Learning Clause Deletion Heuristics with Reinforcement Learning

Pashootan Vaezipoor[1], Gil Lederman[3], Yuhuai Wu[1,2], Roger Grosse[1,2] and Fahiem Bacchus[1]

[1]University of Toronto
[2]Vector Institute
[3]UC Berkeley

Abstract

We propose a method for training of clause deletion heuristics in DPLL-based solvers using Reinforcement Learning. We have implemented it as part of a software framework `SAT-Gym` which we plan to release as an OpenAI Gym compatible environment. We present experiments and preliminary results for the clause deletion heuristic in Glucose.

1 Introduction

Solvers for difficult combinatorial problems such as SAT, QBF, etc., rely on heuristics that are used in many different phases of their computation. Years of human experience and experimentation have lead to very effective heuristics for many different types of solvers. However, these achievements have been quite painstaking and leave open the question of whether other heuristics could yield better performance. The focus of this work, similar to [5, 6], is to use *Machine Learning (ML)* to automatically learn those heuristics. In particular, there has been recent interest in framing the learning process in a *Reinforcement Learning (RL)* setting [4, 3, 7]. We argue that this is a desirable design choice for several reasons: First, a solver is a dynamic process and changes to the heuristics directly affect the landscape of the future observations. Hence a model that is trained offline in a supervised manner might fail to capture this inherently non-stationary behaviour; Second, an RL-based approach, allows for training the heuristic directly towards optimizing the desired metric (e.g., number of decisions, running time, etc.). This is again in contrast to the supervised setting in which one often needs to substitute that desired metric with a surrogate labeling mechanism. Under these considerations, we propose a RL architecture to learn a *clause deletion* policy to improve the running time of SAT solvers.

Why Clause Deletion? In order to use *Deep Neural Networks (NN)* in the loop of a modern solver we need to address the time-scale mismatch: In the time it takes the NN to make one informed decision, the solver can take thousands. Consequently, we need to make less frequent queries to the NN oracle for it to be feasible. Clause deletion in *Conflict-Driven Clause Learning (CDCL)* SAT solvers naturally fits this criteria as it is executed much less frequently than other heuristics.

Clause deletion is an integral component of CDCL solvers, as the solver accumulates a large number of clauses and as a result starts to slow down. Deleting the learned clauses periodically has proven to be effective in speeding up the process.

2 Clause Deletion in `SAT-Gym`

In (episodic) RL, we consider an agent that interacts with an environment over finite discrete time steps. The agent gets observations from the environment, takes actions, and accumulates reward. In our setting the environment is Glucose, a time step is a garbage collection, and the agent takes the decision of which learned clause to keep.

Observation: The observation is a set of solver state features that includes: 1. The ratio of learned to original clauses, 2. A histogram of the LBD scores of recently learned clauses along with their average, 3. Moving averages of both the recent trail size and recent decision levels.

Action: We use a policy gradient algorithm that directly optimizes a stochastic policy, which at each time step outputs a distribution over the set of actions. In each time step (garbage collection) the agent has to decide for each clause out of N whether to keep or to drop it. A naïve implementation results in a discrete action space of size 2^N, where N is on the order of 2000. In order to overcome this curse of dimensionality we use the *Literals Block Distance (LBD)* [1] value of a clause, which is the standard metric for the "usefulness of clauses" (clauses with lower LBD values are more useful). We constrain our policy to output as action an integer *LBD threshold*, and delete all clauses with LBD values above the threshold.

Reward: Our goal is to improve the running time of the SAT solver, however due to volatility of CPU time, we count the number of "logical operations" *op* that the solver performs, as a more stable and deterministic replacement. These logical operations consist of the number of times the solver accesses the clauses clause during unit propagation. We have observed a high correlation between the *op* and the wall-clock solving time of an instance, which makes *op* a viable surrogate.

Episodes are rolled until solved, or are *aborted* if they accumulate more than 10^9 logical operations. We define the reward of a successfully solved episode to be $200 - op \times 10^{-7}$. The reward of an aborted episode is set to 0.

3 Training

We provide an OpenAI Gym [2] compatible environment that includes a "solver framework" that allows the user to replace different parts of a standard solver (Glucose in our case). The framework currently supports learning of branching and clause deletion heuristics in SAT and 2QBF solvers.

For our dataset, we used the Cellular Automata benchmark which is part of SATCOMP-2018. We chose this particular benchmark because it allowed us to generate a large dataset with tunable difficulty level.

We have preliminary results indicating we can train a policy to improve its reward on 500 formulas from this dataset using our framework (Figure 1). We are actively working on improving our feature set as well as more elaborate hyper-parameter optimization for our training in order to achieve a viable heuristic that can improve the state of the art.

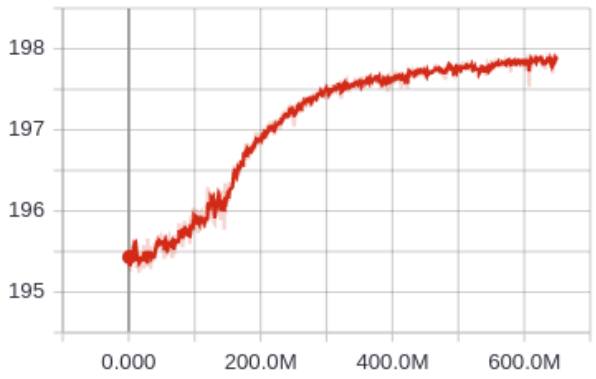

Figure 1: The average episode reward of the agent on 500 formulas from Cellular Automata benchmark over the course of training.

References

[1] G. Audemard and L. Simon. Predicting Learnt Clauses Quality in Modern SAT Solvers. In *Twenty-first International Joint Conference on Artificial Intelligence*, 2009.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] V. Kurin, S. Godil, S. Whiteson, and B. Catanzaro. Improving SAT Solver Heuristics with Graph Networks and Reinforcement Learning, 2019.

[4] G. Lederman, M. N. Rabe, and S. A. Seshia. Learning Heuristics for Automated Reasoning Through Deep Reinforcement Learning. *CoRR*, abs/1807.08058, 2018.

[5] D. Selsam and N. Bjørner. NeuroCore: Guiding CDCL with Unsat-Core Predictions. *arXiv preprint arXiv:1903.04671*, 2019.

[6] M. Soos, R. Kulkarni, and K. S. Meel. CrystalBall: Gazing in the Black Box of SAT Solving.

[7] E. Yolcu and B. Poczos. Learning Local Search Heuristics for Boolean Satisfiability. In *Advances in Neural Information Processing Systems*, pages 7990–8001, 2019.