faia200987.
- 3 Jannick Borowitz, Dominik Schreiber, and Peter Sanders. An empirical study on learned clause overlaps in distributed SAT solving. In *Pragmatics of SAT (PoS)*, 2024. URL: <https://satres.kikit.kit.edu/papers/2024-pos-empirical.pdf>.
- 4 Byron Cook. Automated reasoning’s scientific frontiers. <https://www.amazon.science/blog/automated-reasonings-scientific-frontiers>, 2021. Amazon Science.
- 5 Yoichiro Iida, Tomohiro Sonobe, and Mary Inaba. Parallel Clause Sharing Strategy Based on Graph Structure of SAT Problem. In *Theory and Applications of Satisfiability Testing (SAT)*, pages 17:1–17:18, 2024. doi:10.4230/LIPIcs.SAT.2024.17.
- 6 João Marques-Silva, Inês Lynce, and Sharad Malik. CDCL SAT solving. In *Handbook of Satisfiability*, pages 131–153. IOS Press, 2021. doi:10.3233/faia200987.
- 7 Dominik Schreiber and Peter Sanders. Scalable SAT solving in the cloud. In *Theory and Applications of Satisfiability Testing (SAT)*, pages 518–534. Springer, 2021. doi:10.1007/978-3-030-80223-3_35.
- 8 Dominik Schreiber and Peter Sanders. MallobSat: Scalable SAT solving by clause sharing. *Journal of Artificial Intelligence Research*, 80:1437–1495, 2024. doi:10.1613/jair.1.15827.