IMPROVING SAT SOLVER WITH GRAPH NETWORKS AND RL

Vitaly Kurin, Saad Godil, Shimon Whiteson, Bryan Catanzaro

CAN RL IMPROVE AN EXISTING SAT SOLVER?
BOOLEAN SATISFIABILITY (SAT) PROBLEM

$$(x_1 \ OR \ x_2) \ AND \ (\ NOT \ x_2 \ OR \ x_3)$$
SAT IS IMPORTANT

- Theoretical computer science;
- Automatic theorem proving;
- Circuit design;
SOLVERS RELY ON HEURISTICS METICULOUSLY CRAFTED BY HUMANS
WHAT DO WE HAVE NOW?

- Graph-Q-SAT (GQSAT), a branching heuristic
- >2x iteration speed-up on random 3-SAT problems
- Generalization to problems 5x in size
- SAT -> unSAT
HOW DID WE ACHIEVE THAT?

- Injecting a model into an existing algorithm
- Graph Representation
- Graph Neural Networks
- Reinforcement Learning (DQN)
def CDCL(formula):
    if trivially_satisfiable(formula):
        return True
    if trivially_unsatisfiable(formula):
        return False

    literal, value = pick_literal(formula)
    formula = propagate(formula, literal, value)
    return CDCL(formula)
CONFLICT LEARNING

X_1 AND x_2 AND x_3 => unSAT?

Add `\textbf{NOT} (x_1 \text{ AND } x_2 \text{ AND } x_3)` to clauses
Injecting a model into an existing algorithm

VSIDS

\[
\begin{array}{ccc}
x_1 & x_2 & x_3 \\
0 & 0 & 0 \\
\end{array}
\quad \rightarrow \quad
\begin{array}{ccc}
x_1 & x_2 & x_3 \\
4.2 & 3.1 & 2.7 \\
\end{array}
\]

\((x_1 \ OR \ x_2) \ AND \ (NOT \ x_2 \ OR \ x_3)\)
SAT AS A GRAPH

\[(x_1 \text{ OR } x_2) \text{ AND } (\neg x_2 \text{ OR } x_3)\]
GRAPH NEURAL NETWORK

Graph Representation

DQN

Graph Nets
DQN

\[ Q(\cdot) = \]

Reward is -0.1 for a non-terminal step.

DQN
TRAINING PIPELINE

- Train model on SAT 50-218 train data
- Evaluate every k-th epoch
- Pick the best
- Evaluate on the test set
METRIC OF SUCCESS

\[
\begin{bmatrix}
\frac{\text{Minisat Steps}}{\text{Our Steps}} & \frac{\text{Minisat Steps}}{\text{Our Steps}} & \cdots & \frac{\text{Minisat Steps}}{\text{Our Steps}} \\
\text{problem 1} & \text{problem 2} & \cdots & \text{problem 100}
\end{bmatrix}
\]
Table 2: MRIR for GQSAT trained on SAT-50-218. Evaluation for SAT-50-218 is on a separate test data not seen during training.

<table>
<thead>
<tr>
<th>dataset</th>
<th>mean</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAT 50-218</td>
<td>2.46</td>
<td>2.26</td>
<td>2.72</td>
</tr>
<tr>
<td>SAT 100-430</td>
<td>3.94</td>
<td>3.53</td>
<td>4.41</td>
</tr>
<tr>
<td>SAT 250-1065</td>
<td>3.91</td>
<td>2.88</td>
<td>5.22</td>
</tr>
<tr>
<td>unSAT 50-128</td>
<td>2.34</td>
<td>2.07</td>
<td>2.51</td>
</tr>
<tr>
<td>unSAT 100-430</td>
<td>2.24</td>
<td>1.85</td>
<td>2.66</td>
</tr>
<tr>
<td>unSAT 250-1065</td>
<td>1.54</td>
<td>1.30</td>
<td>1.64</td>
</tr>
</tbody>
</table>
WHY IS GQSAT EFFICIENT?
WARMING UP THE EXISTING ALGORITHM
PROBLEM STRUCTURE GENERALIZATION
DATA EFFICIENCY

The graph illustrates the relationship between training set size and iterations improvement for different SAT and unSAT categories. The y-axis represents iterations improvement, while the x-axis shows the training set size. Different lines correspond to different SAT and unSAT categories, with SAT 100-430, SAT 250-1065, SAT 50-218, unSAT 100-430, unSAT 250-1065, and unSAT 50-218.
FURTHER WORK

- Training on problems with larger horizon.
- Scaling to larger problems.
- From reducing number of iterations to wallclock time speedup.
ACKNOWLEDGEMENTS

- **Bryan Catanzaro** for inviting me, giving a problem to solve and constant feedback.
- **Saad Godil** for endless everyday discussions, feedback, coding and experiments.
- **Andrew Tao** and **Guy Peled** for computing support.
- **Rajarshi Roy, Robert Kirby, Yogesh Mahajan, Alex Aiken, Mohammad Shoeybi, Rafael Valle, Sungwon Kim, Ryan Prenger** and the rest of the ADLR team @NVIDIA for great discussion and being amazing people.
- **University team** and **ERC @NVIDIA** for making this a great experience.
- **Shimon Whiteson** @Oxford for supervising me.
- **Samsung R&D UK** for funding my DPhil.
- **Wendy Poole** @Oxford for organizing everything related to CDT.

@y0b1byte   vitaliykurin@gmail.com   https://yobibyte.github.io