Mathematical Modeling and Algorithm Application
ISSN: 3006-0842 | Vol. 2, No. 1, 2024

A Review of Research on Algorithms for Solving SAT Problems

Mengyu Guo[1, *]

[1] School of Software, Henan Polytechnic University, Jiaozuo, Henan Province, China
[*] Corresponding Author

Abstract: The SAT problem solving algorithms are an important research area in computer science aimed at solving the Boolean satisfiability problem. With the wide application of SAT problems in the fields of hardware design, automated planning, and artificial intelligence, researchers have proposed a variety of efficient solution algorithms. The classical algorithms include exhaustive search and backtracking search, but they are less efficient when dealing with large-scale problems. Heuristic search methods accelerate the search process by introducing randomization or evolutionary algorithms to improve the solution efficiency. Learning-based methods utilize learning and memorizing the explored search space or learning heuristic information to guide the search process. The CDCL algorithm, on the other hand, is an efficient conflict-driven learning algorithm that achieves impressive results by learning and applying conflict information to prune the search space. Parallel and distributed solving as well as quantum computing methods are hot research topics in recent years, which provide new ideas and technical support for solving large-scale problems. The research on algorithms for solving SAT problems continues to innovate, providing powerful tools and technical support for real-world problem solving.

Keywords: Exhaustive search; backtracking search; SAT solver algorithm; Conflict-Driven Learning Algorithms.

1. Introduction

The study of methods for solving SAT problems has received widespread attention and intensive research in the field of computer science since the 1970s.The SAT problem, or Boolean satisfiability problem, is a classical combinatorial optimization problem in which the objective is to determine whether there exists a set of assignments of variables to a given Boolean formula that makes the formula true.The SAT problem is proved to be an NP-complete problem[1], which means that No efficient algorithm has been found to solve the problem in polynomial time, but it is possible to verify the correctness of a given solution in polynomial time. Due to its importance and wide range of applications, the solution of the SAT problem has been a research hotspot in computer science, attracting the attention and efforts of many researchers.

With the continuous progress of computer hardware and algorithm technology, SAT solving methods are also developing and evolving. Early SAT solving methods are mainly based on exhaustive search or heuristic search algorithms, such as Davis-Putnam-Logemann-Loveland (DPLL) algorithm[2] and WalkSAT algorithm[3]. These algorithms are more effective in dealing with small-scale problems, but when facing large-scale and complex SAT instances, it is often difficult to obtain satisfactory results.In this paper we summarize the research progress of the SAT problem around three aspects: theoretical foundations, solution algorithms and practical applications, introduce the applications of the SAT problem in different fields, discuss the challenges faced by the SAT problem, and provide possible future research directions.

In recent years, with the improvement of computer hardware performance and the continuous innovation of algorithm technology, SAT solving methods have made great progress. Among them, the solution algorithms based on CDCL (Conflict-Driven Clause Learning)[4] have become the mainstream.CDCL algorithms dynamically adjust the search strategy by learning and applying the conflict information, which improves the solution efficiency and solving capability. Representative CDCL algorithms include MiniSat[5], Glucose[6], and so on. These algorithms have achieved excellent results in SAT competitions and become effective tools for solving large-scale SAT problems.

In addition, some new solving methods have emerged in recent years, such as quantum SAT solvers[7, 8], parallel and distributed solving algorithms[9] and learning-based solving algorithms[10]. The development of quantum computing technology has brought new ideas and opportunities for solving SAT problems, and although quantum SAT solvers are still in the early stage, their potential acceleration effects have attracted more and more researchers' attention. Meanwhile, with the development of computer clusters and cloud computing technology, researchers have also begun to explore parallel and distributed methods for solving SAT problems to cope with the increasing problem size and complexity.

The research progress of SAT problem solving methods is increasingly rapid, from the traditional exhaustive search and heuristic search to the efficient algorithms based on CDCL, and then to the latest quantum computation and parallel solving techniques, each step of which provides new ideas and tools for solving practical problems. With the continuous development and progress of science and technology, it is believed that SAT problem solving methods will continue to innovate and break through, providing more effective solutions to solve more complex practical problems.

2. Key Steps of The SAT Problem Solving Algorithm

The SAT problem solving requires attention to modeling accuracy, selection of appropriate solution algorithms, optimization of search strategies, handling of the search space, validation and interpretation of results, as well as performance evaluation and tuning to ensure that real-world

problems can be solved effectively. In applying SAT problems to solve real-world problems, first, the problem to be solved is transformed into the form of a SAT problem, i.e., the problem is expressed as a Boolean formula. This stage requires appropriate abstraction and modeling of the original problem, transforming the variables, constraints, etc. in the problem into Boolean variables and logical relationships to form a logical representation of the SAT problem. After determining the logical representation of the SAT problem, it is necessary to choose an appropriate solution strategy. According to the scale of the problem, structural characteristics and algorithm performance and other factors, select the appropriate solution algorithm and heuristic strategy. Common solution strategies include exhaustive search, heuristic search, and learning-based methods.

During the solution process, the algorithm continuously searches the solution space for an assignment scheme that satisfies the Boolean formula according to the selected strategy. This process usually involves steps such as assigning values to variables, logical reasoning, conflict detection and learning. Through continuous search and reasoning, it tries to find a solution that satisfies the condition or prove that no such solution exists.

(1) Variable decision-making

In SAT problem solving, the variable decision is the decision of which variable to choose for assignment during the search process. A suitable choice of variables can significantly affect the efficiency and performance of the solution algorithm. Good decision variables can lead to a smooth search process. The solution is found by absconding, while poor decision variables may lead to the search to the last leaf node of the binary tree before the solution is obtained. Various decision-making strategies have been proposed in order to select the key variables initially in the solution process. Jeroslow-Wang (JW) algorithm [11], Bohm algorithm [12], Maximum Occurrence in Minimization (MOM) algorithm [13] and Dynamic Largest Individual Sum (DLIS) algorithm [4], all of these algorithms require frequent operations on the set of clauses to find those specific clauses that match the conditions, and therefore require a lot of time. Individual Sum (DLIS) algorithms, all of which require frequent operations on the set of clauses to find those specific clauses that meet the conditions, and thus take a lot of time. In recent years, the mainstream heuristic branching decision strategies are based on conflict-based variable activity evaluation. Some of the better performing algorithms are Variable State Independent Decaying Sum(VSIDS)[14], Conflict History-based Branching(CHB)[15], and Global Learning Rate(GLR)[16].

(2) Conflict Analysis

During the solution process, if the current variable assignment leads to a conflict, i.e., a null clause (all literals are false), then a conflict analysis is required. Conflict analysis and learning is required when the algorithm encounters a conflict during the search process. Usually the method of reverse inference is used, starting from the empty clause of the conflict and deducing forward, and gradually finding out where the cause of the conflict lies. Conflict analysis can get one or more key clauses which are the root causes of the conflict. By analyzing the cause of the conflict, the clauses or variables that lead to the conflict can be found and learned based on the conflict information. The learning process can help the algorithm avoid similar conflicts and adjust the search strategy to improve the solution efficiency.

(3) Clause Learning

Once the key clauses leading to the conflict have been identified, these clauses are added to the formula to form new clauses. These learned clauses are usually obtained by applying some transformations to the text in the clauses that led to the conflict to ensure that the learned clauses are different from the original clauses so as to avoid getting stuck in a dead-end loop. Learned clauses may contain new assignment information, which needs to be propagated to other unassigned variables. This can be achieved by techniques such as single clause propagation, which further reduces the search space. The current mainstream SAT solver Minisat uses this technique; JIN et al. propose a robust conflict analysis method that additionally generates an appended learning clause that complements the standard learning clause[17]; HAMADI et al. summarize and introduce the mainstream learning methods in the CDCL SAT solver[18].

(4) Backtracking

When a conflict occurs, backtrack to the previous decision point, undo the previous assignment, and select an alternative assignment method. If no other assignment method exists, it continues to backtrack to an earlier decision point. Improvement methods based on the backtracking process can effectively improve the efficiency and performance of the SAT problem solving algorithm. These improvement methods mainly include pruning strategy, learning and memory, parallelization, heuristic information and local search. By combining these methods, unnecessary searches can be reduced, repeated computations can be avoided, the search process can be accelerated, and the solution space can be explored more efficiently in the search space. These improved methods make the backtracking-based SAT solving algorithm perform better when dealing with large-scale and complex SAT problems, and improve the practicality and application value of the algorithm.

(5) Reboot

Restart is a heuristic technique used in the SAT solving process to avoid falling into local optima and to improve the global search capability of the algorithm. The basic idea of restart is to periodically abandon the current search state during the search process and restart the search from the initial state in the expectation of reactivating the search process and finding the solution faster. The key steps of the restart technique include setting a restart strategy, executing a restart operation, avoiding repeated searches, and optimizing the restart strategy. By setting appropriate restart conditions and frequency, as well as optimizing the restart strategy, the global search capability of the algorithm can be effectively improved and the solution process can be accelerated. Based on experimental results, GOMES et al. show that there is a heavy-tailed distribution phenomenon in the combinatorial search process[19], most of the backtracking process in the search has little effect on the solution results, and only a few variable assignments play a key role in obtaining the final results. For larger instances, often the decisions at the more top level of the binary tree have a greater impact on the search space, but it is possible that the most primitive decision order is not the best decision order. Restarting means undoing all variable assignment operations, re-selecting decision variables for assignment, and then restarting a new round of search. nested geometric growth sequence restarting used in the PicoSat solver[20]; and Luby sequence restarting used in the MiniSAT version 2.2.2 solver[21]. The dynamic restart

strategy has shown better solution performance in international SAT competitions in recent years.

The more mainstream algorithm for solving SAT problems is based on the conflict-driven clause learning algorithm. The CDCL algorithm dynamically adjusts the search path and gradually narrows the search space by continuously performing the steps of variable assignment, conflict detection, conflict analysis, and the addition of learning clauses to finally find a solution that satisfies the conditions or to prove that no solution exists. This conflict-driven solution strategy makes the CDCL algorithm achieve better results in practice and become one of the core algorithms in modern SAT solvers.

3. Algorithms for Solving SAT Problems

3.1. The classical solution method

Traditional solution algorithms for SAT problems mainly use the ideas of exhaustion and backtracking.

Exhaustive Search: The most straightforward approach is to search for all possible variable assignments by exhaustive search and then verify that each assignment satisfies the Boolean formula. This method is only applicable to very small scale problems in practice.

Backtracking Search: Backtracking search is a classical method for solving SAT problems, which traverses the search space by means of depth-first search and uses a pruning strategy to improve the search efficiency.

The most primitive completeness algorithm is the Davis-Putnam algorithm proposed by Davis et al[22]. The core idea of the Davis-Putnam algorithm is to gradually find assignment cases that satisfy all clauses in the search space by backtracking the search. By applying single clause propagation and suitable variable selection strategies, the search space can be cut down during the search process and the efficiency of the algorithm can be improved. However, the Davis-Putnam algorithm is a completion algorithm whose time complexity depends on the size and structure of the problem, so it may face efficiency problems when dealing with large-scale and complex SAT problems. To solve this problem, the DPLL algorithm improves on the Davis-Putnam algorithm[2]. The traditional DPLL algorithm uses a branch-and-bound search strategy to recursively search the solution space and simplifies the problem by single clause propagation and variable assignment. While the modern Conflict-Driven Clause Learning algorithm is based on the DPLL algorithm and introduces a conflict-driven learning mechanism, which dynamically adjusts the search path through continuous learning and learning clause addition to effectively reduce the search space and improve the solution efficiency. The CDCL algorithm is a conflict-driven learning-based algorithm for efficient SAT solving[4]. It iteratively searches the solution space and learns conflicts to prune the search space to achieve the purpose of solving SAT problems quickly. Representative implementations of the CDCL algorithm include MiniSat, Glucose, and so on, which have achieved good results in SAT competitions, and become the mainstream algorithms for solving large-scale SAT problems.

In practice, complete algorithms are often considered one of the most desirable solutions because they guarantee that a solution to the problem will be found in a finite amount of time. However, in some cases, complete algorithms may not be practical due to the excessive complexity of the problem or resource constraints, and it may be necessary to use approximation or heuristic algorithms to solve the problem.

3.2. Solution Methods for Heuristic Search

Heuristic search is a search method based on empirical and heuristic information used to solve complex problems where the search space is too large. In heuristic search, the algorithm selects the most promising path to search based on a predefined heuristic function to find the solution or reach the goal of the problem as soon as possible.

The state space of the problem is first represented in a suitable way as a graph or tree structure. The state space represents the possible states of the problem and the transitions between them. Then, a heuristic function is defined to evaluate how "good or bad" the current state is. The heuristic function can be designed according to the characteristics of the problem and empirical knowledge, and usually includes some estimates or rules to guide the search process. Based on the current state and the heuristic function, the most promising states are selected for expansion. This usually involves generating a successor state to the current state and evaluating the merits of the successor state based on the heuristic function. Based on the selected heuristic function and state expansion strategy, a suitable search strategy is selected for the search. During the search process, detect whether the goal state of the problem is reached or the termination condition of the problem is satisfied. If the target state is reached, the search ends; otherwise, continue with the next state expansion and search.

The more popular solution methods of heuristic search mainly include: randomization algorithms explore the search space by randomly selecting the assignment of variables, such as WalkSAT algorithm[3], GSAT algorithm[23] and so on. Evolutionary algorithms search for optimal solutions by simulating the process of biological evolution, such as genetic algorithms, simulated annealing algorithms and so on. Simulated annealing algorithms search for the optimal solution by randomly perturbing the current solution and accepting the worse solution according to a certain probability in order to jump out of the local optimal solution and find the global optimal solution.

Heuristic search methods are applicable to many fields, including artificial intelligence, operations research, and computer vision. It can help find solutions efficiently in large-scale search spaces and improve the search efficiency and performance of algorithms. However, heuristic search methods also have some limitations, such as the possibility of falling into local optimal solutions, and the design of heuristic functions requires some experience. Therefore, it is necessary to carefully choose the appropriate heuristic function and search strategy when applying the heuristic search method to ensure that satisfactory results are obtained.

3.3. Learning-based solution methods

Machine learning based SAT solving algorithms attempt to utilize machine learning techniques to improve the efficiency of solving SAT problems. These algorithms are usually trained based on a large number of SAT problem instances and utilize machine learning models to predict variable assignments or aid in the solution process.

There are five main categories of learning-based SAT solving methods that are more popular today.

(1) Classifier-based approach: This approach transforms a SAT problem into a classification problem, where each

training sample corresponds to a SAT problem instance and its solution (satisfying or unsatisfying). A classifier is then trained using a supervised learning algorithm, such as a support vector machine (SVM), a decision tree, or a neural network, to predict the solution of a new SAT problem instance.

(2) Reinforcement learning-based approach: this approach treats the SAT solving process as a reinforcement learning problem, where the SAT solver acts as an intelligent body that chooses an action (i.e., assignment of a variable) based on the current state (i.e., the state of the CNF formula) and receives a reward or penalty based on the outcome of the action (i.e., whether it leads to a conflict). Reinforcement learning algorithms, such as Q-learning or deep reinforcement learning, are then used to learn a policy to maximize future rewards[24, 25].

(3) Feature engineering based approach: this approach transforms a SAT problem into a feature vector representation, where each feature corresponds to some property or characteristic of the SAT problem. A model is then trained using feature engineering techniques and traditional machine learning algorithms, such as logistic regression, random forests, or gradient boosting trees, to predict the solution of the SAT problem.

(4) Approach based on graph neural network modeling: the SAT problem is represented as a graph structure, where variables and clauses correspond to the nodes of the graph, and dependencies between variables and text in clauses correspond to the edges of the graph. The graph neural network is then utilized to learn the graph representation in order to predict variable assignments or solutions[26].

(5) Integrated Learning Approach: this approach integrates multiple different SAT solvers or machine learning models to improve solution efficiency and performance. For example, multiple learning-based SAT solvers or traditional SAT solvers can be integrated to obtain the final solution using voting or weighted averages[27, 28].

These machine learning SAT solving algorithms improve the efficiency and performance of solving SAT problems to some extent, but they also face some challenges, such as data sparsity, generalization ability, and training time. Therefore, appropriate feature representations, model structures, and training methods need to be carefully selected when applying these algorithms to obtain the best performance and results.

3.4. Parallel and Distributed Solution Methods

With the development of computer hardware and parallel computing technology, researchers have begun to explore parallel and distributed methods for solving SAT problems in order to speed up the solution process and deal with large-scale problems. The methods for solving SAT problems in parallel include distributed search, parallelized search, and multi-core search. Recently, with the advancement of quantum computing technology, some researchers have started to try to solve SAT problems using quantum computers. With the potential of parallel processing capability and exponential acceleration, quantum computing may revolutionize SAT problem solving. The most classical is the SATZilla solver designed by Xu et al. This solver, on the other hand, utilizes existing solvers for the purpose of solving more instances or more efficiently, which with the help of machine learning algorithms designs a system that predicts the most suitable solver for a given SAT problem, i.e., the one that is able to solve the instance the fastest. Similarly, the solver HordeSat utilizes a series of existing SAT solvers to solve instances in parallel on the cluster until one of the solvers finds a solution for the instance.

These parallel and distributed SAT solving methods can significantly improve the efficiency and performance of the SAT solving process, especially when dealing with large-scale and complex SAT problems, which can fully utilize the computational resources and accelerate the solving process. However, there are some challenges in designing parallel and distributed solving methods, such as task partitioning, communication overhead, load balancing, etc., and various factors need to be considered to design appropriate parallel and distributed solving strategies.

SAT problem solving methods have made great progress in both theory and practice, from the initial classical methods to the current efficient algorithms, researchers continue to explore and innovate, providing powerful tools and technical support for solving practical problems.

4. Summary

The study of algorithms for solving SAT problems has been one of the hotspots in the field of artificial intelligence. Modern SAT solvers are mainly based on the CDCL (Conflict-Driven Clause Learning) algorithm. In recent years, researchers have optimized the CDCL algorithm by improving the learning strategy, pruning technique, and variable selection strategy to improve the solving efficiency and performance. With the increasing computational resources, researchers have focused on how to utilize parallel computing and distributed computing resources to accelerate the SAT solving process. Parallelization and distributed solving techniques have become important means to improve the efficiency of SAT solving.

In recent years, researchers have tried to combine the SAT solving algorithm with other solving techniques to form a hybrid solving strategy, such as combining SAT solving with integer programming, simulated annealing, and genetic algorithms to improve the applicability and efficiency of the algorithm. With the development of quantum computing technology, researchers have begun to explore the use of quantum computing to solve SAT problems. Quantum SAT solving algorithms can theoretically provide exponential speedups and have great potential, although they are still in the experimental stage. Besides, with the development of deep learning technology, the method of solving SAT problems based on neural network models has been widely studied, and researchers try to learn the solution strategies of SAT problems by training neural networks. The advantage of deep learning for solving SAT problems is that it can learn the structural features and solution strategies of the problem, thus improving the solution efficiency and performance to some extent. However, this approach also faces challenges such as difficulties in data labeling, long training time, and weak model generalization ability. Therefore, how to design effective neural network architectures, appropriate training data and loss functions, etc. are still hot research topics in deep learning for solving SAT problems.

Future work should focus on further improving the performance and efficiency of SAT solving algorithms. This includes optimizing the search strategies, pruning techniques, and learning mechanisms of the traditional algorithms, as well as exploring new algorithmic frameworks and optimization techniques to cope with larger scale and more complex SAT problems. In addition to applications in traditional computer

science fields, SAT problem solving algorithms can be expanded to more practical application areas, such as artificial intelligence, bioinformatics, automation design, etc., to provide more options and possibilities for solving practical problems.

References

[1] Cook S. The complexity of theorem-proving procedures: Proceedings of the third annual ACM symposium on Theory of computing[C], 1971.

[2] Davis M, Logemann G, Loveland D. A machine program for theorem-proving[J]. Journal of the ACM, 1962,5(7):394-397.

[3] Selman B, Kautz H A, Cohen B. Local Search Strategies for Satisfiability Testing[J]. cliques coloring & satisfiability second dimacs implementation challenge, 1996.

[4] Silva J P M, Sakallah K A. GRASP-A new search algorithm for satisfiability: Proceedings of International Conference on Computer Aided Design[C], San Jose, CA, USA, 1996.

[5] Luby M, Sinclair A, Zuckerman D. Optimal speedup of Las Vegas algorithms[J]. Information Processing Letters, 1993,47(4):173-180.

[6] Audemard G, Simon L. Predicting Learnt Clauses Quality in Modern SAT Solver: International Joint Conference on Artificial Intelligence[C], 2009.

[7] Glover F, Kochenberger G, Hennig R, et al. Quantum bridge analytics I: a tutorial on formulating and using QUBO models[J]. Annals of Operations Research, 2022,314(1):141-183.

[8] Reifenstein S, Leleu T, McKenna T, et al. Coherent SAT solvers: a tutorial[J]. Advances in Optics and Photonics, 2023,15(2):385-441.

[9] Tiejun L, Kefan M, Jianmin Z. An Parallel FPGA SAT Solver Based on Multi-Thread and Pipeline[J]. Chinese Journal of Electronics, 2021,30(6):1008-1016.

[10] Guo W, Zhen H, Li X, et al. Machine Learning Methods in Solving the Boolean Satisfiability Problem[J]. Machine Intelligence Research, 2023,20(5):640-655.

[11] Jeroslow R G, Wang J. Solving propositional satisfiability problems[J]. Annals of Mathematics and Artificial Intelligence, 1990,1(1):167-187.

[12] Kotelnikov V A, Kotelnikov M V. Use of the Bohm's formula and its analogues in probe diagnostics[J]. High Temperature, 2017,55(4):477-480.

[13] Freeman J W. Improvements to propositional satisfiability search algorithms[M]. University of PennsylvaniaComputer and Information Science Dept, 2000.

[14] Moskewicz M W, Madigan C F, Zhao Y, et al. Chaff: engineering an efficient SAT solver: Proceedings of the 38th Design Automation Conference[C], Las Vegas, NV, USA, 2001.

[15] H L J, V G, P P. Exponential recency weighted average branching heuristic for SAT solvers: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence[C], 2016.

[16] Liang J H, V. K. H G, Poupart P, et al. An Empirical Study of Branching Heuristics Through the Lens of Global Learning Rate: Theory and Applications of Satisfiability Testing – SAT 2017[C], 2017.

[17] Jin H, Somenzi F. Strong Conflict Analysis for Propositional Satisfiability: Proceedings of the Design Automation & Test in Europe Conference[C], Munich, Germany, 2006.

[18] Hamadi Y, Jabbour S, Saïs L. What we can learn from conflicts in propositional satisfiability[J]. Annals of Operations Research, 2016,240(1):13-37.

[19] Gomes C P, Selman B, Crato N, et al. Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems[J]. Journal of Automated Reasoning, 2000,24(1):67-100.

[20] Biere A. PicoSAT Essentials.[J]. Journal on Satisfiability Boolean Modeling and Computation, 2008,2-4(4):75-97.

[21] Luby M, Sinclair A, Zuckerman D. Optimal speedup of Las Vegas algorithms[J]. Information Processing Letters, 1993, 47(4):173-180.

[22] M D, H P. A Computing Procedure for Quantification Theory [J]. Journal of the ACM, 1960,3(7):201-215.

[23] Selman B, Levesque H, Mitchell D. A New Method for Solving Hard Satis ability Problems: In Proceedings of the tenth national conference on Artificial intelligence[C], 1992.

[24] Kurin V, Godil S, Whiteson S, et al. Can Q-Learning with Graph Networks Learn a Generalizable Branching Heuristic for a SAT Solver? Neural Information Processing Systems[C], 2020.

[25] Ozolins E, Freivalds K, Draguns A, et al. Goal-Aware Neural SAT Solver[J]. International Joint Conference on Neural Networks, 2021:1-8.

[26] Bünz B, Lamm M. Graph Neural Networks and Boolean Satisfiability[J]. arXiv preprint, 2017.

[27] Balyo T, Sanders P, Sinz C. HordeSat: A Massively Parallel Portfolio SAT Solver: Theory and Applications of Satisfiability Testing[C], 2015.

[28] Xu L, Hutter F, Hoos H H, et al. SATzilla: Portfolio-based Algorithm Selection for SAT[J]. Journal of Artificial Intelligence Research, 2008,32(1):565-606.