A Comparative Analysis of UNSAT Core Extraction Methodologies for Unweighted MaxSAT

José João Alves dos Santos Ferreira

April 19, 2026

1 Introduction

Boolean Satisfiability (SAT) [1] is the fundamental problem of determining if there exists an assignment of truth values to variables that satisfies a given formula in Conjunctive Normal Form (CNF) [2]. Maximum Satisfiability (MaxSAT) [3] extends this by partitioning clauses into hard and soft sets, where the objective is to satisfy all hard clauses while minimizing the total cost of falsified soft clauses. In the unweighted variant of MaxSAT, each soft clause carries equal weight, effectively turning the optimization goal into minimizing the total number of violated soft clauses. SAT solvers have seen remarkable improvements over the past decades. MaxSAT solvers typically address optimization problems by making repeated calls to a SAT solver, each time with a modified formula. To manage the growth of these formulas effectively, modern MaxSAT solvers adopt core-guided paradigms. These approaches leverage the ability of SAT solvers to identify unsatisfiable subformulas (cores), using this information to iteratively refine the search and converge toward an optimal solution.

One of the most prominent core-guided strategies for unweighted MaxSAT problems is the Msu3 algorithm [4, 5]. Compared to its predecessors, Msu3 is generally more efficient for unweighted problems than Msu1, which adds relaxation variables to every soft clause from the start, or Msu2, which lacks the more efficient cardinality constraint management found in Msu3. Since Msu3 targets the minimum number of falsified clauses, its efficiency depends heavily on the precision of the core extraction process. So, a critical decision in implementing Msu3 is the choice of core extraction methodology. The most common approach utilizes internal variable assumptions, where the solver returns a core derived from the conflict analysis performed during the search. Alternatively, one can utilize Delete Resolution Asymmetric Tautology (DRAT) proofs, which is the standard format for certifying unsatisfiability.

This study aims to compare these two methodologies — assumptions-based and proof-based extraction — to evaluate their effectiveness within the Msu3 framework. The primary objective is to determine which method identifies a value closer to the global optimum during the initial disjoint core extraction phase, as this phase is significantly faster. Additionally, this research provides a comparative analysis of the size of the extracted UNSAT cores across successive iterations, which ideally should be small.

This report describes research conducted between November and December 2025 in the Information and Communication Technologies (ICT) field. The work was performed at the Computer Science Department (CSD) of Carnegie Mellon University (CMU) under the supervision of Professor Ruben Martins. This opportunity was made possible by the CMU Portugal Visiting Students mobility initiative, supported by the Fundação para a Ciência e a Tecnologia (FCT). At the time, I was a Master's student in Computer Science and Engineering at Instituto Superior Técnico, University of Lisbon.

2 Methodology

The Msu3 algorithm [4] operates in two distinct phases to solve the unweighted MaxSAT problem.

Phase 1: Disjoint Core Extraction. In the first phase, the solver establishes an initial lower bound λ_1 by identifying a maximal set of disjoint UNSAT cores. A SAT solver is called iteratively; once a core is identified, the soft clauses involved in that core are removed from the formula before the next solver call. In the SAT community, cores are considered disjoint if their sets of soft clauses have an empty intersection. While cores may share hard clauses, removing the soft clauses ensures that each identified core in this phase is unique regarding the optimization objective. After no more disjoint cores can be found, the algorithm re-adds all soft clauses, each associated with a unique relaxation variable.

Phase 2: Optimization. In the second phase, Msu3 applies cardinality constraints over the relaxation variables to assert that the number of falsified soft clauses equals the current lower bound. If the formula is unsatisfiable, a new core is identified, the lower bound is incremented, and the clauses are relaxed further with additional variables. This process repeats until the formula becomes satisfiable, yielding the minimum number of violated soft clauses λ_2. Unlike the first phase, cores found here are not disjoint, as

soft clauses are not removed between iterations. Despite this, the study tracks and analyzes core sizes across both phases.

The disjoint core extraction phase is actually optional, but it speeds up the process by tightening the initial lower bound and minimizing the relaxation variables and cardinality constraints needed for optimization. Implementation is made accessible through PySAT [6], a Python toolkit providing a streamlined interface for various state-of-the-art SAT solvers. It follows the MSU3 pseudocode provided by its authors [4], is open-source and is available in a public repository [1]. Two methodologies for extracting UNSAT cores were implemented and compared using PySAT:

Assumptions-based. This is an in-memory process utilizing the PySAT API. Each soft clause is assigned an assumption literal. When a formula is found to be unsatisfiable, the solver returns a core consisting of these literals, allowing for the direct identification and removal of the corresponding soft clauses.

Proof-based. This method involves generating DRAT proofs, which are written to external files. These proofs are processed using the DRAT-TRIM [7] binary. While this introduces disk I/O overhead and external process calls, DRAT-TRIM can refine the core by trimming redundant clauses that internal heuristics might include, potentially producing smaller, more precise cores.

Glucose 3 (G3) [8] is an integrated solver within PySAT that supports both of these pathways, making it the ideal candidate for evaluating their respective impacts on solver performance and core minimality.

3 Results and Discussion

For an initial comparative analysis, we tested five problems from various domains reduced to MaxSAT, each with known optimums (λ) between 20 and 30. Table 1 presents the results. For these five test instances, there are two sequences showing the size of each UNSAT core found per iteration (UCS). Blue sequences represent the assumptions-based method, while red sequences denote the proof-based method. Higher opacity marks cores found during Phase 1 (disjoint core extraction) of our MSU3 implementation, where the initial lower bound is defined as $\lambda_1 = |UCS_1|$. Lower opacity, then, identifies cores found during Phase 2 (optimization), leading to the final result $\lambda_2 = |UCS|$.

In all instances, both methods correctly identified the optimum ($\lambda_2 = \lambda$). In the first instance, the proof-based method achieved a λ_1 closer to the optimum; however, the assumptions-based method performed better in two others. In the remaining two, both methods were equal, with one of those two instances reaching the optimum exclusively within Phase 1. Despite the small sample size, these results suggest that while proof-based extraction can be competitive, assumptions-based extraction tends to provide a more consistent initial lower bound, hinting at the results found in Table 2.

Instance																										
causal-discovery-causal_n6_i8_N1000_uai14_constant_int																										
40	45	45	40	416	417	553	475	698	972	1395	1278	2947	4466	6111	8311	8127	9821	11468	12156	12027	13873	14039	14925	16462	17836	
40	40	40	40	201	169	269	449	242	508	375	449	874	2086	2619	3740	7857	7672	12355	11384	11895	12546	13889	14847	16442	16856	
extension-enforcement-extension-enforcement_strict_com_150_0.1_2_15_0																										
17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	286	316	
17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	286	348	
mbd-b22_C-mbd14-0272																										
52	62	9	86	32	62	10	53	4	34	141	23	30	43	51	17	20	51	69	110	76	45	7	70	24	2837	3252	5925
4	7	6	7	7	7	7	7	7	9	11	15	24	20	23	45	48	48	86	43	50	60	74	62	264	1541	7157	7441
mqc-10tree510p																										
590	530	650	754	686	608	810	431	2007	4030	5755	10658	9024	10560	9716	10533	10187	11640	11728	11798	12283						
626	646	413	471	982	657	2036	3027	3451	7062	7353	7862	8931	8620	8420	9068	10204	10949	10576	10649	13000						
xai-mindset2-zoo																										
22	24	22	22	23	24	22	22	22	22	22	22	45	45	87	510	600	620	548	623							
22	24	22	22	22	22	22	22	22	22	22	22	440	705	821	919	1064	1146	1300	1143							

Table 1: Comparison of UNSAT core sizes per iteration: • assumptions-based vs. • proof-based extraction

To validate these trends, a broader analysis was conducted using a sample size of 20 distinct instances. We ensured that each instance originated from a different problem reduced to MaxSAT. This sample includes the five instances previously analyzed in Table 1. We evaluate four primary metrics: the average UNSAT core size throughout the execution ($\overline{UCS} = \frac{1}{|UCS|} \sum_{ucs \in UCS} ucs$), the initial lower bound (λ_1), the final solution (λ_2), and total execution time.

In our results, blue markers denote the assumptions-based method, while red markers denote the proof-based method. We also present the official optimum (λ) for each instance. To make the comparison intuitive, we highlight metric results whenever one method outperforms the other. Several patterns and conclusions emerged from these tests:

- **Optimality and Correctness:** Both methods reached the official optimum ($\lambda_2 = \lambda$) in all conducted tests, confirming the robustness of our MSU3 implementation. Notably, the optimum was

[1] https://github.com/jjasferreira/maxsat-core-bench

reached during Phase 1 ($\lambda_1 = \lambda$) in 15 of the 40 total trials. This occurred for both methods in 7 instances, and for the assumptions-based method alone in one additional instance.

- **Lower Bound Precision:** Regarding the initial lower bound (λ_1), the assumptions-based method outperformed the proof-based method in 6 instances, while the proof-based method was superior in only 3.
- **Core Minimality:** In terms of $\overline{UCS}$, assumptions-based extraction achieved a lower average core size in 12 instances, compared to 4 for the proof-based method. In 2 instances, the results were identical, and the remaining 2 instances were satisfiable from the start, yielding no cores.
- **Runtime Efficiency:** While execution time was not a primary objective, we noted that the assumptions-based method was consistently faster. This is expected, as it avoids the disk I/O and external process overhead required by DRAT-TRIM, making it the more practical choice for in-memory solvers.

We observed a potential correlation between achieving a lower average core size ($\overline{UCS}$) and a higher initial lower bound (λ_1). This held true in 7 instances, though it was disproven in 2 others where the average size was skewed by significantly larger cores identified during the complex constraints of Phase 2.

Instance		$\overline{UCS}$	λ_1	λ_2	λ	t(s)
causal-discovery-causal_n6_i8_N1000_uai14_constant_int	•	6113.2	10	26	26	26.242
	•	5303.2	13	26		1707.8
CircuitDebuggingProblems-ac97_ctrl-debug.dimacs	•	21.000	1	1	1	0.5352
	•	21.000	1	1		2.2347
CircuitTraceCompaction-ctrl	•	5789.8	1	15	15	2.5432
	•	5068.9	1	15		683.01
close_solutions-teams16_l6a.cnf	•	1498.6	10	14	14	12.067
	•	794.21	11	14		239.91
extension-enforcement-extension-enforcement_strict_com_150_0.1_2_15_0	•	38.846	26	26	26	6.9658
	•	40.077	26	26		157.69
fault-diagnosis-s38417_nan_explicit_27_0	•	394.60	67	67	67	9.8953
	•	500.28	64	67		1163.1
logic-synthesis-normalized-jac3.opb.msat	•	207.67	14	15	15	2.8863
	•	817.20	13	15		25.110
max-realizability-power-distribution_5_4	•	——	0	0	0	0.0133
	•	——	0	0		0.0124
MaxSATQueriesinInterpretableClassifiers-titanic_test_9_CNF_5_1	•	6.4400	50	50	50	0.1052
	•	6.4400	50	50		3.1776
mbd-b22_C-mbd14-0272	•	471.25	25	28	28	5.9537
	•	610.00	25	28		472.98
min-fill-MinFill_R5_jean	•	1378.8	4	11	11	2.9174
	•	1481.5	4	11		56.612
mqc-10tree510p	•	5951.3	9	21	21	21.889
	•	5952.5	7	21		514.51
privilege-escalation-cloud_access_control_repair_task-56	•	67.444	8	9	9	5.7410
	•	52.111	7	9		115.93
protein_ins-1vii_.1cph_.g.wcnf.t	•	2910.0	2	7	7	0.2079
	•	3056.0	2	7		51.164
pseudoBoolean-normalized-aim-200-6_0-yes1-2.opb.msat	•	226.97	120	200	200	7.2243
	•	318.47	124	200		68.608
ramsey-ram_k5_n14.ra0	•	——	0	0	0	0.0074
	•	——	0	0		0.0079
treewidth-computation-TWComp_hailfinder_N56	•	120.75	4	4	4	1.9727
	•	186.25	4	4		116.11
vpa-UAutomizer_seq_true-unreach-call_true-termination.i_Abstraction1	•	2.0000	299	299	299	1.3803
	•	2.0033	299	299		24.725
wqueens-wqueens18_16.wcsp.dir	•	765.00	3	6	6	0.3127
	•	1234.2	2	6		15.905
xai-mindset2-zoo	•	167.35	15	20	20	0.4073
	•	392.50	12	20		18.601

Table 2: Comparison of performance metrics: • assumptions-based vs. • proof-based extraction

Benchmarks were conducted on a system featuring an Intel Core i7-10750H CPU @ 2.60GHz with 12 GB of RAM. All of the test instances presented above are part of the test set consisting of unweighted instances from the 2024 MaxSAT Evaluation (MSE 2024) exact track [2].

4 Conclusion

Ultimately, Table 1 and Table 2 suggest that while resolution proofs aim to trim redundancy and in our experiment can sometimes produce smaller cores in Phase 1, they lack the robustness required to consistently identify smaller, independent conflicts during the optimization phase of the MSU3 framework. This limitation results in significantly larger cores during later iterations, which increases the overall average core size in comparison to the assumptions-based alternative. In 60% of instances, this approach achieved better core minimality compared to only 20% for the proof-based method.

The λ_1 analysis shows that assumptions-based extraction provides a more effective starting point for the MSU3 algorithm. By identifying more disjoint cores in general, it established a tighter initial lower bound in 30% of cases, compared to just 15% for proof-based extraction. This higher precision often allows the solver to bypass Phase 2 steps or even reach the global optimum during the disjoint core extraction phase.

Even accounting for the fact that execution time is a somewhat biased metric — given that the proof-based method currently suffers from the lack of a native Python library for DRAT extraction — the performance gap remains clear. While proof-based extraction can occasionally offer a strong start, it lacks the overall robustness found in the internal assumption-based approach. The data consistently shows that assumptions provide tighter initial lower bounds and smaller average core sizes throughout the iterative process. Consequently, the internal assumption methodology stands out as the one-fits-all choice for high-performance MAXSAT solvers, if any had to be picked. Despite this, proof-based extraction shows promising potential.

Future research could expand on these findings by (1) analyzing the structural properties of the instances. Key factors such as the ratio of hard to soft clauses, total variable count, and the specific optimum values for the MSE 2024 exact track benchmarks are available for consultation [3]. Investigating how these factors correlate with core sizes and extraction efficiency would clarify performance trade-offs between assumption-based and proof-based methods. Other future work could involve (2) minimizing cores for the assumption-based approach, or (3) guiding core extraction using proofs with information from MAXSAT. This guidance is particularly relevant because the current core sizes are larger than expected, since we only care about the soft clauses.

References

[1] Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971.

[2] John Alan Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.

[3] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maximum Satisfiability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, Frontiers in Artificial Intelligence and Applications, pages 929–991. IOS Press, 2021.

[4] João Marques-Silva and Jordi Planes. On using unsatisfiability for solving maximum satisfiability. *CoRR*, abs/0712.1097, 2007.

[5] Ruben Martins, Saurabh Joshi, Vasco Manquinho, and Inês Lynce. Incremental cardinality constraints for MaxSAT. In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, Lecture Notes in Computer Science, pages 531–548. Springer, 2014.

[6] Alexey Ignatiev, António Morgado, and João Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, Lecture Notes in Computer Science, pages 428–437. Springer, 2018.

[2] https://maxsat-evaluations.github.io/2024/benchmarks.html
[3] https://maxsat-evaluations.github.io/2024/results/exact/unweighted.html

[7] Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, Lecture Notes in Computer Science, pages 422–429. Springer, 2014.

[8] Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, Lecture Notes in Computer Science, pages 309–317. Springer, 2013.