Bima Journal of Science and Technology, Vol. 8(1B) Apr, 2024 ISSN: 2536-6041

DOI: 10.56892/bima.v8i1.626

Comparison of the Empirical Performance of Minisat and GSAT Solvers on SATLIB Benchmark

A. J. Kawu[1]* , G. M. Wajiga[2] and Y. M. Malgwi[2]

[1]Gombe State University, Gombe State

[2]Modibbo Adama University Yola, Adamawa State

Corresponding Author: ahmadkawujibir@gsu.edu.ng

ABSTRACT

In this study, the effectiveness of Minisat(Minimal SAT solver) and Greedy Randomized Adaptive Search (GSAT, Simulated Annealing option) solvers is compared on a number of SATLIB Benchmark problems. Gaining understanding of their effectiveness in resolving SAT cases was the main objective, with an emphasis on application and randomly generated instances. The experimentation showed that Minisat performed better than GSAT in every case, proving to have better solution abilities. A 2.40GHz Core i5 CPU, fitted in a Dell Latitude E7470 laptop, was used for the experiments. All things considered, this research helps to compare the strengths and weaknesses of the GSAT and Minisat solvers and provides insightful information about SAT problem-solving techniques.

Keywords: Boolean Satisfiability, SAT solver, Stochastic Local Search, NP-Complete, Algorithms.

INTRODUCTION

The difficulty of determining whether a propositional logic formula may ever evaluate to true is known as Boolean Satisfiability (SAT). In computer science, this problem has long enjoyed a unique standing. Theoretically speaking, it was the first issue to be declared NP-complete. It is well known that NP-complete problems are notoriously difficult to solve; in the worst scenario, the computing time of any known solution for a problem in this class grows exponentially with the problem instance size (Malik & Weissenbacher, 2012). Satisfiability solvers, despite having the worst-case exponential run time of all known algorithms, are becoming more and more popular as general-purpose tools in a variety of fields, including planning, scheduling, software and hardware verification, automated test pattern generation, and even difficult algebraic problems (Gomes et al, 2008; Janota et al, 2024).

Practically speaking, SAT shows up in a number of significant application areas, including artificial intelligence applications and the design and verification of hardware and software systems. Consequently, there is a great incentive to create SAT solvers that are practically helpful. Nonetheless, the NP-completeness raises doubts because it seems improbable that we will be able to scale the answers to many real-world examples. While there have been ongoing efforts for nearly 63 years to create SAT solvers that are practically useful, for the greatest period of time they were primarily academic endeavors with little prospect of being put to use. Thankfully, a number of relatively recent advances in research have made it possible for us to address cases with millions of variables and constraints. As a result, SAT solvers may now be successfully used in real-world scenarios, such as software system analysis and verification. (Malik & Weissenbacher, 2012) Further evidence that typical-case complexity

DOI: 10.56892/bima.v8i1.626

and the complexity of real-world instances of NP-complete problems are much more amenable to efficient general purpose solution techniques than worst-case complexity results might imply comes from the success of current SAT solvers on numerous real-world SAT instances with millions of variables (Gomes et al, 2008). The SAT problem has been extensively researched globally due to its practical and theoretical applicability. Numerous algorithms are suggested as a result of its popularity. The widely studied SAT solvers are complete and incomplete. Complete SAT solver is the algorithm that checks the satisfiability of the SAT problem. It can tell whether a problem is satisfiable or not. They use DPLL (Davis-Putnam-Logemann Loveland) SAT algorithm with heuristic improvements in selection of variable to branch on, lazy data structures, restart policies and many more techniques. Incomplete SAT solvers employ local search algorithms to randomly search the assignment space using various heuristics such as Simulated Annealing (as in GSAT), Tabu search (as in TSAT), and so forth.

GSAT is one of the original and most popular Stochastic Local Search (SLS also known as incomplete solver) solvers for the SAT. (Hossen, & Polash, 2019). Minisat is a modern Conflict Driven Clause Learning (CDCLalso known as complete solver) SAT solver.

Even though MiniSat and GSAT have both shown a great deal of success in answering SAT instances, the kind of problem cases they are used to solve can greatly affect their performance characteristics. Prior work has demonstrated that various SAT instance kinds, including handcrafted, application-specific, and random, can provide unique difficulties for SAT solvers, requiring a detailed comprehension of solver behavior in a variety of issue domains.

In this paper, we undertake a comparative analysis of the performance of MiniSat and GSAT solvers on three categories of SAT instances: random, application-specific, and handcrafted. Our objective is to elucidate the strengths and weaknesses of each solver paradigm under varying problem conditions, shedding light on their relative efficacy in different problem-solving contexts. The aim of this research is to provide an insight into the performance variation of Minisat and GSAT solvers on a number of SAT instances from different categories drawn from SATLIB and DIMACS benchmarks. Our objectives are to empirically prove the general belief that complete SAT solvers perform better on application problems, while incomplete SAT solvers excel on handcrafted and random SAT instances. This work is motivated by the need to understand the strengths and weaknesses of both complete and incomplete solvers. This will serve as a guide to developing a hybrid solver that will harness the potentials of the two SAT solvers and reduce their shortcomings.

The remaining parts of the paper is organized as follows: Section 2 briefly explain the major concepts of Boolean satisfiability, categories of sat solvers, Minisat and GSAT solvers. Section 3 present a review of some related work. Section 4 described the materials and method used in the experimental setup. Section 5 presents and discussed the experimental results obtained. The last section concludes the paper.

BOOLEAN SATISFIABILITY

Boolean Satisfiability is attracting more interest because nowadays more problems are being solved faster by SAT solvers than other means. This breakthrough was made possible due to the newly developed search techniques. Thus, many problems originating in one of these fields typically have multiple

DOI: 10.56892/bima.v8i1.626

translations to Satisfiability and there exist many mathematical tools available to the SAT solver to assist in solving them with improved performance. Boolean satisfiability problem is the problem of determining whether there is an assignment that satisfies a given propositional formula. A SAT solver is a program that determines whether or not a given formula is satisfiable. A propositional or Boolean formula is a logic expression defined over variables (or literals) that take value in the set {TRUE, FALSE}. (Biere, et al, 2009; Vizel, Weissenbacher, & Malik, 2015)

Stochastic Local Search (Incomplete algorithms)

The effectiveness of local search algorithms is what makes them so popular; on a wide range of problems, they can outperform fully systematic search techniques like backtracking and backjumping. The first type of SAT instance solving algorithms are called stochastic local search (SLS) algorithms; they can be solved quickly and are not complete. Incomplete solutions algorithms SAT makes no promises regarding when it will report a satisfiable assignment or indicate that the provided formula is not satisfiable (Examples of these algorithms include WalkSAT (Selman et al. 1994; Ginsberg & McAllester, 1994), Hill Climbing, Genetic Algorithms, Simulated Annealing and so on. (Larrosa, Lynce & Marques-Silva, 2010).

Complete Algorithms

The satisfiability problem is solved using a second class of methods known as the full algorithm, in which the SAT solver is built on contemporary enhancements or modifications of a Davis-Putnam–Logemann–Loveland (DPLL) algorithm. To locate the satisfying assignments for the variable in the search space, the DPLL SAT solver essentially uses the backtracking search approach (Bordeaux, Hamadi & Zhang, 2006). The search method, sometimes known as the "DPLL" or "DLL" algorithm, was first described in works from the 1960s. According to Marić and Janičić (2010), the DPLL method is sound and complete, which implies that it will only find the answer in the event that the formula is satisfiable.

The satisfiability problem is solved using a second class of methods known as the full algorithm, in which the SAT solver is built on contemporary enhancements or modifications of a Davis-Putnam–Logemann–Loveland (DPLL) algorithm. The majority of SAT solvers available now come in two flavors: look-ahead (Giunchiglia et al. 2003) and conflict-driven (Formisano & Vella 2014). By incorporating efficient features for managing SAT instances in electronic design automation (EDA) tools like conflict analysis, clause learning, non-chronological backtracking (back jumping), "two-watched-literals" unit propagation, adaptive branching, and random restarts, conflict-driven solvers extended the DPLL search routine. Look-ahead solutions are superior to conflict-driven solvers in challenging and complex cases, while conflict-driven solvers perform better in large instances. Look-ahead solvers enable reductions and heuristics.

The benefit of the whole procedure is that it can demonstrate that the formula is unsatisfiable or guarantee the solution of the associated SAT issue. Regretfully, the overall algorithm's efficiency is incredibly low. The worst-case time complexity is exponential, despite the typical time complexity being polynomial. The entire technique actually consists of a depth-first search of the whole solution space; but, because the search space is so big, the computer might not be able to return the results in a reasonable amount of time. The computation time is intolerable due to the combinatorial explosion issue. (Gong & Zhou, 2017)

Bima Journal of Science and Technology, Vol. 8(1B) Apr, 2024 ISSN: 2536-6041

DOI: 10.56892/bima.v8i1.626

GSAT Solver

GSAT is the foundation of WSAT, TSAT, and so forth. Graph coloring, scheduling, binary CSPs, SAT, and other classes of constraint satisfaction issues have all been effectively solved using GSAT. GSAT also pave the way for research into dynamic local search algorithms for SAT and MAXSAT problems by using Initial weights to direct local search (Ishtaiwi & Abu Al-Haija, 2021). Among the many significant factors contributing to the great interest in Stochastic Local Search (SLS) algorithms is the advent of the GSAT algorithm in 1992 (Selman & Kautz, 1993). GSAT was shown to perform well on propositional encoding of N-queens problem, graph coloring problem and Boolean induction problem. As a result, since then, a lot of scholars have focused on researching and looking into ways to create SLS algorithms that are more reliable and effective. As a result, SLS emerged as a primary approach for managing and resolving real-world issues (Ishtaiwi & Abu Al-Haija, 2021). In order to solve a SAT-based problem, the GSAT algorithm deterministically selects a variable with the highest score iteratively and modifies (flips) its value (Steinruecken, 2007). Consequently, the GSAT method either finds a solution or reaches a point where it is unable to identify any additional variables that would lower the cost of the partial solution as it is (Ishtaiwi & Abu Al-Haija, 2021).

```
Procedure GSAT (SATproblem P, MAX TRIES, MAX FLIPS)
    for i=1 to MAX TRIES
     let A be a random initial assignment
      for j=1 to MAX FLIPS if A satisfies P, return true
        else let F be the set of variable-value pairs that, when flipped to, give a maximum increase
        in the number of satisfied constraints;
       pick one f ε F and let new A be current A with f flipped
      end
   end
  return false
end (Kask & Dechter, 1995)
```

Minisat Solver

Modern SAT solvers like the Minisat solver are renowned for their ability to solve Boolean Satisfiability Problems (SAT) quickly and effectively. It utilizes a conflict-driven clause learning (CDCL) algorithm to effectively explore the solution space by fusing potent conflict analysis approaches with effective search strategies. According to Sakallah (2011) Minisat is one of the best implementation of conflict-driven clause learning. It has history of winning SAT competitions, is open-source, and is quite easy to modify.

Related Work

Gent and Walsh (1993) concentrated on evaluating the performance of GSAT on two distinct classes: of SAT instances random structured problems and random uniform problems without any structure. It has been demonstrated that certain structured issues are very easy for the Davis-Putnam technique but very difficult for the GSAT. These issues are so challenging that not even heuristics like clause weighting, which were initially intended to aid in escaping local minimums brought on by the unique structure of the problem, appear to be very helpful. The

DOI: 10.56892/bima.v8i1.626

presence of closely linked variable clusters that are further related loosely by additional global restrictions is a defining trait of these challenges. (Niewiadomski, Switalski, Sidoruk & Penczek, 2019) compared the performance of 8 modern SAT solvers on different hard problems .

Similarly, Hoos and Stutzle (2000) performed a detailed comparative analysis of the GSAT and WalkSAT solvers using benchmark sets that which contains instances from randomized distributions. Their empirical analysis addresses among other issues, how the performance of different solvers compares across a range of problem instances. Dutertre (2020) evaluated the performance of 16 state-of-the-art SAT solvers on a large number of CNF problems drawn from SATLIB repository. They evaluated the solvers based on their performance on the number of instances and total execution time. Mu (2015) investigated the empirical scaling of incomplete (WalkSAT/SKL, BalancedZ and ProbSAT) and complete SAT solvers (kncfs,march_hi and march_br). Sakallah (2011) assessed the effectiveness of seven distinct Minisat 2.2.0 settings using a set of 1000 CNF instances selected from twelve distinct application areas. Sundermann et al (2023) investigated the empirical performance of 21 publicly and available #SAT solvers on 130 feature models from 15 subject system.

MATERIALS AND METHODS

The two SAT solvers Minisat 2.2.0 and GSAT version 35 (written in C++) were downloaded and compiled on a Dell Latitude E7470 laptop. The system has RAM of size 16GB and runs Ubuntu 22.04.3 LTS operating system. The SATLIB benchmark suite which comprise of diverse set of benchmarks that come from random, handcrafted and application instances is selected for evaluation of the GSAT and Minisat solvers. The Minisat Solver run for 900s where if no solution is found it is aborted. On the GSAT side, the version we used have Simulated annealing option. All instance were run with the Simulated Annealing option. The parameters used for the Simulated Annealing option are 300 steps for a temperature of 200 to 20 by factor of 0.8. number of iterations is set to 20 and number of flips was set equal to the number of variables in the instance. After each run a problem instance should have a status; satisfiable, unsatisfiable if a model is obtained or it has been proved unsatisfiable respectively. A status of indeterminate is recorded for Minisat solver if 900s set has elapsed while for GSAT if the number of tries has elapsed without obtaining a model.

Since the two solvers use two different approaches, it is not possible to compare them apple for apple. Hence the metrics used for the comparison are run time and number of solved instances. Each solver is run three times and the average execution time and the status of the instance is recorded.

RESULTS AND DISCUSSION

This section compares the effectiveness of the applied SAT-solvers and discusses the experimental outcomes.

Table 1 shows the number of instances solved by Minisat solver. The result shows that Minisat performed well on all instances except f and g where it aborted on 3 instances.

Table 2 shows the number of instances solved by the GSAT solver. The result shows that the solver aborted on most instances. It succeeded is solving a number of instances in aim, Dubois, ii and jnh categories. Most of the categories where GSAT failed are really instances that are not satisfiable and GSAT can only find a solution if one exist but cannot prove unsatisfiability. This is the reason why they were classified as unsolved.

Bima Journal of Science and Technology, Vol. 8(1B) Apr, 2024 ISSN: 2536-6041

DOI: 10.56892/bima.v8i1.626

Table 1: Number of Instances solved by Minisat Solver

Instance Family	No. of instances	Solved	Aborted
aim50-200	72	72	0
bf	4	4	0
dubois	13	13	0
f	3	0	3
g	4	1	3
hanoi	2	2	0
ii	40	40	0
jnh	50	50	0
logistics	4	4	0
par	30	30	0
hole	5	5	0
ssa	8	8	0

Table 2: Number of instances solved by GSAT solver

Instance Family	No. of instances	Solved	Aborted
aim50-200	72	15	60
bf	4	0	4
dubois	13	8	5
f	3	0	3
g	4	0	4
hanoi	2	0	2
ii	40	6	34
jnh	50	14	36
logistics	4	0	4
par	30	8	22
hole	5	0	5
ssa	8	0	8

Figure 1 depicts the execution time of Minisat solver with initial random activity and without initial random activity. The result shows that random initial activity has significant impact on a few sat instance like hanoi category in the categories selected for this experiment.

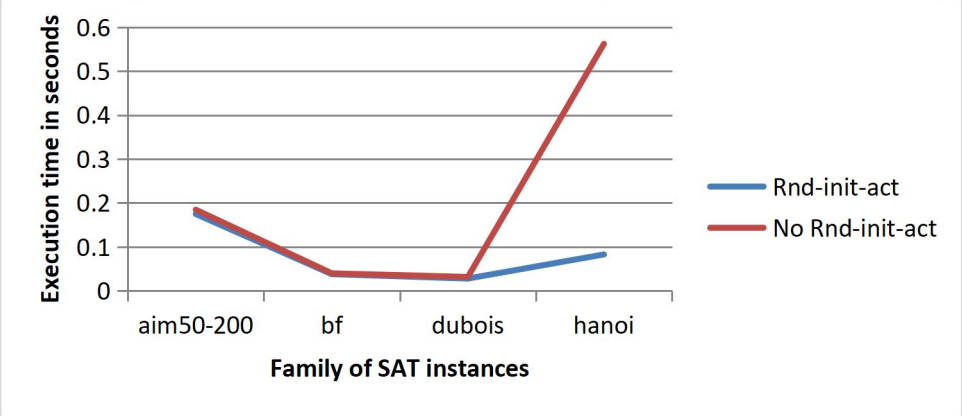

Figure 1: Comparison of the execution time of Minisat with and without random initial activity

Figure 2 shows the execution time of GSAT and Minisat solvers on the selected SAT instances. The result shows that Minisat perform better than GSAT on most instances. However, this is mostly due to the fact that most of such instances are unsatisfiable and Minisat (CDCL solver) quickly prove unsatisfiabilty by deriving an empty clause. This makes the execution time shorter compared to GSAT solver that has to perform

DOI: 10.56892/bima.v8i1.626

all the number of iterations and flips before reporting inability to find a solution.

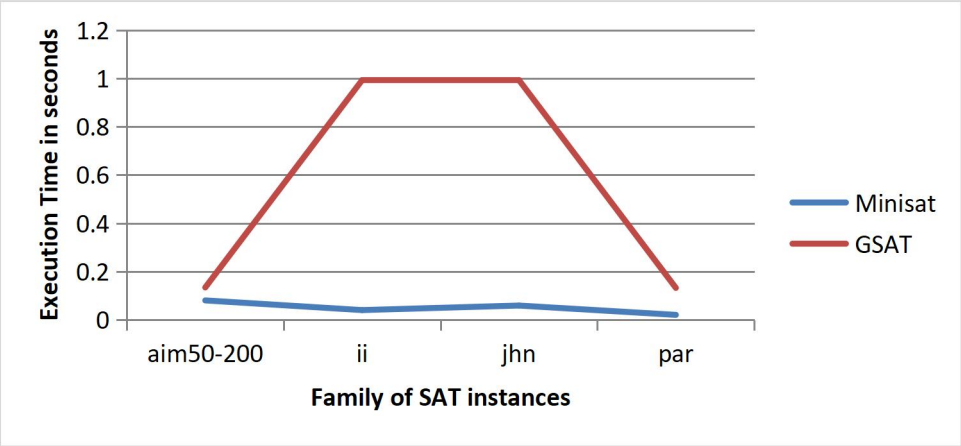

Figure 2: Comparison of the execution time of Minisat and GSAT on commonly solved instances

Table 3 depicts the number of instances solved correctly by GSAT and Minisat. The result shows that out of the 24 satisfiable instances of the Aim category, GSAT was able to solve only 15. However, Minisat solved all the 72 instances. The same result was obtained for the other random instances jhn and Dubois. Furthermore, Minisat also outperformed GSAT on Inductive inference (II) and Logistics instances.

Table 3: Comparison of the number of instances solved by Minisat and GSAT

Instance Family	No. of instances	Minisat	GSAT
aim50-200	72	72	15
bf	4	4	0
dubois	13	13	8
f	3	0	0
g	4	1	0
hanoi	2	2	0
ii	40	40	6
jnh	50	50	14
logistics	4	4	0
par	30	30	8
hole	5	5	0
ssa	8	8	0

CONCLUSION

This paper compares the performance of GSAT and Minisat solvers on a selected SAT instances from SATLIB and DIMACS. This is to give more insight on the strengths and weaknesses of the two SAT solvers and serve as a guide to develop hybrid solvers that can perform well on different classes of SAT instance. Our aim is to gain more insights into the strengths and weaknesses of the two SAT solvers compared in this research. This will serve as a guide to development of Hybrid SAT solvers.

The results obtained showed that Minisat performs better on both random and application SAT instances. We suggest further research on fine tuning the Simulated Annealing parameters to find the best for both random application instances.

REFERENCES

Balint, A. (2014). Engineering stochastic local search for the satisfiability problem

DOI: 10.56892/bima.v8i1.626

(Doctoral dissertation, Universität Ulm).

Balint, A., Belov, A., Järvisalo, M., & Sinz, C. (2015). Overview and analysis of the SAT Challenge 2012 solver competition. *Artificial Intelligence, 223,* 120-155.

Beringer, A., Aschemann, G., Hoos, H. H., Metzger, M., & Weiss, A. (1994). GSAT versus simulated annealing. In *ECAI* (Vol. 94, pp. 130-134). PITMAN.

Biere, A., Heule, M., van Maaren, H., and Walsh, T. (2021). In *Handbook of Satisfiability* (pp. 131- 153). Frontiers in Artificial Intelligence and Applications.

Bordeaux, L., Hamadi, Y., & Zhang, L. (2006). Propositional satisfiability and constraint programming: A comparative survey. *ACM Computing Surveys (CSUR)*, *38*(4), 12-es.

Chu, Y., Luo, C., Cai, S., & You, H. (2019). Empirical investigation of stochastic local search for maximum satisfiability. *Frontiers of Computer Science, 13,* 86-98.

Dutertre, B. (2020). An Empirical Evaluation of SAT Solvers on Bit-vector Problems. In *SMT* (pp. 15- 25).

E´en and S¨orensson. MiniSAT (version 2.2.0) http://minisat.se/downloads/minisat-2.2.0.tar.gz, 2021

Formisano, A., & Vella, F. (2014). On multiple learning schemata in conflict driven solvers. In CEUR Workshop Proceedings Volume 1231, 2014, Pages 133-146 (Vol. 1231, pp. 133-146).

Gent, I. P., & Walsh, T. (1993). An empirical analysis of search in GSAT. *Journal of Artificial Intelligence Research*, *1*, 47-59.

Ginsberg, M. L., & McAllester, D. (1994). GSAT and dynamic backtracking. In *International Workshop on Principles and Practice of Constraint Programming* (pp. 243-265). Berlin, Heidelberg: Springer Berlin Heidelberg.

Gning, S., & Mailly, J. G. (2020). On the Impact of SAT Solvers on Argumentation Solvers. In *SAFA@ COMMA* (pp. 68-73).

Gomes, C. P., Kautz, H., Sabharwal, A., & Selman, B. (2008). Satisfiability solvers. *Foundations of Artificial Intelligence*, *3*, 89-134.

Gong, W., & Zhou, X. (2017). A survey of SAT solver. In *AIP Conference Proceedings* (Vol. 1836, No. 1). AIP Publishing.

Gu, J. (1993). Local search for satisfiability (SAT) problem. *IEEE Transactions on systems, man, and cybernetics, 23*(4), 1108-1129.

Gu, J., Purdom, P. W., Franco, J., & Wah, B. W. (1996). Algorithms for the satisfiability (SAT) problem: A survey. *Satisfiability problem: Theory and applications, 35,* 19-152.

Hoos, H. H., & Stützle, T. (2000). Local search algorithms for SAT: An empirical evaluation. *Journal of Automated Reasoning*, *24*(4), 421-481.

Hoos, H. H., & Stützle, T. (2000). SATLIB: An online resource for research on SAT. *Sat*, *2000*, 283- 292.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2010). Tradeoffs in the empirical evaluation of competing algorithm designs. *Annals of Mathematics and Artificial Intelligence, 60,* 65-89.

Ishtaiwi, A., & Abu Al-Haija, Q. (2021). Dynamic Initial Weight Assignment for MaxSAT. *Algorithms*, *14*(4), 115.

DOI: 10.56892/bima.v8i1.626

Janota, M., Chow, C., Araújo, J., Codish, M., & Vojtechovský, P. (2024). SAT-based Techniques for Lexicographically Smallest Finite Models.

Kask, K., & Dechter, R. (1995). GSAT and local consistency. In *IJCAI (1)* (pp. 616-623).

Katebi, H., Sakallah, K. A., & Marques-Silva, J. P. (2011). Empirical study of the anatomy of modern SAT solvers. In *International conference on theory and applications of satisfiability testing* (pp. 343-356). Berlin, Heidelberg: Springer Berlin Heidelberg.

Kautz, H. A., Sabharwal, A., & Selman, B. (2021). Incomplete Algorithms. *Handbook of satisfiability*, *185*, 185-203.

Koundinya, P. P., Reddy Y, S. K., Deepak, V. M., Rutwesh, K., & Deshpande, A. (2021). Test Set Generation for Multiple Faults in Boolean Systems using SAT Solver. *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT),* 1-5. doi:10.1109/ICCCNT51525.2021.9579930.

Larrosa, J., Lynce, I., & Marques-Silva, J. (2010). Satisfiability: Algorithms, Applications and Extensions. SAC.

Lee, E. A., Roychowdhury, J., & Seshia, S. A. (2010). Fundamental Algorithms for System Modeling, Analysis, and Optimization. *University of California, Berkeley.*

Li, G. (2004). Search space structures of local search algorithms for SAT.

Malik, S., & Weissenbacher, G. (2012). Boolean satisfiability solvers: techniques and extensions. *Software Safety & Security Tools for Analysis and Verification; IOS Press: Amsterdam, The Netherlands*.

Mitchell, D., Selman, B., & Leveque, H. (1992). A new method for solving hard satisfiability problems. In *Proceedings of the tenth national conference on artificial intelligence (AAAI-92)* (pp. 440-446).Mu, Z. (2015). *Analysing the empirical time complexity of high-performance algorithms for SAT and TSP* (Doctoral dissertation, University of British Columbia).

Sakallah, K. A. (2011). Anatomy and empirical evaluation of modern SAT solvers. *Bulletin of the EATCS*, (103), 96-121.

Selman, B., & Kautz, H. (1993). Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *IJCAI* (Vol. 93, pp. 290-295).

Steinruecken, C. (2007). SAT-Solving: Performance Analysis of Survey Propagation and DPLL.

Sundermann, C., Heß, T., Nieke, M., Bittner, P. M., Young, J. M., Thüm, T., & Schaefer, I. (2023). Evaluating state-of-the-art# SAT solvers on industrial configuration spaces. Empirical Software Engineering, 28(2), 29.

Vizel, Y., Weissenbacher, G., & Malik, S. (2015). Boolean satisfiability solvers and their applications in model checking. *Proceedings of the IEEE*, *103*, pp. 2021-2035.