International Journal of Computer Science and Information Technology
ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 2, Number 2, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v2n2.25
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Progress in the Study of the Boolean Satisfiability Problem

Mengyu Guo*

School of Software, Henan Polytechnic University, Jiaozuo, Henan Province, China
*Corresponding Author: Mengyu Guo

ABSTRACT

The Boolean Satisfiability problem (SAT) is a classical problem in computer science and artificial intelligence, and the most widely studied NP-complete problem. It aims to determine whether there exists an assignment of a value to a variable of a given Boolean formula that makes the formula true. In recent years, with the continuous improvement of theoretical research and practical application needs, the research on SAT problems has achieved remarkable results. These advances have not only promoted the in-depth study of the SAT problem itself, but also provided strong support for the development of computer science, artificial intelligence and other related fields. Currently, the research of SAT problem involves several aspects, this paper mainly focuses on the three aspects of the improvement of SAT solving algorithm, the deepening of the theoretical foundation of SAT and the expansion of the application area to sort out the related research results, summarize the limitations of the existing methods, and propose possible future work.

KEYWORDS

The Boolean satisfiability problem; NP Complete Problems; SAT solver algorithm; SAT Theory Foundation

1. INTRODUCTION

The Boolean satisfiability problem (SAT) is a classical NP-complete problem that refers to determining whether there exists a set of variable assignments to a given Boolean expression that makes the expression true. The Cook-Levin theorem shows that any NP problem can be reduced to a SAT problem in polynomial time, proving the NP-completeness of the SAT problem[1]. The generality of the NP problem and the difficulty of solving it have inspired a great deal of research on NP-complete problems. Therefore, the study of SAT problems is not only important for understanding computational complexity, but also has far-reaching implications for practical applications, such as software verification, artificial intelligence, and circuit design.

At present, the research of SAT problem involves several aspects, including the deepening of the theoretical foundation, the improvement of the solution algorithm and the expansion of the application field. In terms of theoretical research, the theoretical foundation of SAT problems involves several fields such as mathematical logic and complexity theory. In recent years, researchers have conducted in-depth studies on the computational complexity and satisfiability determination of SAT problems. For example, for the computational complexity of SAT problems, researchers have explored the difficulty of solving them by proving NP completeness and designing more efficient algorithms. In terms of the improvement of the solution algorithm, the researchers conducted a systematic study on the key technologies of coding, preprocessing and solution algorithms for SAT problems. They proposed a series of improvement strategies and new methods to further enhance the success rate and performance of SAT problem solving by optimizing the preprocessing process and improving the efficiency of the solving algorithm. In terms of application area expansion, with the continuous improvement of the solution method and the deepening of the theoretical foundation, the application area of SAT problems is also expanding. At present, SAT problems have been applied in many fields such as circuit design, program verification, plan scheduling, bioinformatics and so on. For example, in circuit design, SAT problem solving methods can be used to automatically verify the correctness of circuits; in program verification, SAT problem solving methods can be used to detect errors and loopholes in the program; in scheduling, SAT problem solving methods can be used to optimize task allocation and resource scheduling. In the future, with the continuous progress of science and technology and the growing application demand, the application areas of SAT problems will continue to expand.

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.

2. CONCEPTS RELATED TO SAT QUESTIONS

The SAT problem is a classical computer science problem, usually described and modeled in the form of Boolean logic, where the goal is to determine whether there exists a set of Boolean variable assignments that satisfy the conditions for a given Boolean formula to be true.

Boolean Formulas: Boolean formulas are logical expressions consisting of Boolean variables, logical operators, and parentheses that describe a set of logical relationships. In SAT problems, the Collocation Paradigm (CNF) is often used as the standard form of Boolean formulas.

For example, P denotes the CNF formula for a SAT instance.

$$P = (x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee x_3) \wedge (\neg x_3 \vee \neg x_4) \tag{1}$$

Boolean Variables: Boolean variables are variables in logical expressions that can only take two values True or False. There are 4 Boolean variables in P, $I = \{x_1, x_2, x_3, x_4\}$.

Literal: A literal is a Boolean variable or its negative form used to form a clause. A literal is called a positive literal if it is the variable itself, and a negative literal if it is the negative form of the variable.

Clause: The clause is the basic unit in the CNF form and represents the parsing of multiple Boolean variables. Each clause may contain one or more literals, and each literal may be a Boolean variable or its negation. There are 3 clauses $C_1 = (x_1 \vee \neg x_2 \vee x_4)$,$C_2 = (x_2 \vee x_3)$,$C_3 = (\neg x_3 \vee \neg x_4)$ in P .

Satisfiability: a SAT problem is a problem of determining whether or not there exists an assignment to a given Boolean formula that satisfies the condition. If there exists a set of variable assignments such that P is true, the problem is said to be satisfiable, otherwise it is called unsatisfiable.

These basic concepts form the basis for understanding and studying SAT problems, and are important for conducting theoretical research and solving practice of SAT problems.

3. OVERVIEW OF RESEARCH ON SAT PROBLEMS

3.1. Improvements in SAT problem solving methods

In terms of solution algorithms, complete and incomplete algorithms are the two main categories of methods. Completeness algorithms use the ideas of exhaustion and backtracking, which theoretically guarantee the satisfiability of a given propositional formula and give a completeness proof in case the instance is unsolved. However, for large-scale SAT problems, deterministic algorithms may not be applicable. In contrast, incomplete algorithms are based on the idea of local search, and although most stochastic search algorithms cannot determine the unsatisfiability of a SAT problem, they can often obtain a solution faster than deterministic algorithms when dealing with satisfiable large-scale stochastic classes of problems, due to the use of heuristic strategies to guide the search.

3.1.1. Traditional SAT Solution Methods

Completion algorithms are mainly based on the idea of backtracking search.In the late 1970s and early 1980s, the Davis-Putnam-Logemann-Loveland (DPLL) algorithm was proposed, marking the modern phase of SAT solving[2]. This algorithm, based on backtracking search and pruning techniques, provides a powerful framework for solving SAT problems. As the size of SAT problem increases, the traditional DPLL algorithm is inefficient in some cases. In order to improve the solution efficiency, researchers have proposed many heuristic search strategies to accelerate the solution process by improving the variable decision making, conflict analysis and clause learning, learning clause deletion strategy and restarting strategy, respectively. Among them, the conflict-driven clause learning (CDCL) algorithm is now the most widely used SAT solving algorithm[3]. The CDCL algorithm drove the subsequent development where improvements were made for different parts of the solver such as branching decision heuristics, clause learning, learning clause deletion strategies, restarting, etc., which improved the performance of the solver, and a series of more efficient SAT solvers were proposed, such as Chaff, MiniSat, Glucose, BerkMin, CryptoMiniSAT, PicoSAT, and MapleLCMDist.

Incomplete algorithms usually employ heuristic strategies to guide the search process. Stochastic Local Search (SLS) algorithms are the basis of most incomplete algorithms[4]. This class of algorithms performs a randomized search in the space of assignments of variables by means of a certain strategy. It first generates a set of random assignments to all the variables, and then flips the assignments to some of the variables under the guidance of the objective function, gradually approaching the final solution. Selman, Levesque, and Mitchell proposed the greedy local search algorithm GSAT[5]. after randomly generating the initial truth assignments, the algorithm chooses at each step of the search the global variant that lets the number of unsatisfiable clauses drop the most to be flipped until the solution of the problem is found. Although the solution is fast, it is prone to local minima and search loops. In 1994, scholars Selman et al. improved the framework of the GSAT algorithm and proposed the WalkSAT algorithm focusing on random walks[6]. The algorithm picks variants from unsatisfiable clauses with the goal of minimizing the number of clauses that change from true to false. This approach of picking variants from unsatisfiable clauses greatly reduces the computational cost of local search and introduces a perturbation strategy to efficiently solve SAT problems with tens of thousands of variants. Most of the subsequent incomplete algorithms are derived from these two algorithms.In 2010, Balint and Fröhlich proposed the Sparrow algorithm, which introduces a probability distribution function in variable selection, where each variable is selected with a certain probability[7].Most of these solvers use relatively complex decision heuristics.In 2011, Shao-Wei Tsai scholars proposed the Configuration Checking strategy, i.e., checking the neighbor information of a variable before flipping it, which was applied to the WalkSAT framework to obtain the SWcc algorithm[8]. In 2012, Balint and Schöning proposed probSAT based on simple probability distributions, which analyzes different functions for the probability distributions and selects the next flip variable based on the performance of the solver, which provides a significant performance improvement over the previous solvers[9].In 2016, Liu et al. utilized the concept of key variables to assign probabilities to the flip variables in the SLS process, and introduced the "separation non-caching" technique to improve the efficiency of the SAT solver[10]. Subsequently, YalSAT implemented multiple variants and extensions of the probSAT algorithm and won the gold medal in the 2017 SAT competition.In 2019, in order to avoid the local search process from stalling due to generating the same solution again, Hossen et al. introduced similarity detection to jump out of the stagnant state of local search in the CCAnrSim solver[11].

3.1.2. SAT solving methods based on machine learning

Classical algorithms for determining satisfiability problems usually have a defined algorithmic flow, amount of data to process, and form. Deep neural networks represent the target in multiple layers by constructing a multilayer network in the hope of obtaining better feature robustness by representing the abstract semantics of the data through multiple layers of high-level features.

Initially machine learning based SAT solving algorithms usually extracted various features of CNF formulas manually, such as clause lengths, frequency of occurrence of variables, relationships between clauses, etc., and then learned and predicted them using traditional machine learning algorithms such as decision trees, support vector machines, etc[12]. Devlin et al. viewed an instance of a SAT problem as a vector with a number of features, each representing some attribute or property of the SAT problem, and then transformed the solution of the SAT problem into a binary classification task on the feature vector[13].Danisovszky et al. constructed a 48-wit lexicon to improve the classification performance of a machine learning model by selecting appropriate features[14]. Between learning better clauses and the computational overhead of reconstructing the search tree, Liang et al. introduced a new machine learning-based restart strategy that predicts the quality of the next learned clause based on the history of previously learned clauses[15]. However, many of these machine learning solutions rely on well-designed features. Considering that SAT problem instances are rich in structure, which tends to be lost by most classical feature extraction techniques, much of the recent work has been based on neural networks to extract structural features of SAT problems.

CNNSAT: For the random 3-SAT problem, state-of-the-art solvers still do not scale well to formulas with hundreds of variables. To solve this problem, some researchers have proposed to consider the determination of the satisfiability problem as a classification problem and introduced the CNNSAT framework, which is based on a convolutional neural network that encodes the clauses and variables in the CNF formulas into vectors and constructs an $m \times n$ sparse matrix based on whether the literal appears in the clauses or not, where m is the number of clauses and n is the number of variables, aiming to accelerate and improve the accuracy of statistical decision making in SAT[16].

Graph-Q-SAT: Considering the solution process of the SAT problem as a reinforcement learning problem, deep reinforcement learning algorithms are used to learn how to make variable assignments and decisions to maximize the performance of the solver, Kurin et al. proposed a new model based on the Graph-Q-SAT model, which employs value-based reinforcement learning and the use of graphical neural networks as an approximation of the Q-function to construct a new branching heuristic algorithm for SAT[17]. A framework for solving combinatorial optimization problems using graph neural networks and reinforcement learning is applied to NP puzzles such as the Traveling Salesman Problem and the Backpack Problem and demonstrates performance improvements over traditional algorithms[18]. In addition, Fei et al. proposed another approach to solve the SAT problem by modeling the symbolic reasoning problem directly as a game and using deep reinforcement learning to leverage the decision-making capabilities of the neural network[19].

NeuroSAT: CNF formulas are represented as graph structures and then graph neural networks are used to learn and predict the graph structures. This approach better captures structural information and dependencies in CNF formulas.Selsam et al. used an undirected graph representation of the CNF in the NeuroSAT model and constructed the model with two vectors, three multilayer perceptrons, and a two-layer paradigm LSTM[20]. However, the model requires the generation of a specific type of pair of instances to model the SAT problem, with the difference between the two instances being only the negation of a single literal in a single clause. As a result, the training data is limited by this requirement, and NeuroSAT cannot accurately predict satisfiability when the number of variables is large. Later, Selsam et al. improved NeuroSAT by proposing Neurocore, which uses unsaturated core predictions to guide the CDCL solver, and the CDCL solver with Neurocore solves 6%-11% more instances than the original solver[21].

NLocalSAT: 2020, NLocalSAT combines the SLS process with a solution prediction model that optimizes an SLS-based SAT solver by changing initialization assignments via a neural network. Specifically, NLocalSAT uses a solution prediction model that outputs a probability distribution for the next variable assignment by inputting the current set of conflicting clauses and variable assignments[22]. This probability distribution is used to guide the initialization of the LS algorithm. During the solution process, NLocalSAT continuously uses the neural network model to adjust the initial assignments of the variables in the expectation of finding the solution faster. By combining the LS process with the solution prediction model, NLocalSAT is able to converge to the solution faster in most cases, thus improving the performance of the SAT solver.

GAT-SAT: GAT-SAT is a SAT solver based on Graph Attention Network (GAT). The method utilizes graph neural networks to represent CNF formulas and solves SAT problems by learning the relationships between the variables in the formulas[23].The core idea of GAT-SAT is to represent the CNF formulas as a graph, where the variables and clauses are represented as the nodes and edges in the graph, and then utilize the graph attention network to learn the interactions between the nodes, so as to predict the values of the variables. Compared with traditional SAT solving methods, GAT-SAT is able to better capture the dependencies between variables and has higher solving efficiency and generalization ability. The method performs well in dealing with large-scale and complex SAT problems and brings new ideas and methods to the field of SAT solving.

Shi et al. made the first attempt to solve the MaxSAT problem using Transformer. They represented a given SAT instance as a two-part graph and used Transformer to aggregate messages at these nodes.SATformer applies GNN to obtain clause feature representation in CNF[24]. A hierarchical Transformer architecture is applied to the clause feature representation to capture the relationship between clauses, give higher weight to the clause feature representation in the Minimum Unsatisfied Core (MUC), and learn the correlation between clauses efficiently. Based on Transformer, HGT defines a meta-path based word self-attention mechanism and a two-part graph link based word-subclause cross-attention mechanism for solving the CSP problem[25].

3.2. Deepening of the theoretical foundations of the SAT problem

The theoretical basis of SAT problems involves a variety of fields such as mathematical logic and complexity theory. In recent years, researchers have conducted in-depth studies on the computational complexity and satisfiability determination of SAT problems. For example, for the computational complexity of SAT problems, researchers have explored the difficulty of solving them by proving NP completeness and designing more efficient algorithms. Meanwhile, important progress has also been made in the research on satisfiability determination, including proposing new determination methods and improving the efficiency of the determination process. The results of these theoretical researches not only provide a more solid theoretical foundation for the solution of SAT problems, but also provide a reference for the researches in other related fields.

Complexity theory: The SAT problem is shown to be NP-complete, i.e., a solution can be verified to be correct in nondeterministic polynomial time. This result suggests that if the SAT problem can be solved in polynomial time, then all problems in the NP class can be solved in polynomial time. Therefore, the study of SAT problems is important for understanding NP-completeness and computational complexity theory[26].

Difficulty analysis: Difficulty analysis of SAT problems is one of the most important aspects of SAT theoretical research. Researchers are concerned with determining the difficult nature of SAT problems, such as Phase Transition Phenomena, Threshold Phenomena, etc., and the average complexity of SAT problems. These analyses contribute to a deeper understanding of the nature and difficulty of solving SAT problems[27].

Fixed-parameter complexity: Fixed-parameter complexity theory is an approach to study the complexity of a problem under specified parameters.The study of fixed-parameter complexity of SAT problems focuses on exploring the solvability of SAT problems under different parameters, such as the number of variables, the length of clauses, and other parameters, so as to reveal the structural characteristics and the difficulty of solving SAT problems[28].

Model Checking and Verification: SAT problems have a wide range of applications in formal methods and software engineering, such as model checking and program verification, etc. The theoretical study of SAT problems involves the improvement and optimization of algorithms for model checking and verification, as well as the application and diffusion of SAT solving techniques in practical engineering[29, 30].

Randomization algorithm: Randomization algorithms are one of the most important methods for solving SAT problems. Theoretical research focuses on the analysis and design of efficient stochasticity algorithms, such as randomized local search and random walk algorithms, as well as the performance analysis and theoretical guarantees of these algorithms.

Structural analysis: Structural analysis of SAT problems is the study of structural features of CNF formulas, with the aim of discovering and exploiting particular structures in CNF formulas to improve solution efficiency. This includes the study of aspects such as clauses, variables, and dependencies between clauses in SAT problems, as well as algorithm design and complexity analysis of these structural features.

Automated reasoning and proof: Automated reasoning and proofing of SAT problems is another important direction of research in SAT theory. This includes research on proof systems, automated reasoning algorithms, and proof search techniques for SAT problems, as well as applications and extensions of these techniques to areas such as logical reasoning, formal verification, and artificial intelligence.

Theoretical studies of SAT problems cover a wide range of aspects, from complexity theory to stochasticity algorithms, as well as in-depth analyses of problem structure and solution methods, which provide an important theoretical foundation for understanding the nature of SAT problems, improving their efficiency, and applying them to real-world problems

3.3. Expansion of the field of application of SAT problems

The solution methods of SAT problems have a wide range of applications in several fields. With the continuous improvement of the solution methods and the deepening of the theoretical foundation, the application areas of SAT problems are also expanding.

In electronic design automation (EDA), by transforming a circuit design problem into a SAT problem, a SAT solver can be used to quickly find a solution that satisfies the design constraints.SAT problems can be used for logic optimization and synthesis in logic synthesis to help design more efficient and compact circuits. In the wiring phase of circuit design, SAT problems can be used to solve wiring planning and wiring optimization problems to minimize signal delays, power consumption, etc. SAT problems can be used to automatically generate test patterns to help detect faults and errors in circuits.SAT problems can be used in troubleshooting and fault-tolerant design to help detect and repair errors and faults in circuits. Overall, the application of SAT problems in EDA covers all stages of circuit design and can increase design efficiency, reduce design costs, and improve circuit performance and reliability[31, 32].

In the field of software engineering, SAT problems play an important role in software testing, model checking, program analysis, etc. SAT problems can be used to automatically generate test cases to help testers find errors and defects in the software.SAT problems can be used in program analysis and verification, including model checking, static analysis of code, etc., to ensure the correctness and safety of the software. In software optimization and synthesis, SAT problems can be used to solve problems such as logic optimization and code synthesis to improve software execution efficiency and resource utilization. In software-hardware co-design, SAT problems can be used to solve problems such as software-hardware interface matching and communication protocol verification to ensure the correctness and consistency of software-hardware systems[33].

In the field of Artificial Intelligence. the SAT problems are used to solve Boolean logical reasoning problems such as logical inference, logical inference, propositional logic, etc. These problems are very important in the process of knowledge representation and reasoning in artificial intelligence. Automated planning is an important area in artificial intelligence that involves finding an optimal solution given constraints.SAT problems can be used in automated planning to help solve a variety of planning and scheduling problems.SAT problems can also be used as the basis of logic programming languages for implementing a variety of intelligent systems and expert systems. By formulating problems as logical formulas and solving these formulas using SAT solvers, a variety of logical reasoning and inference tasks can be implemented.In the field of bioinformatics, SAT problems are used in bioinformatics for genome sequence comparison, protein folding prediction, and molecular structure design. By modeling a biological problem as a SAT problem, a solution can be obtained quickly using a SAT solver[34, 35].

In the field of network security, SAT problems are used for tasks such as cryptanalysis, access control policy verification, intrusion detection, etc. For example, it is used for password cracking, analyzing the security of network protocols, detecting anomalous behaviors in networks, etc. SAT problems are used in automated planning to generate constraint-compliant action sequences. For example, it is used for path planning of intelligent bodies, motion control of robots, scheduling of production lines, etc[36].

Besides, SAT problems have a wide range of applications in combinatorial optimization, such as the traveler's problem, the knapsack problem, and the graph coloring problem. By transforming combinatorial optimization problems into SAT problems, SAT solvers can be used to find optimal or near-optimal solutions[37].

The application areas of SAT problems continue to expand and deepen, and with the development of technology and the increase in the complexity of the problems, SAT solving techniques will continue to play an important role and bring new applications and breakthroughs in more fields.

4. SUMMARY

In recent years, researchers have proposed numerous new solution methods for SAT problems, including backtracking-based search algorithms, machine learning-based solvers, parallel and distributed solving techniques. These methods have made significant progress in terms of solving efficiency, scalability and generalization. Among them, backtracking-based search algorithms find explanations that satisfy Boolean formulas by constructing search trees. These algorithms combine the ideas of local search and global optimization to find a solution or prove the absence of a solution in a shorter time. Machine learning-based solvers, on the other hand, utilize machine learning techniques to improve the performance of the solver by training models to predict the decisions during the search process, thus improving the solution efficiency. In addition, parallel and distributed solving techniques have become an important direction in the study of SAT problems as computational resources continue to improve. These techniques achieve parallel processing of SAT problems by utilizing computational resources such as multi-core processors and cloud computing, thus further improving the solving efficiency. Deep learning methods have also been widely used in the field of SAT solving. Researchers have proposed some neural network-based SAT solvers that utilize deep learning techniques to improve the performance and robustness of the solvers. In addition to the traditional SAT solving problems, researchers have also applied SAT problems to a wider range of fields, such as bioinformatics, cybersecurity, and artificial intelligence. These new application areas provide new challenges and opportunities for the research of SAT problems.

Although the SAT solver has made great progress, there are still some difficulties and challenges, such as dealing with large-scale problems and improving the robustness of the solver. Therefore, further improvement of the solving algorithms is still an important research direction. The application of deep learning methods in the field of SAT solving is still in its infancy, and the combination of deep learning and traditional SAT solving techniques can be further explored in the future to improve the performance and efficiency of the solver. With the application of SAT problems in more fields, the potential of SAT problems in bioinformatics, cybersecurity, artificial intelligence and other fields can be further explored in the future, and corresponding solving methods and tools can be developed.

In conclusion, the SAT problem, as a classical combinatorial optimization problem, still has a broad research prospect and application prospect in the future. Through continuous research and innovation, we are expected to further improve the performance and application scope of the SAT solver and provide more effective tools and methods 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] 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.

[4] Ruiz R, Pranzo M. Stochastic Local Search: Foundations and Applications,Holger H. Hoos, Thomas Stützle, Morgan Kaufmann Publishers[J]. European Journal of Operational Research, 2006,172(2):716-718.

[5] Yung W H, Seung Y W, Lee K H, et al. A Runtime Reconfigurable Implementation of the GSAT Algorithm[J]. Springer-Verlag, 1999.

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

[7] Balint A, Fröhlich A. Improving Stochastic Local Search for SAT with a New Probability Distribution: Theory and Applications of Satisfiability Testing – SAT 2010[C], 2010.

[8] S. C, K. S. Local Search with Configuration Checking for SAT: IEEE 23rd International Conference on Tools with Artificial Intelligence[C], 2011.

[9] A B, U S. Choosing probability distributions for stochastic local search and the role of make versus break: Proceedings of the 15th international conference on Theory and Applications of Satisfiability Testing[C], 2012.

[10] Liu S, Papakonstantinou P. Local Search for Hard SAT Formulas: The Strength of the Polynomial Law: Proceedings of the AAAI Conference on Artificial Intelligence[C], 2016.

[11] Hossen M S, Polash M M A. Implementing an Efficient SAT Solver for Structured Instances: 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)[C], Spokane, WA, USA, 2019.

[12] 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.

[13] Devlin D, O Sullivan B. Satisfiability as a Classification Problem.: In Proc. of the 19th Irish Conf. on Artificial Intelligence andCognitive Science[C], 2008.

[14] Danisovszky M Z G Y. Classification of SAT Problem Instances by Machine Learning Methods: International Conference on Applied Informatics[C], 2020.

[15] Liang J H, Oh C, Mathew M, et al. Machine Learning-Based Restart Policy for CDCL SAT Solvers: Theory and Applications of Satisfiability Testing – SAT 2018[C], 2018.

[16] Wang Y, Cn Y S N E, Fengjuan G A A L. CNNSAT: Fast, Accurate Boolean Satisfiability using Convolutional Neural Networks: ICLR[C], 2019.

[17] 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.

[18] Mazyavkina N, Sviridov S, Ivanov S, et al. Reinforcement learning for combinatorial optimization: A survey[J]. Computers & Operations Research, 2021,134:105400.

[19] Wang F, Rompf T. From Gameplay to Symbolic Reasoning: Learning SAT Solver Heuristics in the Style of Alpha(Go) Zero[J]. ArXiv, 2018.

[20] Selsam D, Lamm M, Bünz B, et al. Learning a SAT Solver from Single-Bit Supervision: International Conference on Learning Representations[C], 2018.

[21] Selsam D, Bjørner N. Guiding High-Performance SAT Solvers with Unsat-Core Predictions: Theory and Applications of Satisfiability Testing – SAT 2019[C], 2019.

[22] Zhang W, Sun Z, Zhu Q, et al. NLocalSAT: Boosting Local Search with Solution Prediction: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence[C], 2020.

[23] Chang W, Zhang H, Luo J. Predicting Propositional Satisfiability Based on Graph Attention Networks[J]. International Journal of Computational Intelligence Systems, 2022,15(1):84.

[24] Shi Z, Li M, Khan S, et al. SATformer: Transformers for SAT Solving[J]. arXiv preprint, 2022.

[25] Shi F, LI C, Bian S, et al. Transformers satisfy: ICLR[C], 2021.

[26] Arora, Sanjeev, Barak B. Computational Complexity: A Modern Approach[M]. Cambridge University Press40 W. 20 St. New York, NYUnited States, 2009.

[27] 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.

[28] Downey R, Flum J, Grohe M, et al. Bounded fixed-parameter tractability and reducibility[J]. Annals of Pure and Applied Logic, 2007,148(1):1-19.

[29] Biere A, Cimatti A, Clarke E M, et al. Symbolic model checking using SAT procedures instead of BDDs: Proceedings 1999 Design Automation Conference[C], New Orleans, LA, USA, 1999.

[30] Biere A, Cimatti A, Clarke E, et al. Symbolic Model Checking without BDDs: Tools and Algorithms for the Construction and Analysis of Systems[C], 1999.

[31] Larrabee T. Test pattern generation using Boolean satisfiability[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1992,11(1):4-15.

[32] Chen P, Keutzer K. Towards true crosstalk noise analysis: 1999 IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers [C], San Jose, CA, USA, 1999.

[33] Jackson D, Schechter I, Shlyakhter I. Alcoa: the Alloy constraint analyzer: Proceedings of the 2000 International Conference on Software Engineering[C], Limerick, Ireland, 2000.

[34] Muise C, McIlraith S A, Beck J C, et al. Dsharp: Fast d-DNNF Compilation with sharpSAT: Advances in Artificial Intelligence[C], 2012.

[35] Rintanen J. Planning as satisfiability: Heuristics[J]. Artificial Intelligence, 2012,193:45-86.

[36] Rintanen J, Heljanko K, Niemelä I. Planning as satisfiability: parallel plans and algorithms for plan search[J]. Artificial Intelligence, 2006,170(12):1031-1080.

[37] Appel K, Haken W. The solution of the four-color-map problem[J]. Scientific African, 2010,4(237):108-121.