SAT Algorithm Selection Using Graph Neural Networks

Hadar Shavit

h.shavit@umail.leidenuniv.nl

SAT

- Is the boolean formula satisfiable:

$$(a \vee b) \wedge (b \vee c)$$

- Can't try all options: for n variables there are 2^n options, NP-Hard problem.
- SAT Solvers – Heuristic on which assignments should be tried first

Algorithm Selection of SAT Solvers

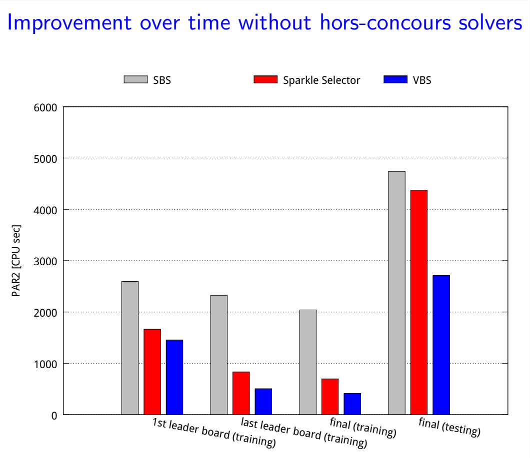
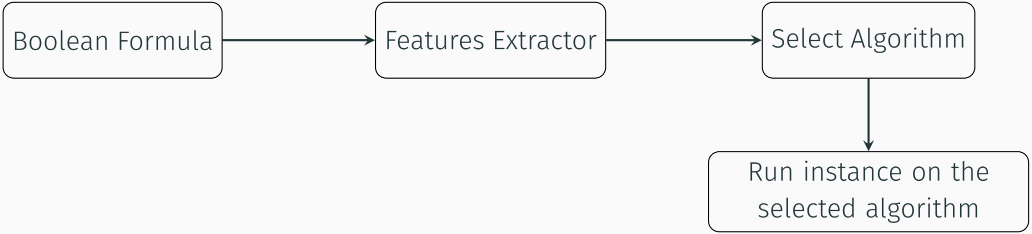


Figure 1: Results of the 2018 Sparkle SAT Challenge [LH18]

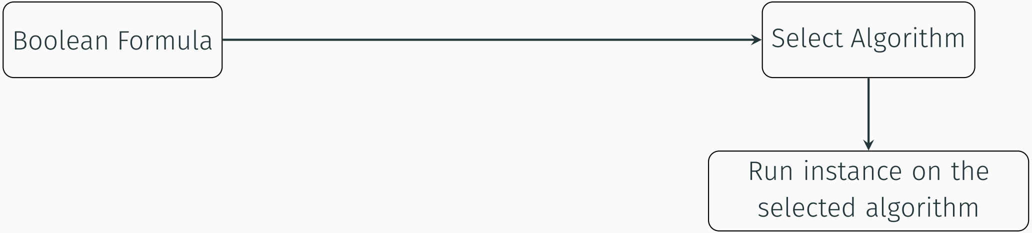
Algorithm Selection of SAT Solvers

- Current state-of-the-art algorithm selection uses features such as:
  - Number of clauses and variables
  - Statistics of the Variable-Clause and Clause graphs



Features Extractor

- Features Extraction requires human knowledge
- Requires recognizing the right set of features
- Can we learn directly from the formula?



End-to-End Algorithm Selection

- Lorregia et al. [LMSS16] showed a CNN for algorithm selection by encoding SAT instance to an image.
- Downsides:
  - No notion of locality
  - Image has resized the image to 128x128
  - Did not reach state-of-the-art level

Graph Neural Networks

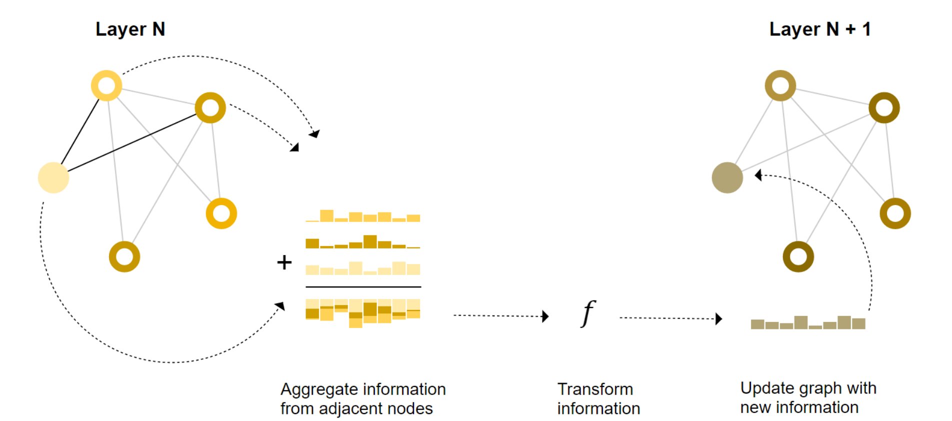


Figure 2: Graph Neural Networks, adapted from [SLRPW21]

Graph Neural Networks for SAT Algorithm Selection

- Multiple binary classification heads, each one selects the best solver from a pair of solvers
- Multi-class classification
- Probability to solve an instance
- Runtime Regression

Closed Gap

$$\frac{m_{SBS} - m_S}{m_{SBS} - m_{VBS}}$$

m_{SBS} is the total runtime of the single best solver, m_S is the total runtime of the AS, m_{VBS} is the virtual best

(Some) Results

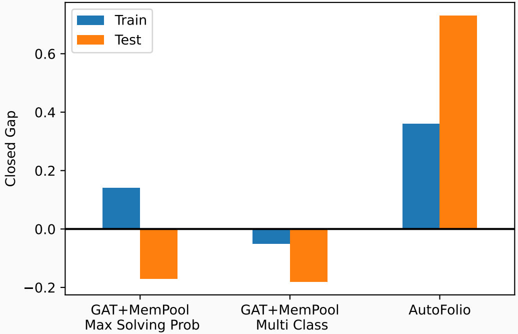


Figure 3: Closed Gap on SAT2011 Crafted + Industrial (only small instances)

Challenges and Future Work

- Large Formulas that cannot fit into GPU memory.
  Possible Solution: sampling.
- Not enough data: there are around 1300 training instances.
  Possible Solution: generate more data (in progress), transfer learning.

References i

- Chuan Luo and Holger H Hoos, *Sparkle sat challenge 2018*, 2018.
- Andrea Loreggia, Yuri Malitsky, Horst Samulowitz, and Vijay Saraswat, *Deep learning for algorithm portfolios*, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, AAAI Press, 2016, p. 1280–1286.
- Inneke Mayachita, *Understanding graph convolutional networks for node classification*, Aug 2020.
- Eugene Nudelman, Kevin Leyton-Brown, Holger H. Hoos, Alex Devkar, and Yoav Shoham, *Understanding random sat: Beyond the clauses-to-variables ratio*, Principles and Practice of Constraint Programming – CP 2004 (Berlin, Heidelberg) (Mark Wallace, ed.), Springer Berlin Heidelberg, 2004, pp. 438–452.

References ii

- Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill, *Learning a SAT solver from single-bit supervision*, CoRR **abs/1802.03685** (2018).
- Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko, *A gentle introduction to graph neural networks*, Sep 2021.

Algorithm Selection of SAT Solvers

Currently, we have manual features

Problem Size Features:
1. **Number of clauses:** denoted c
2. **Number of variables:** denoted v
3. **Ratio:** c/v

Variable-Clause Graph Features:
4-8. **Variable nodes degree statistics:** mean, variation coefficient, min, max and entropy.
9-13. **Clause nodes degree statistics:** mean, variation coefficient, min, max and entropy.

Variable Graph Features:
14-17. **Nodes degree statistics:** mean, variation coefficient, min and max.

Balance Features:
18-20. **Ratio of positive and negative literals in each clause:** mean, variation coefficient and entropy.
21-25. **Ratio of positive and negative occurrences of each variable:** mean, variation coefficient, min, max and entropy.
26-27. **Fraction of binary and ternary clauses**

Proximity to Horn Formula:
28. **Fraction of Horn clauses**
29-33. **Number of occurrences in a Horn clause for each variable:** mean, variation coefficient, min, max and entropy.

DPLL Probing Features:
34-38. **Number of unit propagations:** computed at depths 1, 4, 16, 64 and 256.
39-40. **Search space size estimate:** mean depth to contradiction, estimate of the log of number of nodes.

Local Search Probing Features:
41-44. **Number of steps to the best local minimum in a run:** mean, median, 10th and 90th percentiles for SAPS.
45. **Average improvement to best in a run:** mean improvement per step to best solution for SAPS.
46-47. **Fraction of improvement due to first local minimum:** mean for SAPS and GSAT.
48. **Coefficient of variation of the number of unsatisfied clauses in each local minimum:** mean over all runs for SAPS.

Figure 4: SATzilla Features

Graph Representation of SAT formula

- Variable Graph [NLBH+04]: node for each variable, an edge between variables that occur in at least one clause

Graph Representation of SAT formula

- Clause Graph [NLBH+04]: node for each clause, edge whenever they share a negated literal

Graph Neural Networks

- Message passing between nodes
- In the kth iteration, the value of node i is: $x_i^{(k)} = \gamma^{(k)}(x_i^{(k-1)}, \square_{j \in N(i)} \phi^{(k)}(x_i^{(k-1)}, x_j^{(k-1)}, e_{j,i}))$ γ, ϕ are differentiable functions (MLP, LSTM), $\square$ is an aggregation function (sum, mean), $N(i)$ are the neighbors of i.

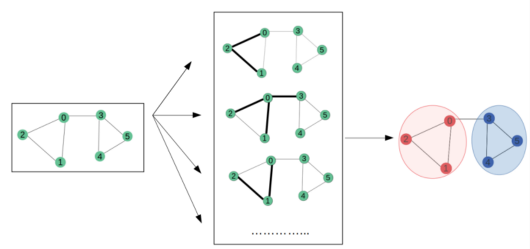


Figure 5: From [May20]

Graph Representation of SAT formula

- Variable Clause Graph [NLBH+04]: node for each variable, clause and edge whenever a variable occurs in a clause

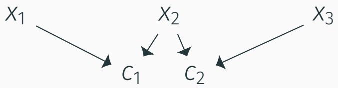


Figure 6: Literal Clause Graph [SLB+18]

Graph Representation of SAT formula

- Literal Clause Graph [SLB+18]: node for every literal (2 nodes per variable, x and $\neg x$), edge between each literal and a clause that contains it, the second type of edges between the two literals of the same variable

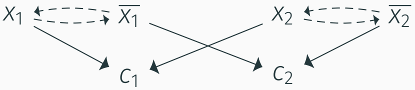


Figure 7: Literal Clause Graph [SLB+18]

Graph Level Prediction

- Each node has a “vote” and we aggregate the votes
- Aggregate the nodes latent features using add, mean or max
- Local Pooling: similar to pooling in CNNs

Graph Neural Networks for SAT

- End to End solving:
  - Predicting whether the formula is SAT or unSAT
  - NeuroSAT –
    - End to End SAT solving from literal clause graph.
    - Uses an LSTM to calculate the node encodings.
    - Used on small formulas (up to 40 variables).
    - Prediction is performed per node, and aggregated by the global mean.

Graph Neural Networks for SAT

- Heuristics for existing solvers: Providing a prediction per node whether this node is "interesting"
  - NeuroCore – NeuroSAT's version for heuristics (Main differences: LSTM $\rightarrow$ MLP, less iterations)
  - NeuroComb – Another architecture to provide predictions for important clauses and variables, trained on small-medium sized formulas, tested on SATCOMP 2021.
  - GraphQSAT – learning a heuristic with Q learning and GNN, trained and tested on small formulas (1065 variables, 250 clauses)