Academic Journal of Applied Sciences
ISSN: 3105-0549 (Print), ISSN: 3105-0557 (Online) | Vol. 1, No. 3, 2026

A GRU-Based SAT Phase Prediction Algorithm

Hanchang Wang *

Henan Polytechnic University, Jiaozuo 454000, China

Abstract: In the field of computer science, the Boolean Satisfiability Problem (SAT) was the first problem proven to be NP-complete and has been widely applied in artificial intelligence, formal verification, and many other research domains. Most modern SAT solvers are built upon the Conflict-Driven Clause Learning (CDCL) framework, whose performance heavily relies on heuristic mechanisms such as variable branching decisions and phase selection strategies. Although graph neural network-based heuristic guidance has become a popular research direction, existing approaches still suffer from significant limitations when handling large-scale industrial instances, particularly in capturing long-range variable dependencies and achieving sufficient reasoning depth. To address these challenges, this paper proposes NeuroLogic-GSM, a SAT phase prediction model based on Gated Recurrent Units (GRUs). The proposed model reconstructs the traditional static convolution-stacking paradigm into a recurrent iterative reasoning architecture, where weight-shared logical processing units are employed to simulate unit propagation and conflict analysis mechanisms, while GRUs dynamically track the evolution of variable states to effectively capture long-range logical dependencies. The introduction of recurrent reasoning units not only enhances the logical reasoning depth of the model, but also improves parameter utilization efficiency and training stability. Experimental results on the main tracks of the SAT Competitions from 2023 to 2025 demonstrate that NeuroLogic-GSM achieves solver performance improvements of 2.5%, 1.77%, and 1.48%, respectively, compared with NeuroBack, validating its effectiveness in handling complex logical reasoning tasks.

Keywords: Boolean Satisfiability Problem, Gated Recurrent Unit, Phase Prediction, Backbone Variables.

1. Introduction

The Boolean Satisfiability Problem (SAT) [1] is one of the most classical combinatorial optimization problems in computer science and plays a fundamental role in computational complexity theory. Since Cook proved in 1971 that SAT is an NP-complete problem, it has become a crucial benchmark for measuring computational complexity, implying that any NP problem can be reduced to SAT in polynomial time. Therefore, SAT is not merely a fundamental logical solving problem, but also serves as a unified representation framework for a wide range of complex computational tasks. Research on efficient SAT solving methods is of great theoretical and practical significance for addressing many classical NP-hard problems, such as the Traveling Salesman Problem, Graph Coloring Problem, and Knapsack Problem.

Although modern SAT solving techniques have made significant progress, especially with solvers based on the Conflict-Driven Clause Learning (CDCL) algorithm that can handle industrial instances with millions of variables, traditional methods still face severe challenges when dealing with increasingly complex modern problems. CDCL algorithms mainly rely on heuristic strategies, which are often based on local statistical features such as variable activity. As a result, they struggle to capture complex global topological dependencies among variables. When solvers encounter highly structured instances, this limitation can easily lead them into inefficient search regions, resulting in combinatorial explosion.

As problem scales grow exponentially, traditional solvers suffer from substantial memory consumption and computational overhead when handling large-scale instances. Moreover, due to the inherently sequential nature of logical reasoning, it is difficult to effectively achieve parallel acceleration on modern multi-core hardware. These limitations indicate that purely manually designed heuristic rules are gradually reaching their performance ceiling, highlighting the urgent need for new computational paradigms to break this bottleneck.

In recent years, deep learning (DL) [2] techniques represented by Graph Neural Networks (GNNs) [3] have demonstrated strong capabilities in feature extraction and pattern recognition, providing new opportunities for SAT solving. Since CNF formulas can naturally be modeled as graph structures, using GNNs to learn complex topological relationships between variables and clauses, thereby assisting solvers in more accurate phase prediction and variable branching, has become a major research focus in the academic community.

2. Background and Related Work

2.1. Boolean Formula

The essence of the Boolean Satisfiability Problem (SAT) is to determine whether there exists an assignment of Boolean variables that makes a given Boolean logic formula evaluate to true. In practical applications, SAT problems are commonly transformed into Conjunctive Normal Form (CNF) for easier processing. A CNF formula consists of multiple clauses, where each clause is a disjunction (logical OR) of one or more literals, and the entire formula is a conjunction (logical AND) of these clauses. A literal represents either a Boolean variable or its negation.

$$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_4)$$

As shown in the above formula, it consists of three clauses, where literals within each clause are connected by logical OR operations, while the clauses are connected by logical AND operations. The SAT solving task is to determine whether there exists a truth assignment for variables x_1 x_2 x_3 x_4 that makes the entire CNF formula true. For example, when the

assignment is x_1=1 x_2=1 x_3=1 x_4=0, the formula evaluates to true.

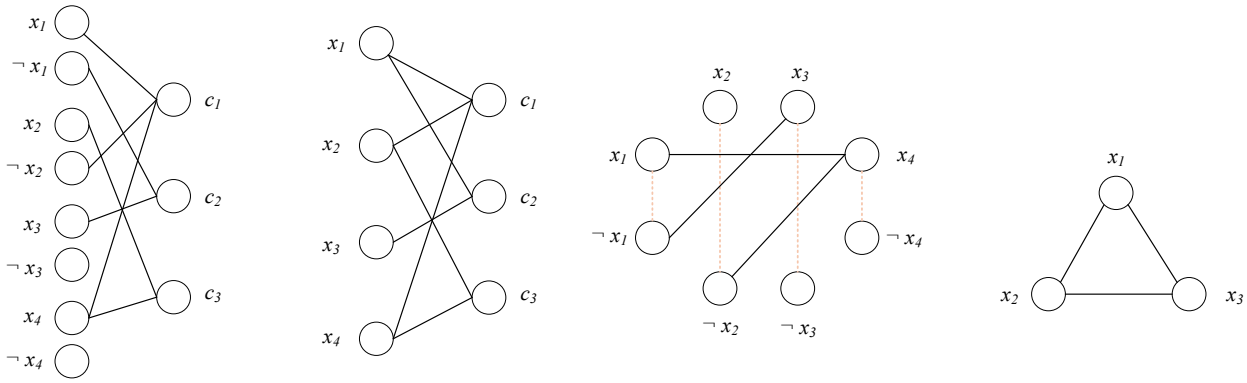
Furthermore, a CNF formula can also be transformed into a graph structure, enabling structured modeling and reasoning using deep learning methods such as Graph Neural Networks. This provides a new intelligent paradigm for SAT solving, which will be discussed in detail in Section 2.2.

2.2. Graph representation of Boolean formulas

To apply Graph Neural Networks (GNNs) to SAT problems, the first step is to encode CNF formulas into graph structures. The structural properties of SAT instances have been extensively studied from a graph-theoretic perspective, and any CNF formula can naturally be represented as a bipartite variable–clause graph, also referred to as a SAT factor graph.

In practice, CNF formulas can be represented using four common graph encodings, as illustrated in Figure 1: (1) Literal–Clause Graph (LCG), (2) Literal Interaction Graph (LIG), (3) Variable–Clause Graph (VCG), and (4) Variable Interaction Graph (VIG). The LCG is a bipartite graph in which one set of nodes corresponds to literals and the other corresponds to clauses, with edges connecting each literal to the clauses in which it appears. The LIG, on the other hand, consists solely of literal nodes, where an edge is added between two literals if they co-occur within the same clause. The VCG and VIG are derived by merging the positive and negative literals of each variable into a single node representation.

These four graph representations form a hierarchy of decreasing structural complexity, reflecting progressively stronger levels of information compression. The LCG preserves complete structural information, allowing the original CNF formula to be fully reconstructed, whereas the VIG retains only coarse-grained relationships and loses most detailed structural information. As a result, LCG and LIG are more commonly adopted in practical SAT learning and modeling frameworks due to their richer representational capacity.



(a) LGC (b) VCG (c) LIG (d) VIG

Figure 1. SAT instance graph

2.3. Graph Representation of Boolean Formulas

The CDCL [4] algorithm consists of several key steps, including Boolean propagation (Boolean Propagation, BP), decision making, conflict detection, clause learning, and backtracking. Whenever a conflict is encountered, the solver learns a new conflict clause to prevent the same type of error from reoccurring in the future. Through continuous learning, the CDCL algorithm performs pruning-based exploration in the search space, significantly reducing the solving path.

However, the performance of CDCL solvers heavily depends on the coordinated optimization of several internal heuristic strategies. The most widely used heuristics include variable branching heuristics, phase heuristics, clause management heuristics, and restart heuristics.

(1) Variable Branching Heuristic

Variable branching [6] refers to the process of selecting an unassigned variable as the next decision variable during solving. This choice has a significant impact on the speed of subsequent Boolean propagation and the likelihood of quickly encountering conflicts. The most commonly used strategy is the VSIDS heuristic, where each variable maintains an activity score representing its frequency of involvement in conflicts. After each conflict, variables appearing in the conflict clause have their scores increased, while all scores gradually decay over time. The solver prioritizes variables with higher activity scores for branching. However, VSIDS relies mainly on local conflict statistics and cannot capture global structural properties (e.g., backbone variables), making it prone to inefficient repetitive search patterns.

(2) Phase Heuristic

Phase heuristics [7] determine the initial Boolean assignment (True or False) of a selected decision variable. This assignment influences the direction of Boolean propagation and thus affects both the timing and structure of conflicts. For instance, assigning a variable as True may immediately satisfy multiple unit clauses and accelerate the search, while assigning it as False may trigger early conflicts, which can also be beneficial for pruning.

Backbone variables are defined as variables that take the same Boolean value across all satisfying assignments. In other words, if a variable xi is always True or always False in all solutions, it is considered a backbone variable. This concept reflects the structural properties of the solution space.

In the SAT solving framework, phase heuristics and backbone variables exhibit a natural synergy. Since backbone variables have a unique assignment in all satisfying solutions, phase selection essentially serves as a key mechanism for identifying these "correct decision paths." If phase prediction can accurately capture the polarity of backbone variables and initialize them correctly, the solver may directly follow a satisfiable branch, significantly reducing search depth and triggering stronger Boolean propagation effects. Conversely, incorrect assignments to backbone variables inevitably lead to deep clause conflicts and increased backtracking. Therefore, incorporating backbone-aware phase prediction into phase heuristics can accelerate solving by exploiting their

unique-value property. This approach, which maps global solution-space structure into local assignment guidance, provides an effective pathway for overcoming CDCL limitations and achieving efficient pruning in complex SAT instances.

2.4. Gated Recurrent Unit

Gated Recurrent Unit (GRU) [5], an important variant of Recurrent Neural Networks (RNNs), achieves a good balance between model simplicity and the ability to capture long-range dependencies in sequential data. In SAT problems, the sequence of variable assignment decisions often exhibits strong temporal dependencies, where early search decisions directly influence subsequent pruning of the search space and the overall solving trajectory. Therefore, introducing GRU to model SAT decision sequences can effectively capture deep dependency patterns within the sequence.

The key components of GRU, namely the update gate and reset gate, provide a high degree of flexibility for dynamically adjusting the SAT solving state. By selectively retaining historical information and adaptively integrating current inputs, GRU significantly improves both heuristic search efficiency and prediction accuracy in SAT solvers. As an emerging cross-disciplinary approach, GRU-based SAT modeling offers a new theoretical perspective for optimizing solving paradigms in complex instances.

The core mechanism of the Gated Recurrent Unit (GRU) lies in the coordinated control of hidden state evolution through gating units. Its computational process includes the update gate z_t (as shown in Equation (1)), the reset gate r_t (Equation (2)), the candidate hidden state $h_t^{\sim}$ (Equation (3)), and the iterative update of the final hidden state h_t (Equation (4)). The detailed architecture is illustrated in Figure 2.

Here, W_z, W_r, W_h denote learnable weight matrices, while b_z, b_r, and b_h are the corresponding bias vectors. The operator σ represents the Sigmoid activation function, which maps values into the interval [0, 1] to enable gating control. The symbol $\odot$ denotes the Hadamard product, i.e., element-wise multiplication. The input at time step t consists of the current feature vector x_t and the previous hidden state h_{t-1}.

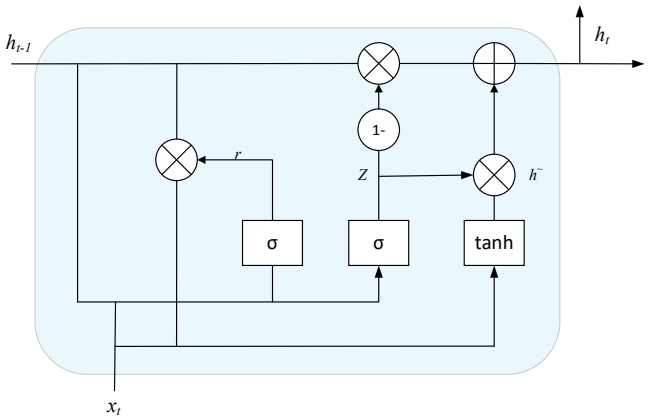


Figure 2. Architecture of the Gated Recurrent Unit (GRU)Network

$$z = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad (1)$$

$$r = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \quad (2)$$

$$h_t^{'} = tanh(W_h \cdot [r \odot h_{t-1}, x_t] + b_h) \quad (3)$$

$$h_t = (1 - z) \odot h_{t-1} + z \odot h_t^{'} \quad (4)$$

The output of the update gate zzz determines the proportion of historical information carried into the current state. Through this nonlinear gating mechanism, the network can adaptively preserve long-term memory or incorporate new input information, thereby exhibiting strong flexibility when processing decision sequences with long-range dependencies.

The reset gate r controls the extent to which the previous hidden state contributes to the candidate hidden state $h_t^{\sim}$ Finally, the current hidden state h_t is obtained by linearly interpolating between the historical state and the candidate state. This structure effectively alleviates the vanishing gradient problem in traditional recurrent neural networks and provides a robust mathematical foundation for modeling complex logical relationships in SAT problems.

3. Model Design Based on Gated Recurrent Units (GRU)

(1) Overall Model Framework

This section systematically describes the complete working mechanism of the NeuroLogic-GSM model, as illustrated in Figure 3. (1) Graph Construction: The input CNF formula is first transformed into a heterogeneous bipartite graph G containing multiple types of nodes. Meta-nodes and various types of logical edges are further introduced to fully capture the topological structure of Boolean constraints. (2) Encoding Stage: The raw attributes of nodes are mapped into a high-dimensional embedding space through an embedding layer. Layer normalization is then applied to stabilize feature distributions, producing initial latent logical states for subsequent iterative reasoning. (3) GRU-Based Recurrent Reasoning Engine: Within TTT predefined iterations, the reasoning module integrates two branches, RGIN and RGAT, to extract neighborhood structural features at each time step. These features are jointly fed into a Gated Recurrent Unit (GRU), where the update gate and reset gate dynamically control the forgetting of historical information and the integration of new information. (4) State Decoding and Phase Prediction: The final logical state after T iterations is passed into an MLP decoder. After nonlinear transformation, the model outputs a probability distribution over variable phases, completing the prediction process. Finally, variables are assigned initial phases in the SAT solver according to the predicted probabilities, enabling the solver to produce either a satisfying assignment or determine unsatisfiability.

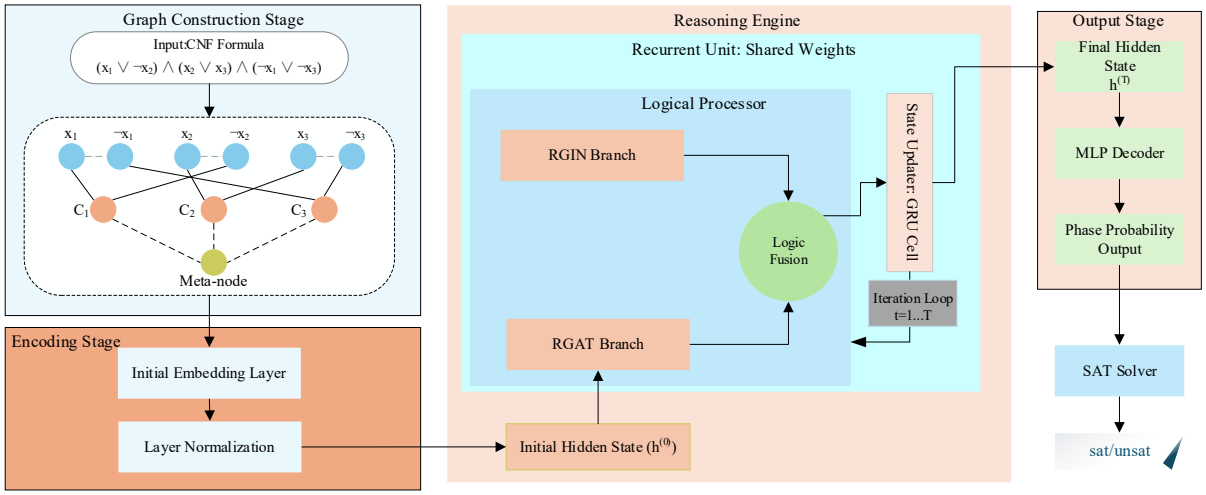


Figure 3. Overall framework of the NeuroLogic-GSM model

(2) Overall workflow of the model

The architecture of the NeuroLogic-GSM model is illustrated in Figure 4. Its core consists of two key components: a weight-shared logical processor and a GRU-based state update mechanism. The overall reasoning process can be divided into five stages.

a) RGAT Branch (Attention-Based Feature Extraction): The model computes attention coefficients α\alphaα to perform weighted aggregation of constraint information from neighboring clause nodes. This allows the model to capture structural dependencies with varying importance within the Boolean formula.

b) RGIN Branch (Feature Extraction without Attention Bias): A multilayer perceptron is used to apply nonlinear transformations to logical messages, followed by a summation operation for uniform aggregation. This ensures that the model preserves the original structural properties of homogeneous components in the Boolean formula.

c) Logic Fusion: The reasoning outputs from the RGAT and RGIN branches are integrated through a nonlinear fusion process. This not only removes redundant feature information but also enhances representational discriminability. The fused output produces the current iterative logical increment mt.

d) GRU-Gated State Evolution: The logical increment mt is fed into the GRU update module, where the reset gate, update gate, and state generation process jointly produce the final hidden state h(t) at the current time step.

e) Recurrent Feedback Loop for Deep Reasoning: The generated hidden state h(t) is not directly used for output; instead, it is fed back into the logical processor as the initial state for the next reasoning iteration. After T iterations, the model outputs the final logical representation for downstream decoding and prediction.

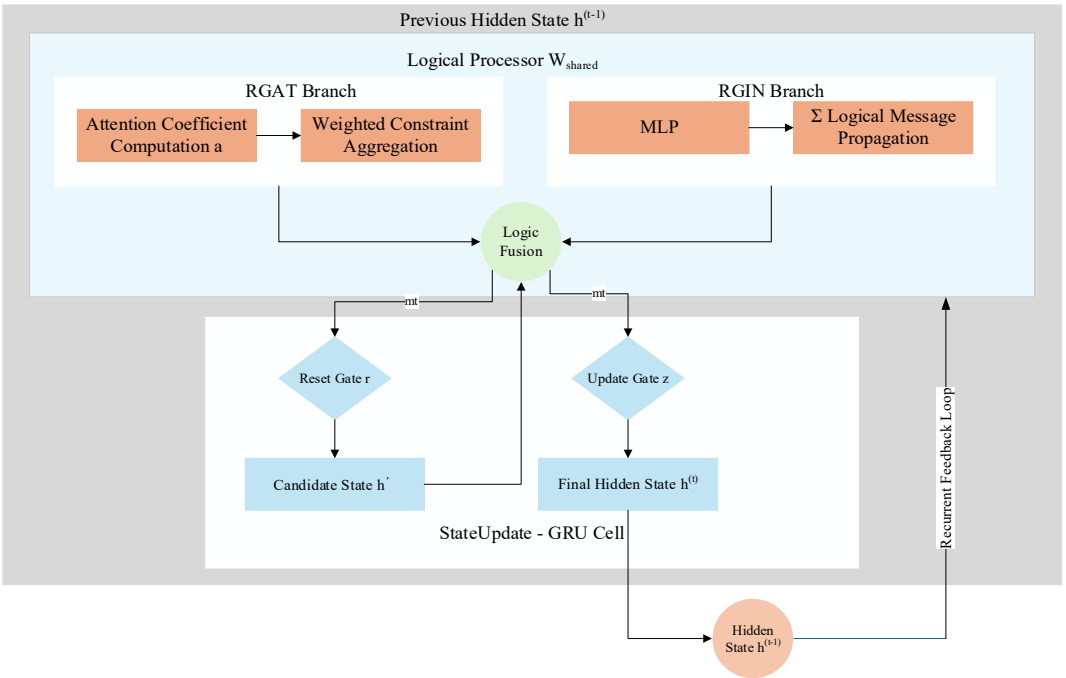


Figure 4. Network flow diagram of the NeuroLogic-GSM model

4. Experimental Design and Result Analysis

4.1. Experimental Dataset

Real-world SAT problems typically exhibit highly complex structural characteristics and are widely used to evaluate the performance of SAT solvers and related optimization algorithms in practical scenarios. Systematic evaluation across diverse SAT instances can more objectively reflect a model's generalization ability and robustness in industrial-grade problems. Based on this consideration, this study

constructs a comprehensive SAT dataset by following the DataBack dataset released by the NeuroBack team, which integrates theoretically generated instances, classical benchmark instances, and competition-level industrial instances.

CNFgen is used to generate structured theoretical SAT instances, covering five categories: random SAT, clique coloring, graph coloring, and counting principles. These provide structurally controllable training samples with well-defined distributions. The SATLIB benchmark library contains seven categories of classical SAT problems, including random SAT, graph coloring, planning, bounded model checking, Latin squares, circuit fault analysis, and full interval series problems. These represent medium-scale SAT instances with diverse structures. SAT Competition (SATCOMP) historical instances are collected from past SAT competition main tracks and include a large number of industrial SAT problems characterized by highly complex clause structures and variable dependencies. The Model Counting Competition (MCCOMP) dataset is further included to supplement large-scale, high-variable-density structural scenarios.

To avoid bias in satisfiability distribution, this study constructs dual versions for all CNF formulas, i.e., both SAT and UNSAT instances, ensuring an approximately balanced ratio of 1:1 between satisfiable and unsatisfiable problems in the dataset. The final dataset contains 63,400 SAT instances in total. The scale and statistics of each subset are summarized in Tables 1 and 2.

CNFgen generates 10,320 formulas with an average of 791 variables per instance. SATLIB contains 40,153 instances with relatively smaller variable sizes but diverse structural types. MCCOMP and SATCOMP include significantly larger-scale instances; in particular, SATCOMP has an average of more than 25,000 variables per instance, making it much closer to real industrial scenarios. Overall, the pretraining dataset contains 61,663 CNF formulas with an average of approximately 2,859 variables.

Regarding backbone variable distribution, significant differences are observed across datasets. CNFgen and SATLIB exhibit relatively higher proportions of backbone variables, while SATCOMP industrial instances show a much lower backbone ratio of only about 8%, further increasing the difficulty of backbone variable prediction tasks.

Table 1. Settings of the datasets used in the pretraining experiments

Pretrain-Data	CNFgen	SATLIB	MCCOMP	SATCOMP	Overall
#CNF	10320	40153	8780	2410	61663
#Var	791	114	16299	25310	2859
#Cla	285201	529	63501	89205	61905
#BackboneVar	284	58	8610	1960	1011

Table 2. Settings of the datasets used in the fine-tuning experiments

Finetune-Data	SATCOMP
#CNF	983
#Var	198700
#Cla	1354851
#Backbonevar	51662

4.2. Model Comparison Experiments and Result Analysis

The training and validation settings of NeuroLogic-GSM are consistent with those of NeuroMGCN and NeuroBack, where 85% of the Pretrain-Data and Finetune-Data are used for training and the remaining 15% is used for validation. This ensures a controlled experimental setting with a single variable, focusing on the impact of model architecture on SAT satisfiability solving performance. The model also reports two sets of results corresponding to pretraining and fine-tuning stages, with detailed metrics shown in Table 3. The evaluation criteria are consistent with previous models, providing a comprehensive assessment of classification performance.

In the pretraining stage, NeuroLogic-GSM already demonstrates superior generalization ability compared to NeuroBack and slightly better performance than NeuroMGCN. Its Precision reaches 0.928 (higher than NeuroBack's 0.892 and NeuroMGCN's 0.915), Recall is 0.821 (compared to 0.781 for NeuroBack and 0.808 for NeuroMGCN), and both F1-score and overall Accuracy reach 0.89 and 0.84 respectively, all exceeding the pretraining performance of the other two models. This indicates that the core component of NeuroLogic-GSM—the logic-aware relational modeling module—can more accurately capture logical dependency patterns in CNF formulas even without dataset-specific optimization, achieving a better balance between precision and coverage in positive class prediction. However, its MCC metric is slightly lower than that of NeuroBack during pretraining, which may be attributed to its more aggressive modeling of logical constraints, leading to a relatively conservative representation of global class correlations in the initial stage.

Table 3. Model Training Results

	Pre	Recall	F1	Acc	Specificity	MCC
NeuroBack	0.892	0.781	0.835	0.748	0.7267	0.6041
	0.938	0.912	0.924	0.887	0.8437	0.6825
NeuroLogic-GSM	0.928	0.821	0.89	0.84	0.7361	0.43
	0.97	0.941	0.963	0.912	0.853	0.692

After entering the fine-tuning stage, NeuroLogic-GSM demonstrates a particularly significant performance improvement. The core metric Precision increases from 0.928 to 0.970, Recall rises from 0.821 to 0.941, while the F1-score and overall Accuracy also improve to 0.963 and 0.912, respectively. In addition, both Specificity and MCC show

consistent improvements.

These results validate that the logic-relation enhancement module of NeuroLogic-GSM is better aligned with the distribution characteristics of the target data. With task-specific fine-tuning, the model can optimize its parameters more efficiently, thereby exhibiting stronger overall performance in SAT satisfiability solving tasks.

4.3. Model Solving Performance and Result Analysis

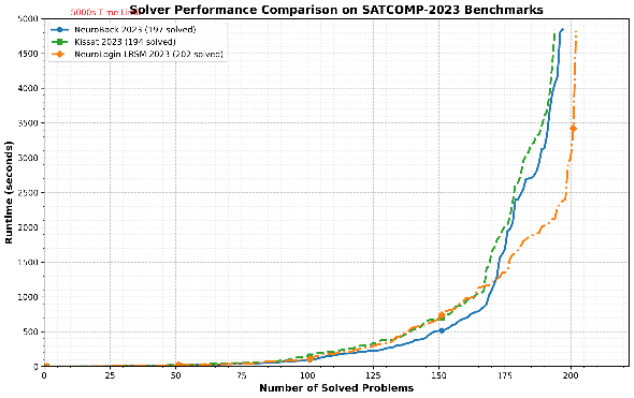
To validate the practical engineering value of the NeuroLogic-GSM model, experiments are conducted on the SATCOMP-2023, 2024, and 2025 industrial datasets as benchmarks. The traditional CDCL solver Kissat and the baseline model NeuroBack are selected as comparison methods. All solvers are evaluated under the same hardware and software environment, with a unified time limit of 5000 seconds on a total of 1200 SAT instances. Model inference relies solely on CPU computation, using 112 independent processes to handle different instances in parallel, ensuring fairness and reproducibility. Among them, NeuroBack successfully solved 353, 331, and 322 instances on SATCOMP-2023, SATCOMP-2024, and SATCOMP-2025, respectively. NeuroLogic-GSM further improves the solving success rate, achieving 355, 331, and 325 solved instances across the three datasets. The CPU inference time of both neural models ranges from 0.1 to 2.3 seconds, with an average of 1.5 seconds. Instances that cannot be successfully solved are reverted to the baseline solver, and therefore are excluded from performance evaluation.

The results are shown in Figure 5. NeuroLogic-GSM demonstrates a clear advantage across all three SATCOMP datasets, consistently maintaining a leading performance throughout the evaluation. On the SATCOMP-2023 dataset, NeuroLogic-GSM successfully solves 202 instances within the 5000-second time limit, outperforming NeuroBack (197 instances, 2.5% improvement) and Kissat (194 instances, 4.12% improvement).

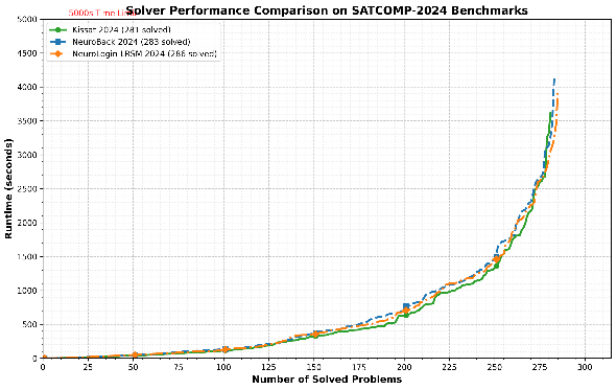
From the curve trends, it can be observed that for hard instances with running times exceeding 1000 seconds, NeuroLogic-GSM exhibits a relatively flatter slope. This indicates that its logical semantic modeling capability is more effective in tackling challenging instances, further reducing redundant backtracking during the search process.

This result validates the unique effectiveness of the recurrent architecture in NeuroLogic-GSM for handling deep logical dependencies. By leveraging a gated recurrent mechanism, NeuroLogic-GSM directly models the logical evolution paths between variables, demonstrating significant advantages in capturing deep logical constraints while maintaining parameter efficiency. It effectively addresses the limitations of traditional CDCL-based methods in modeling long-range dependencies.

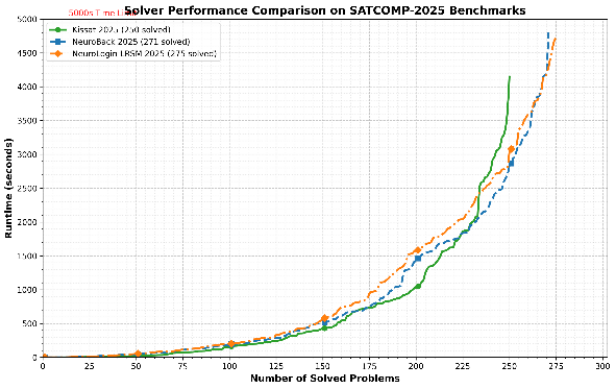
Meanwhile, through iterative state updates, the model continuously enhances its ability to represent global structural information, enabling local decisions to be progressively integrated into a globally consistent reasoning process. As a result, when dealing with SAT instances characterized by long logical chains and complex constraint propagation paths, NeuroLogic-GSM achieves more stable search-space pruning and conflict avoidance, thereby improving overall solving efficiency and robustness.



(a) SATCOMP-2023 Solving Results



(b) SATCOMP-2024 Solving Results



(c) SATCOMP-2025 Solving Results

Figure 5. Model Solving Performance Plot

4.4. Efficiency of Solving SAT and UNSAT Problems

Across a total of 763 instances in the SATCOMP benchmark, there are 360 unsatisfiable (UNSAT) problems and 403 satisfiable (SAT) problems. On the UNSAT subset, the effectiveness of NeuroLogic-GSM is fully validated. Statistical results show that the model outperforms the traditional solver Kissat on 234 instances (65.0%) and outperforms NeuroBack on 173 instances (48.1%), indicating that it has achieved a high level of capability in capturing the structural properties of the Unsat Core.

On the SAT subset, NeuroLogic-GSM also maintains strong performance, outperforming Kissat on 218 instances (54.1%) and NeuroBack on 189 instances (46.9%). These results demonstrate that although neural heuristic methods exhibit slight variations in performance gains across different problem types, NeuroLogic-GSM consistently performs well on both SAT and UNSAT instances, effectively breaking

through the performance bottlenecks of traditional CDCL solvers.

4.5. Ablation Study

To evaluate the contributions of the Gated Recurrent Unit (GRU) and the Logical Processor in the NeuroLogic-GSM model to overall performance, an ablation study is conducted on the same dataset. Three configurations are designed to separately assess the performance of the Logical Processor alone, the GRU alone, and the complete NeuroLogic-GSM model that integrates both components. Table 4 presents the ablation results of each configuration on the pretraining and fine-tuning datasets.

Table 4. Ablation Study Results

DataSet	Gated Recurrent Unit (GRU)	Logical Processor	NeuroLogin-LRSM
Pretrain	0.821	0.751	0.928
Finetune	0.846	0.825	0.97

5. Conclusion

This study focuses on optimizing phase selection heuristics in the Boolean Satisfiability Problem (SAT) and proposes a novel Gated Recurrent Unit-based phase prediction model, NeuroLogic-GSM. First, considering the dynamic evolutionary nature of SAT solving tasks, the overall architecture and design philosophy of the model are introduced in detail, with particular emphasis on the collaborative mechanism between the Logical Processor and the Gated Recurrent Unit (GRU) in multi-round iterative reasoning. Subsequently, under a unified dataset and training configuration, comparative experiments with classical CDCL solvers and existing machine learning methods are conducted. The effectiveness and stability of NeuroLogic-GSM are validated from multiple dimensions, including prediction accuracy, solving efficiency, and engineering robustness.

On this basis, multiple ablation studies are further designed to systematically analyze each core component of the model. The results show that the weight-shared Logical Processor can accurately capture constraint relationships in heterogeneous graphs, while the GRU-based state update mechanism demonstrates significant advantages in modeling temporal dependencies during the reasoning process. In addition, experiments demonstrate that an appropriate number of reasoning iterations helps achieve a balance between model performance and memory consumption. Finally, the results confirm that modeling both logical structures and literal polarities effectively improves the prediction accuracy and stability of the model across different SAT problem instances.

References

[1] Cook S A. The complexity of theorem-proving procedures [M]//Logic, automata, and computational complexity: The works of Stephen A. Cook. 2023: 143-152.

[2] LeCun Y, Bengio Y, Hinton G. Deep learning [J]. nature, 2015, 521(7553): 436-444.

[3] Scarselli F, Gori M, Tsoi A C, et al. The graph neural network model [J]. IEEE transactions on neural networks, 2008, 20(1): 61-80.

[4] Marques-Silva J, Lynce I, Malik S. Conflict-driven clause learning SAT solvers [J]. Handbook of satisfiability, 2009: 131-153.

[5] Cho K, Van Merriënboer B, Gulçehre Ç, et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation [C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1724-1734.

[6] Liang J H, Ganesh V, Poupart P, et al. Learning rate based branching heuristic for SAT solvers [C]//International Conference on Theory and Applications of Satisfiability Testing. Cham: Springer International Publishing, 2016: 123-140.

[7] Bresler G, Huang B. The algorithmic phase transition of random k-sat for low degree polynomials [C]//2021 IEEE 62nd annual symposium on foundations of computer science (FOCS). IEEE, 2022: 298-309.