# Improving SAT Solver Heuristics with Graph Networks and Reinforcement Learning

Vitaly Kurin[1], Saad Godil[2], Shimon Whiteson[1], Bryan Catanzaro[2]

[1] Department of Computer Science, University of Oxford, [2] NVIDIA, US.
Correspondence to `vitaly.kurin@cs.ox.ac.uk`

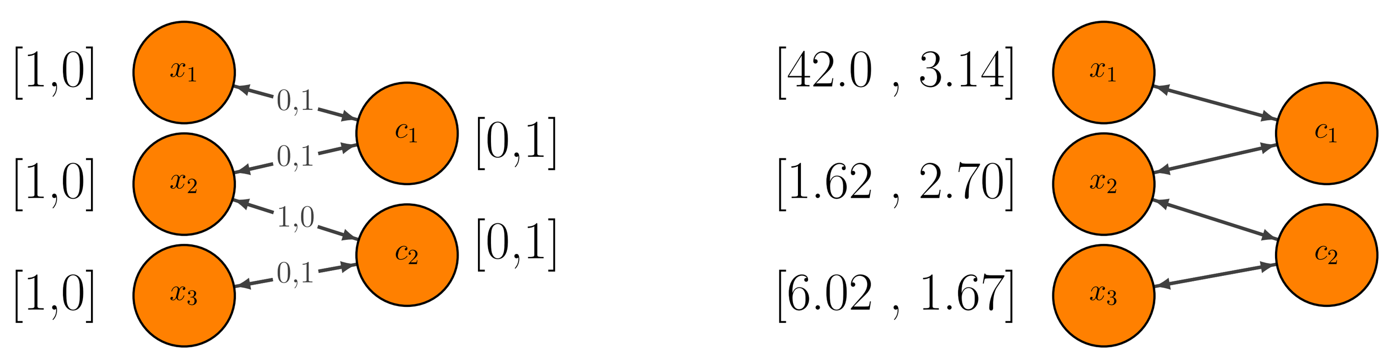
## Can RL improve existing heuristics?

- Boolean Satisfiability (SAT) impacts many fields of the industry and academia, e.g. formal verification, chip design, security, combinatorial optimisation.
- SAT solvers rely on heuristics elaborately crafted with a lot of trial and error by humans.
- Some of the heuristics need a *warm-up* period.
- A solver should always give a correct answer.
- Pre-solving phase computation is cheap (e.g. training models)
- SAT is a sequential decision problem.

## Conflict-Driven Clause Learning (CDCL)

```
Function CDCL(initial assignments):
 while not solved do
   var, value ← branching_heuristic();
   unit_propagation(var, value);
   build_implication_graph();
   analyse_conflicts();
 end
return SAT assignments OR unSAT
```

## SAT problem as a graph




(a) Graph representation of $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$

(b) Graph Q-function values for setting variables to *true* and *false* respectively.

## Graph-Q-SAT (GQSAT)

- GQSAT replaces VSIDS heuristic in CDCL for the first $k$ steps while VSIDS is warming up.
- GQSAT uses DQN with a graph neural network as a function approximator.

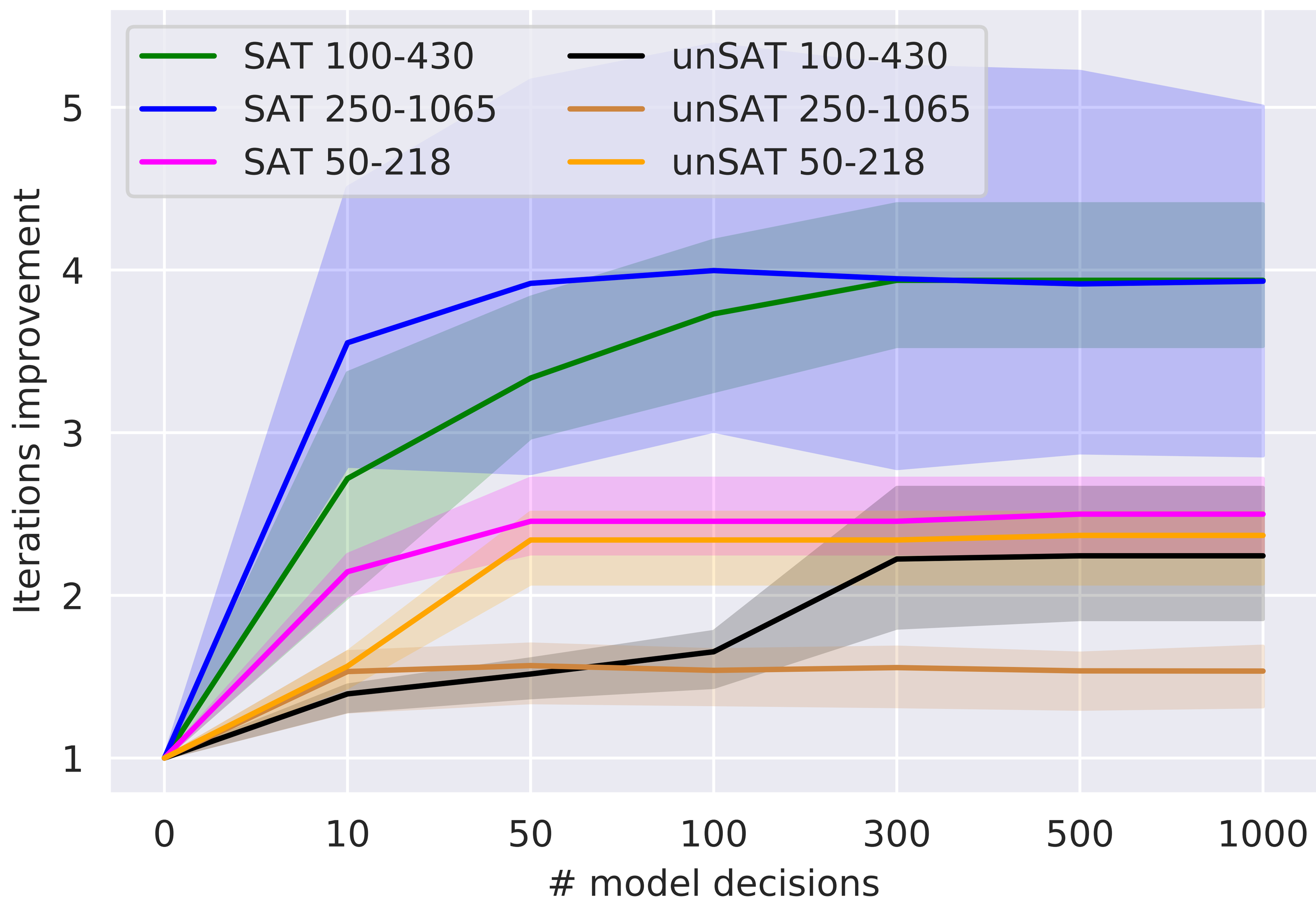
## GQSAT reduces number of decisions by 2-3X

- GQSAT improves VSIDS.
- GQSAT generalises across problem size.
- GQSAT generalises from SAT to unSAT.

MRIR for GQSAT (SAT-50-218)

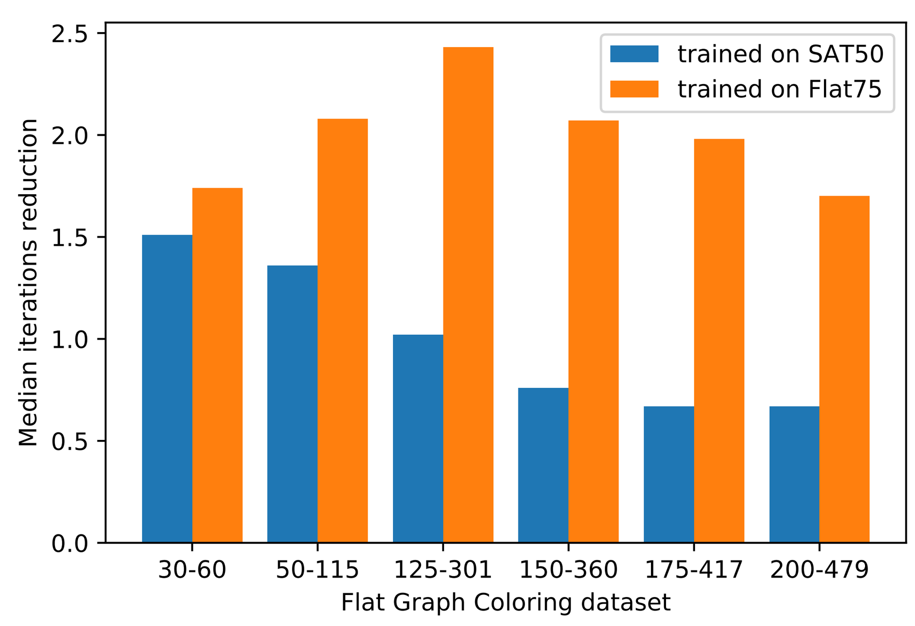
| dataset | mean | min | max |
|---|---|---|---|
| SAT 50-218 | 2.46 | 2.26 | 2.72 |
| SAT 100-430 | 3.94 | 3.53 | 4.41 |
| SAT 250-1065 | 3.91 | 2.88 | 5.22 |
| unSAT 50-128 | 2.34 | 2.07 | 2.51 |
| unSAT 100-430 | 2.24 | 1.85 | 2.66 |
| unSAT 250-1065 | 1.54 | 1.30 | 1.64 |

## GQSAT makes efficient decisions from step one

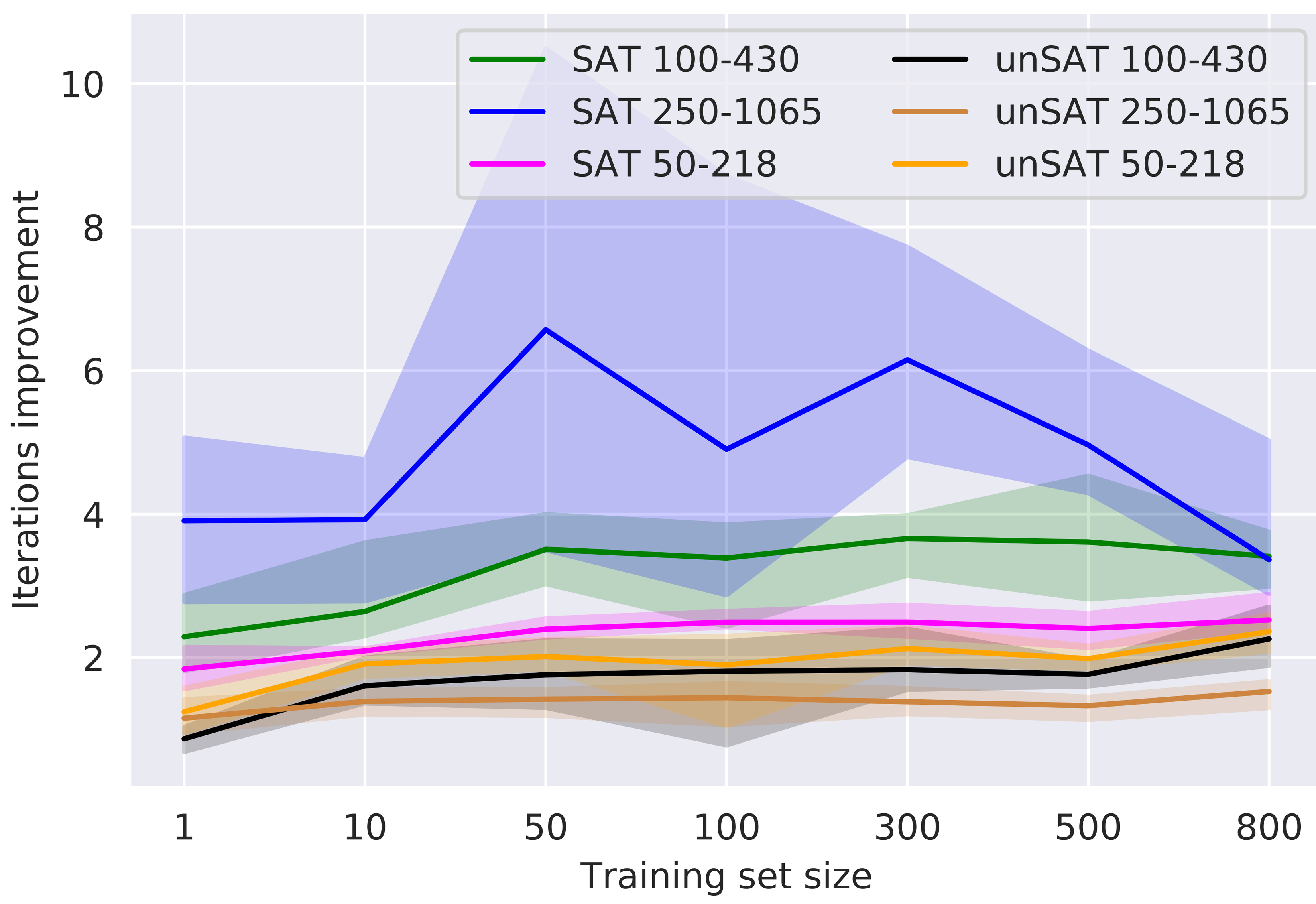



## GQSAT generalizes to other problem structures to a lesser extent

| dataset | variables | clauses | MiniSat iterations |
|---|---|---|---|
| flat-30-60 | 90 | 300 | 10 |
| flat-50-115 | 150 | 545 | 15 |
| flat-75-80 | 225 | 840 | 29 |
| flat-125-301 | 375 | 1403 | 106 |
| flat-150-360 | 450 | 1680 | 179 |
| flat-175-417 | 525 | 1951 | 272 |
| flat-200-479 | 600 | 2237 | 501 |




## GQSAT is data efficient




## Future Work

- Investigating graph structure influence on GQSAT performance.
- Interpreting the results using the graph structure.
- Scaling to larger problems.
- From reducing iterations to speeding up.

**Acknowledgements**: The authors would like to thank Rajarshi Roy, Robert Kirby, Yogesh Mahajan, Alex Aiken, Mohammad Shoeybi, Rafael Valle, Sungwon Kim, Ryan Prenger and the rest of the Applied Deep Learning Research team at NVIDIA for useful discussions and feedback. The authors would also like to thank Andrew Tao and Guy Peled for providing computing support.