AsymSAT: Accelerating SAT Solving with Asymmetric Graph-based Model Prediction

Zhiyuan Yan[1], Min Li[2], Zhengyuan Shi[2], Wenjie Zhang[3], Yingcong Chen[1 4] and Hongce Zhang[1 4]
[1]The Hong Kong University of Science and Technology (Guangzhou) [2]The Chinese University of Hong Kong
[3]Peking University [4]The Hong Kong University of Science and Technology

***Abstract*—Though graph neural networks (GNNs) have been used in SAT solution prediction, for a subset of symmetric SAT problems, we unveil that the current GNN-based end-to-end SAT solvers are bound to yield incorrect outcomes as they are unable to break symmetry in variable assignments. In response, we introduce AsymSAT, a new GNN architecture coupled where a recurrent neural network is (RNN) to produce asymmetric models. Moreover, we bring up a method to integrate machine-learning-based SAT assignment prediction with classic SAT solvers and demonstrate its performance on non-trivial SAT instances including logic equivalence checking and cryptographic analysis problems with as much as 75.45% time saving.**

I. Introduction

Recently, there has been a growing interest in applying machine learning methods to solve SAT problems. For instance, NeuroSAT [1] takes the Conjunctive Normal Form (CNF) formula as input, while DG-DAGRNN [2] directly processes the circuit as input (namely, circuit-SAT problems). Methods like NeuroSAT and DG-DAGRNN predict assignments through graph neural networks (GNNs) that encode either CNF formulas or circuits as graphs.

However, existing GNN-based SAT-solving methods like NeuroSAT and DG-DAGRNN have a fundamental flaw: there exists a set of satisfiable CNF formulas (or a set of satisfiable circuits for the Circuit-SAT problem) whose satisfying assignments cannot be learned by the existing methods in [1] or [2]. Specifically, these CNF formulas (or circuits) have symmetric formulations, but their solutions are asymmetric. The symmetry here means swapping a pair of variables results in an equivalent CNF formula or circuit, but in a satisfying assignment, these symmetric variables must take different values, such as "a XOR b." Because existing GNN models solely learn from the structural information in the SAT problem, a and b will share the same embedding due to symmetry. Therefore, the predictions for a and b inevitably become the same, which are clearly not a satisfying variable assignment. This issue is particularly prominent in contemporary hard SAT problems in the EDA domain, where symmetric structures like XOR are common in, for example, cryptographic circuits.

To address the aforementioned problem, we introduce AsymSAT, a new SAT model prediction approach with an additional recurrent neural network (RNN) in the assignment decoding layers. Since RNN makes sequential predictions, later assignments will be able to "remember" earlier ones, potentially break symmetry and lead to more accurate results.

This work is supported by the National Natural Science Foundation of China (grant no. 62304194) and Guangzhou-HKUST(GZ) Joint Funding Program (grant no. 2023A03J0013).

Furthermore, as end-to-end SAT model prediction can never be 100% accurate, we propose a novel integration strategy to combine GNN predictions with the traditional SAT-solving process. Our approach focuses on the preprocessing stage. Predictions from the machine learning model are converted to additional assumptions to be inserted into the solver, along with the original formula. Our approach preserves the soundness of SAT solving while offering a speed-up using GNN prediction as branch heuristics for nontrivial SAT instances.

II. Our Methods

A. The Proposed GNN Architecture

Graph embedding layers: GNN takes as input the circuit graph where each node corresponds to a circuit input or a logic gate. In GNN, each node is associated with a state vector x_v which is iteratively updated based on the messages aggregated from neighboring nodes. The aggregated message is used to update the state vectors by a standard GRU function $GRU\,(\cdot)$. This is similar to the DG-DAGRNN architecture in the prior work [2].

SAT assignment decoding layers: AsymSAT stands out from the prior works in the assignment decoding layers. For SAT model prediction, we map a sequence of hidden state vectors of the circuit input nodes $X = (x_{i_1}, x_{i_2}, x_{i_3}, ...)$ to a sequence of input assignment L. After iterations of message passing, these hidden state vectors contain the information related to the structure of the graph. If two input nodes are symmetric with respect to each other, their hidden state vectors will be the same. If we individually use each of these vectors to decode a 0-1 assignment, the symmetric nodes will certainly map to the same variable assignment. In our AsymSAT, we use a recurrent neural network (referred to as the $\mathcal{R}$ layer) to generate sequential predictions on variable assignments, so that the model output on a certain circuit input node depends on the predictions of other nodes. We make this $\mathcal{R}$ layer bi-directional to account for dependencies from both sides. Regarding the aforementioned XOR example, we expect this RNN layer will be able to learn to predict different variable assignments for the two symmetric variables after training. We use Cross-Entropy as the loss function to train our model.

B. Guiding the classical SAT solver via AsymSAT

For a variable v_i in Boolean formula F, suppose that AsymSAT predicts that it will likely take a true (or false) value, we can add v_i (or $\neg v_i$) as a unit clause (an assumption) to the existing set of clauses in F. We denote the augmented formula

979-8-3503-4859-0/DATE24/© 2024 EDAA

TABLE I
AVERAGE RUNTIME AND THE NUMBER OF SOLVED PROBLEMS OF SAT SOLVING W/ OR W/O ASYMSAT

	Size	MiniSAT w/o(# Solved)	MiniSAT w/(# Solved)	MiniSAT Inference(s)	MiniSAT w/o(s)	MiniSAT w/(s)	MiniSAT Overall(s)	MiniSAT Time Saving	CadiCal w/o(# Solved)	CadiCal w/(# Solved)	CadiCal Inference(s)	CadiCal w/o(s)	CadiCal w/(s)	CadiCal Overall(s)	CadiCal Time Saving
SAT	86312	19	**20**	25.06	332.99	166.49	**191.55**	**42.48%**	23	**25**	28.92	380.90	64.58	**93.50**	**75.45%**
UNSAT	1613	22	**23**	0.86	383.11	375.80	**376.66**	**1.9%**	26	**26**	0.85	296.53	265.78	**266.63**	**10.08%**

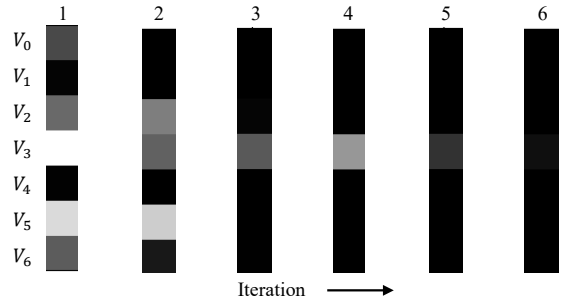

Fig. 1. The confidence level of variable assignments in each iteration on an $SR(7)$ problem.

as F'. If we are certain of multiple variable assignments, we may add multiple unit clauses at the same time. If the solver concludes F' is satisfiable, the original F is also satisfiable. On the other hand, if F' is unsatisfiable, we are unsure if F is also unsatisfiable. For completeness, we also need to check the case when variables are not taking the predicted values. Thus, we construct a clause c' formed by negating the conjunction of all unit clauses that were added in F'. Subsequently, we check the satisfiability of F'', where $F'' := F \wedge c'$. If F'' is satisfiable (or unsatisfiable), then F is also satisfiable (or unsatisfiable).

AsymSAT does not fix the assignment to all variables in F'. Instead, it only selects a subset whose predicted soft assignments are close to 0 or 1. This indicates the model has a higher confidence in these assignments. We pick variables from the results of earlier iterations. An intuitive explanation is illustrated by Figure 1, which shows the confidence level of variables after each message passing iteration. Although nearly all variables converge to a high confidence (indicated by the darker color) at the end of iteration 6, predictions on some variables (such as V_1 and V_4) tend to have a high confidence right from the beginning. Similar to the classic SAT solvers that pick decision variables following certain pre-coded heuristics, AsymSAT seems to also pick the decision variables iteratively. Therefore, by using high-confidence variable assignment in the earlier iterations, we are essentially taking the variable decision order of a neural network solver to a classic SAT solver.

III. EXPERIMENTAL EVALUATION

For training, we use $8K$ $SR(n)$ problems sampled uniformly from $SR(U(3, 10))$. For AsymSAT-guided SAT solving, we mainly target SAT instances that are non-trivial for traditionally classic SAT solvers. We collect a dataset comprising 26 SAT and 26 UNSAT instances sourced from industrial use cases.

TABLE II
SOLVING TIME WHEN USING HIGH-CONFIDENCE VS. LOWER-CONFIDENCE PREDICTED ASSIGNMENTS

MiniSAT	cryp_1	cryp_2	cryp_3	cryp_4	cryp_5	cryp_6	cryp_7	cryp_8
w/o AsymSAT	30.89	168.80	—	183.77	461.07	—	—	—
w/ high con.	**10.91**	188.60	—	**38.05**	**134.63**	**59.42**	**610.77**	**1149.543**
w/ low con.	75.29	**110.26**	—	324.98	485.22	—	2458.32	—

These instances specifically focus on intricate cryptographic and logic equivalence checking problems known for their complexity in SAT solving. We integrate AsymSAT into both MiniSAT [3] and CaDiCaL [4] to demonstrate the universal effectiveness of our method. The time-out limit is 4000 seconds.

Table I presents the average runtime and the number of solved problems for both the baseline and our approach. "Inference" refers to the time required by AsymSAT to make predictions. The average runtime is computed for problems solvable either with or without AsymSAT. In comparison to the baseline solvers, MiniSAT and CaDiCaL with AsymSAT exhibit solving time reduction of 42.48% and 75.45% respectively. Among the 25 solved instances, 17 are solved with only one SAT query, indicating the predictions of a partial solution are successful. Regarding the UNSAT cases, there will certainly be two SAT queries, where each searches for a solution in a smaller space. Though UNSAT cases require two queries, there is still room for speed-up, attributed to the change of variable decision order by selecting high-confidence variables. Actually, the total runtime for UNSAT cases also decreases in the experiment by 1.9% and 10.08%.

The effectiveness of using predicted variable assignment with higher confidence: we utilize 8 cryptography-related circuits, where 7 could not be solved with a single query. Table II compares the outcomes when using assignments with high-confidence variables (w/ high con), vs. low-confidence variables (w/ low con). "—" indicates time-out. It can be seen that using high-confidence variable assignments in general is faster and solves more cases than either using low-confidence assignments or the baseline without AsymSAT. This indicates high-confidence variable assignment in early message-passing iterations is potentially linked with the variable(s) to branch on in SAT solving.

IV. CONCLUSION

This paper addresses the need of considering symmetry-breaking when designing a GNN model for SAT solving by adding a recurrent neural network in assignment decoding layers. We introduce a method to integrate machine-learning-based SAT assignment predictions with traditional SAT solvers. This integration yields notable speed-up when tackling intricate SAT instances involving logic equivalence checking and cryptographic analysis.

REFERENCES

[1] D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, and D. L. Dill, "Learning a SAT solver from single-bit supervision," *ICLR*, 2019.

[2] S. Amizadeh, S. Matusevych, and M. Weimer, "Learning to solve circuit-SAT: An unsupervised differentiable approach," in *ICLR*, 2019.

[3] N. Sorensson and N. Een, "Minisat v1. 13-a SAT solver with conflict-clause minimization," *SAT*, vol. 2005, no. 53, pp. 1–2, 2005.

[4] S. D. QUEUE, "Cadical at the SAT race 2019," *SAT RACE*, vol. 2019, p. 8, 2019.