NeuroSelect: Learning to Select Clauses in SAT Solvers

Hongduo Liu[1], Peng Xu[1], Yuan Pu[1], Lihao Yin[2], Hui-Ling Zhen[2], Mingxuan Yuan[2], Tsung-Yi Ho[1], Bei Yu[1]

[1]The Chinese University of Hong Kong [2]Huawei Noah's Ark Lab

Boolean Satisfiability

- The Boolean satisfiability (SAT) problem involves finding a satisfying assignment for a Boolean formula or proving that none exists.
- SAT has wide applications in circuit verification, test pattern generation, automatic theorem proving, etc.
- SAT is the first problem proven NP-complete.

Learning for SAT

- End-to-end solvers like NeuroSAT [5]: can only handle toy cases, lack of completeness.
- Learning-aided SAT solvers: use machine learning to improve a SAT solver's heuristics like branching heuristic [4], restart policy [3], etc.

Clause Deletion

Clause deletion in CDCL solvers removes less useful learned clauses to manage memory and computational resources.

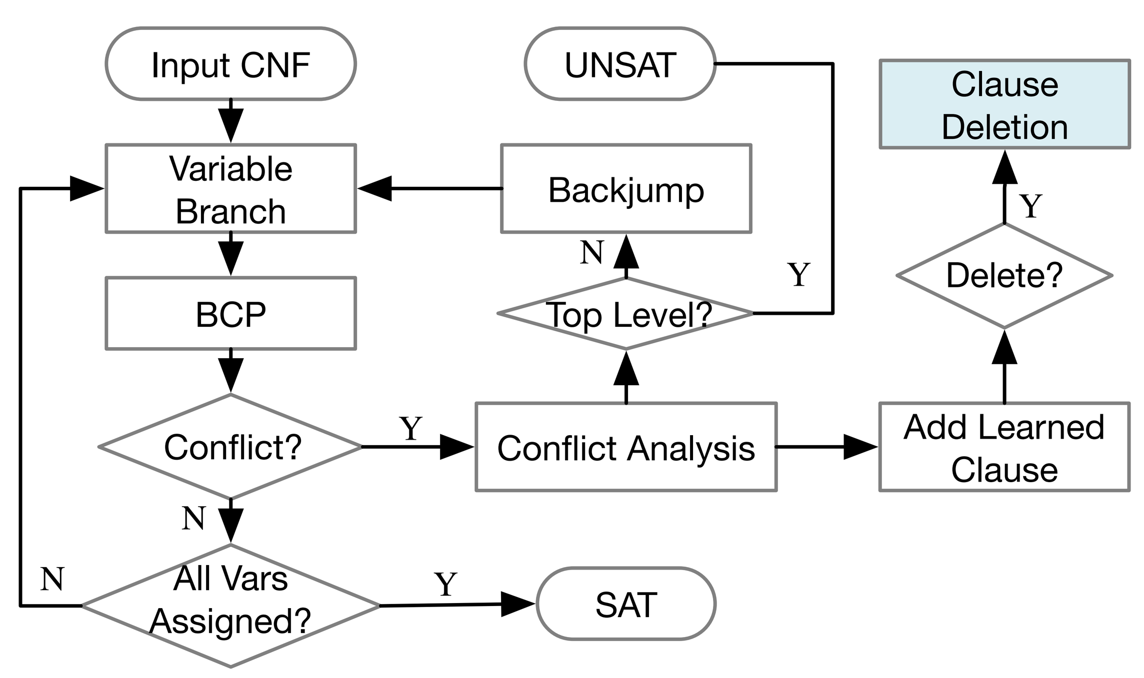


Figure 1. The flow of conflict-driven clause learning (CDCL) algorithm.

Learning for Clause Deletion

How about evaluating the effectiveness of learned clauses using ML?

- ML models are great at identifying patterns in static data, but an SAT solver's state changes frequently as it navigates to a new search space.
- The value of a learned clause depends on its interaction with other chosen clauses, complicating the decision-making process.
- Direct clause evaluation demands model inferences for each learned clause during deletion phases, requiring more computation resources than SAT solving itself.

Clause Evaluation to Policy Evaluation

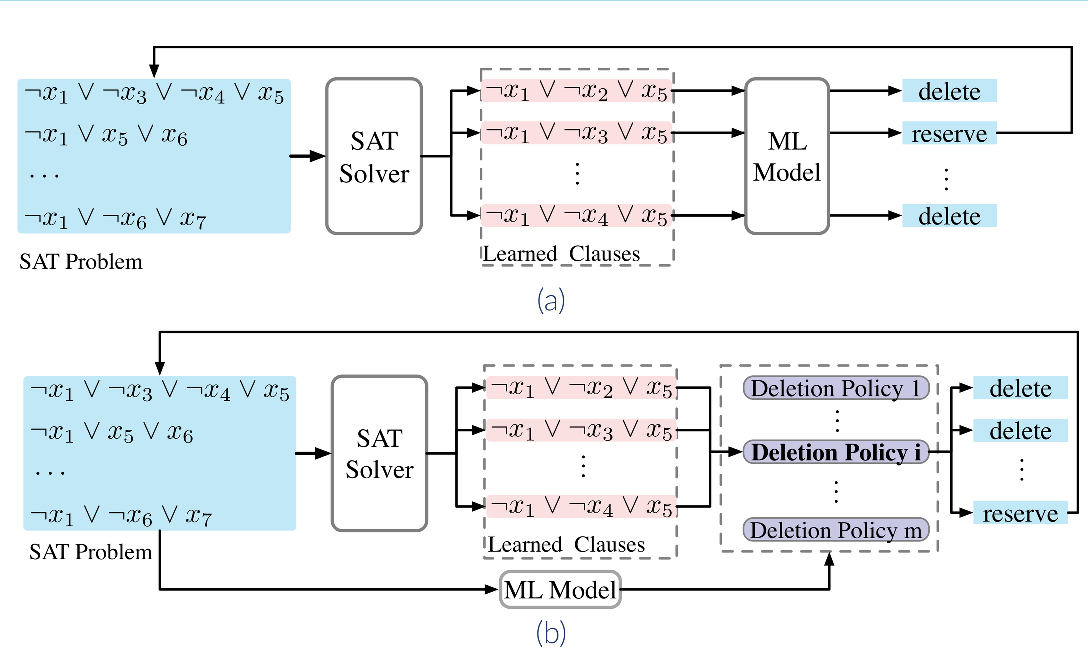


Figure 2. Two learning aided clause deletion mechanisms. (a) Evaluate learned clauses directly; (b) Evaluate clause deletion policies.

Why Evaluating Clause Deletion Policies?

- Effectiveness of clause deletion depends on both characteristics of each learned clause and the deletion policy.
- Clause deletion policy has a lifelong effect during SAT solving.
- Evaluating the deletion policy only requires one-time inference, it can be efficient even on CPUs.

Methodology

- Step 1: generate a complementary clause deletion policy. We propose a new clause deletion metric considering variable propagation frequency.
- Step 2: select the most suitable clause deletion policy. We propose a classification network with local message passing and global attention.

Variable Propogation Frequency

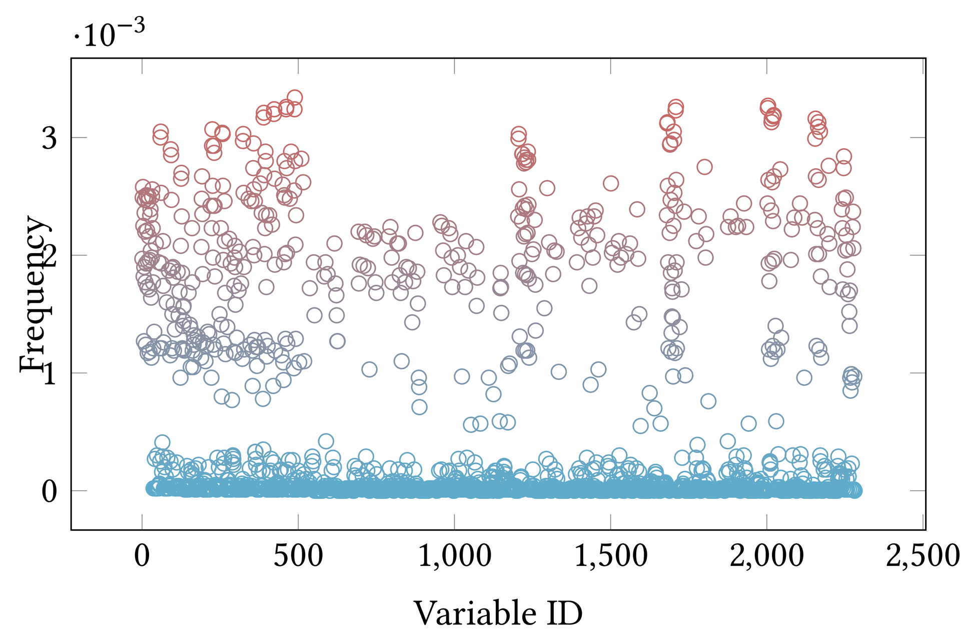


Figure 3. Distribution of variable propagation frequency of a SAT instance from SAT competition 2022. Some variables are propagated significantly more frequently than others.

New Clause Deletion Policy

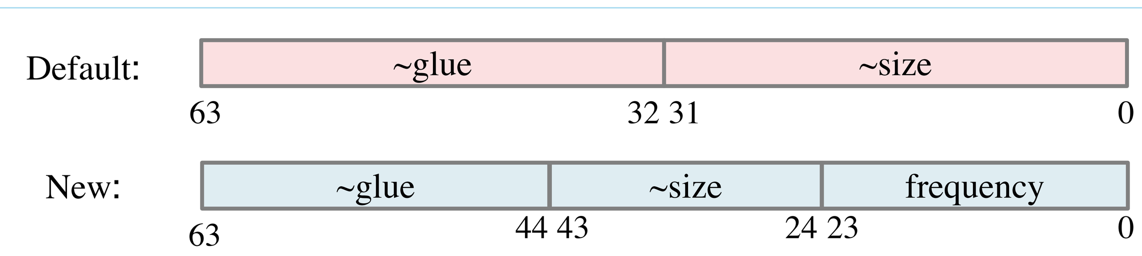


Figure 4. The default learned clause scoring algorithm in Kissat vs. Our new learned clause scoring algorithm considers variable propagation frequency.

$$\text{c.frequency} = \sum_{v \in c} (f_v > \alpha f_{\max}).$$

- f_v indicates the frequency of variable v used to trigger propagation since the last clause deletion.
- $f_{\max}$ represents the maximum propagation frequency among all variables.
- α is an adjustable parameter set to 4/5, according to our empirical studies.

NeuroSelect Overview

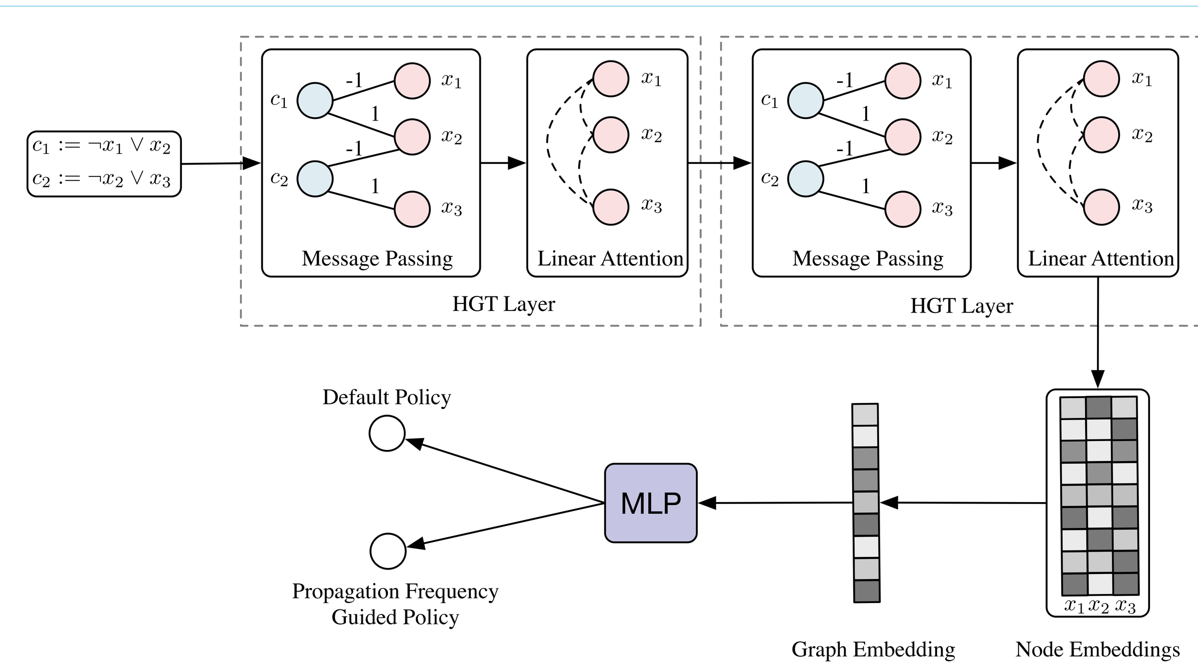


Figure 5. Overview of NeuroSelect.

- Every SAT instance is represented as a weighted bipartite graph.
- The weight is -1 when the variable is negated in the clause.

Hybrid Graph Transformer

- The message-passing comprehends the structural information of the CNF formula.
- The linear attention captures long-term dependencies between variables.
- Linear attention reduces traditional self-attention complexity from quadratic to linear.

Linear Attention

Suppose the input node embedding of the linear attention layer is $\mathbf{Z} \in \mathbb{R}^{N \times d}$. The linear attention function [6] is defined as

$$\begin{aligned} \mathbf{Q} &= f_Q(\mathbf{Z}), & \tilde{\mathbf{Q}} &= \frac{\mathbf{Q}}{\|\mathbf{Q}\|_F}, & \mathbf{V} &= f_V(\mathbf{Z}), \\ \mathbf{K} &= f_K(\mathbf{Z}), & \tilde{\mathbf{K}} &= \frac{\mathbf{K}}{\|\mathbf{K}\|_F}, & \mathbf{D} &= \mathrm{diag}\left(\mathbf{1} + \frac{1}{N}\tilde{\mathbf{Q}}\left(\tilde{\mathbf{K}}^\top \mathbf{1}\right)\right), \end{aligned} \tag{1}$$

where f_Q, f_K, and f_V are linear feed-forward layers to encode the query, key, and value matrix. $\|\cdot\|_F$ denotes the Frobenius norm and $\mathbf{1}$ is an N-dimensional all-one column vector. The *diag* operation changes the N-dimensional column vector into a $N \times N$ diagonal matrix. Subsequently, we have the output of the global attention layer in the format of

$$\mathbf{Z}^{out} = \mathrm{LinearAttn}(\mathbf{Z}) = \mathbf{D}^{-1}\left[\mathbf{V} + \frac{1}{N}\tilde{\mathbf{Q}}\left(\tilde{\mathbf{K}}^\top \mathbf{V}\right)\right]. \tag{2}$$

Datasets

- Training data: SAT competition 2016-2021 instances.
- Testing data: SAT competition 2022 instances.
- An SAT instance is labeled as '1' if it sees at least a 2% reduction in propagations with the new deletion policy compared to the default policy in Kissat; otherwise, it is labeled as '0'.
- Any formula whose graph conversion exceeds 400,000 nodes is excluded to adhere to GPU memory limitations.

Classification Capability of NeuroSelect

Table 1. Performance of different SAT classification models.

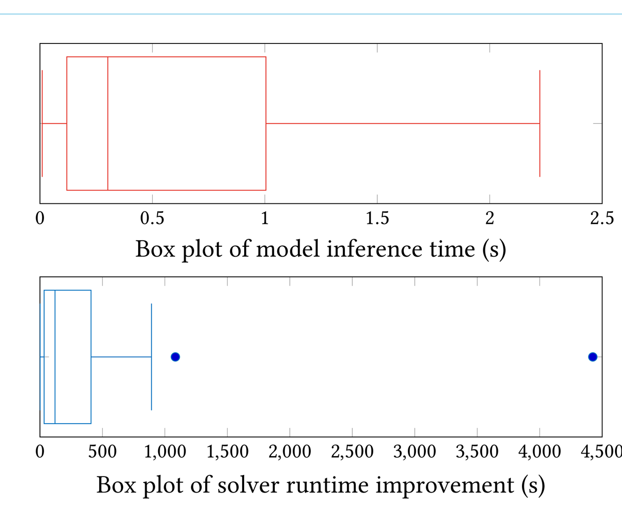
	precison	recall	F1	accuracy
NeuroSAT [5]	47.27%	44.07%	45.61%	56.94%
G4SATBench [2]	43.48%	33.90%	38.10%	54.86%
NeuroSelect w/o attention	56.45%	**58.33%**	57.38%	63.89%
NeuroSelect	**66.00%**	55.00%	**60.50%**	**69.44%**

NeuroSelect-Kissat Performance

Table 2. Runtime statistics of Kissat and NeuroSelect-Kissat on SAT competition 2022 instances.

	solved	median (s)	average (s)
Kissat [1]	274	307.02	713.28
NeuroSelect-Kissat	**274**	**271.34**	**671.73**

Runtime Analysis



Inference time varies between 0.01 and 2.22 seconds on the CPU, while runtime improvement can reach up to 4425 seconds.

References

[1] Armin Biere and Mathias Fleury. Gimsatul, IsaSAT and Kissat entering the SAT Competition 2022. In *Proc. of SAT Competition*, 2022.

[2] Zhaoyu Li, Jinpei Guo, and Xujie Si. G4SATBench: Benchmarking and advancing sat solving with graph neural networks. *arXiv preprint arXiv:2309.16941*, 2023.

[3] Jia Hui Liang, Chanseok Oh, Minu Mathew, Ciza Thomas, Chunxiao Li, and Vijay Ganesh. Machine learning-based restart policy for CDCL SAT solvers. In *Proc. SAT*, pages 94–110, 2018.

[4] Daniel Selsam and Nikolaj Bjørner. Guiding high-performance sat solvers with unsat-core predictions. In *Proc. SAT*, pages 336–353, 2019.

[5] Daniel Selsam, Matthew Lamm, B Benedikt, Percy Liang, Leonardo de Moura, David L Dill, et al. Learning a SAT solver from single-bit supervision. In *Proc. ICLR*, 2018.

[6] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Sgformer: Simplifying and empowering transformers for large-graph representations. In *Proc. NIPS*, 2023.

hdliu21@cse.cuhk.edu.hk