NeuroSelect: Learning to Select Clauses in SAT Solvers

Hongduo Liu\textsuperscript{1}, Peng Xu\textsuperscript{1}, Yuan Pu\textsuperscript{1}, Lihao Yin\textsuperscript{2}, Hui-Ling Zhen\textsuperscript{2}, Mingxuan Yuan\textsuperscript{2}, Tsung-Yi Ho\textsuperscript{1}, Bei Yu\textsuperscript{1}

\textsuperscript{1}The Chinese University of Hong Kong
\textsuperscript{2}Huawei Noah’s Ark Lab
Outline

1 Background
2 Methodology
3 Experimental Results
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\(^1\): can only handle toy cases, lack of completeness.
- Learning-aided SAT solvers: use machine learning to improve a SAT solver’s heuristics like variable branching\(^2\) and restart policy\(^3\).

---


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

The flow of conflict-driven clause learning (CDCL) algorithm.
• **Activity:** Measures frequency of involvement in conflict analysis.
• **Size:** Counts the number of literals in the clause.
• **Glue Value:** Indicates diversity of decision levels involved in the clause.

*Human-designed heuristics are utilized to guide clause deletion. Is it possible to develop a more effective heuristic through learning?*
Dynamic Solver State: A SAT solver’s state changes frequently as it navigates to a new search space.

Inter-clause Dependencies: The value of a learned clause depends on its interaction with other chosen learned clauses.

Clause Evaluation Cost: Direct clause evaluation demands model inferences for each learned clause.
Two learning aided clause deletion mechanisms. (a) Evaluate learned clauses directly; (b) Evaluate 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.
Step 1: Generate a Complementary Clause Deletion Policy
  - Introduced a novel clause deletion metric based on the frequency of variable propagation.

Step 2: Select the Most Suitable Clause Deletion Policy
  - Developed a classification network utilizing local message passing and global attention mechanisms.
Distribution of variable propagation frequency of a SAT instance from SAT competition 2022. Some variables are propagated significantly more frequently than others.
The default learned clause scoring algorithm in Kissat vs. Our new learned clause scoring algorithm considers variable propagation frequency.

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

- \( f_v \) indicates the frequency of variable \( v \) used to trigger propagation since the last clause deletion.
- \( f_{\text{max}} \) represents the maximum propagation frequency among all variables.
- \( \alpha \) is a hyperparameter between 0 and 1.
Runtime Comparison between the default and new clause deletion policy on SAT competition 2022 instances using a standard 5,000 seconds timeout.
Overview

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

Every HGT layer consists of a message-passing layer and a linear attention layer.

- 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

Assume the input node embedding of the linear attention layer is denoted by $Z \in \mathbb{R}^{N \times d}$. The linear attention function\(^4\) is defined as

$$Q = f_Q(Z), \quad \tilde{Q} = \frac{Q}{\|Q\|_F}, \quad V = f_V(Z),$$

$$K = f_K(Z), \quad \tilde{K} = \frac{K}{\|K\|_F}, \quad D = \text{diag} \left( 1 + \frac{1}{N} \tilde{Q} (\tilde{K}^\top 1) \right),$$

where $f_Q, f_K,$ and $f_V$ are linear feed-forward layers to encode the query, key, and value matrix. Subsequently, we have the output of the global attention layer in the format of

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

\(^4\)Qitian Wu et al. (2023). “SGFormer: Simplifying and Empowering Transformers for Large-Graph Representations”. In: Proc. NIPS.
Linear Attention

\[ Z_{out} = \text{LinearAttn}(Z) = D^{-1} \left[ V + \frac{1}{N} \tilde{Q}(\tilde{K}^T V) \right]. \quad (3) \]

- The dimension of \( \tilde{K}^T \) is \( d \times N \) and the dimension of \( V \) is \( N \times d \).
- The complexity of \( \tilde{K}^T V \) is \( N \times d^2 \).
- Given \( d^2 \ll N \), the complexity is linear to \( N \).
### Datasets

#### Table: Statistics of the training and test datasets from SAT competitions.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Year</th>
<th># CNFs</th>
<th># Variables</th>
<th># Clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>2016</td>
<td>74</td>
<td>16,649</td>
<td>86,186</td>
</tr>
<tr>
<td></td>
<td>2017</td>
<td>124</td>
<td>12,863</td>
<td>99,896</td>
</tr>
<tr>
<td></td>
<td>2018</td>
<td>148</td>
<td>13,407</td>
<td>93,094</td>
</tr>
<tr>
<td></td>
<td>2019</td>
<td>131</td>
<td>12,237</td>
<td>68,900</td>
</tr>
<tr>
<td></td>
<td>2020</td>
<td>123</td>
<td>16,921</td>
<td>85,808</td>
</tr>
<tr>
<td></td>
<td>2021</td>
<td>136</td>
<td>16,219</td>
<td>97,434</td>
</tr>
<tr>
<td>Test</td>
<td>2022</td>
<td>144</td>
<td>19,807</td>
<td>104,472</td>
</tr>
</tbody>
</table>

- 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**: Performance of different SAT classification models.

<table>
<thead>
<tr>
<th>Model</th>
<th>precision</th>
<th>recall</th>
<th>F1</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeuroSAT(^5)</td>
<td>47.27%</td>
<td>44.07%</td>
<td>45.61%</td>
<td>56.94%</td>
</tr>
<tr>
<td>G4SATBench(^6)</td>
<td>43.48%</td>
<td>33.90%</td>
<td>38.10%</td>
<td>54.86%</td>
</tr>
<tr>
<td>NeuroSelect w/o attention</td>
<td>56.45%</td>
<td><strong>58.33%</strong></td>
<td>57.38%</td>
<td>63.89%</td>
</tr>
<tr>
<td>NeuroSelect</td>
<td><strong>66.00%</strong></td>
<td>55.00%</td>
<td><strong>60.50%</strong></td>
<td><strong>69.44%</strong></td>
</tr>
</tbody>
</table>


Comparisons between NeuroSelect-Kissat and Kissat on SAT competition 2022 instances.
Table: Runtime statistics of Kissat and NeuroSelect-Kissat on SAT competition 2022 instances.

<table>
<thead>
<tr>
<th></th>
<th>solved</th>
<th>median (s)</th>
<th>average (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kissat</td>
<td>274</td>
<td>307.02</td>
<td>713.28</td>
</tr>
<tr>
<td>NeuroSelect-Kissat</td>
<td>274</td>
<td>271.34</td>
<td>671.73</td>
</tr>
</tbody>
</table>

Both NeuroSelect-Kissat and Kissat solved 274 instances within the time limit. However, NeuroSelet-Kissat has a shorter average runtime, leading to a 5.8% improvement.

---

Inference time varies between 0.01 and 2.22 seconds on the CPU, which can be ignored compared with SAT solving time.
Thanks!