

Hard-Core UNSAT Generation

Generating Hard Unsatisfiable Boolean-SAT Instances by
Leveraging Cores

Boolean Satisfiability:

Circuits \rightarrow Boolean Eq. \rightarrow SAT

SAT is **Hard!**

NP-Complete

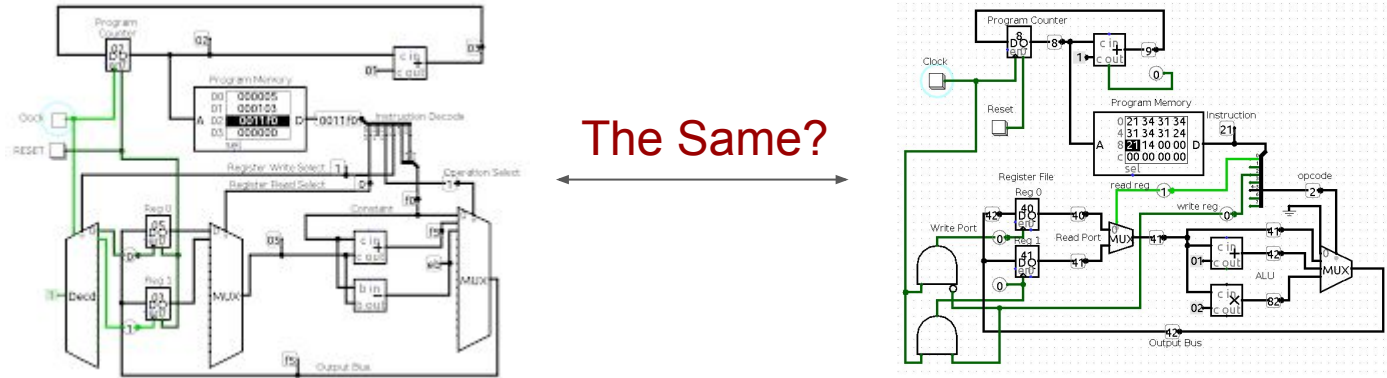
Scarce realistic data



$$(\bar{A} \vee \bar{B} \vee C) \wedge (A \vee \bar{C}) \wedge (B \vee \bar{C})$$

Motivation

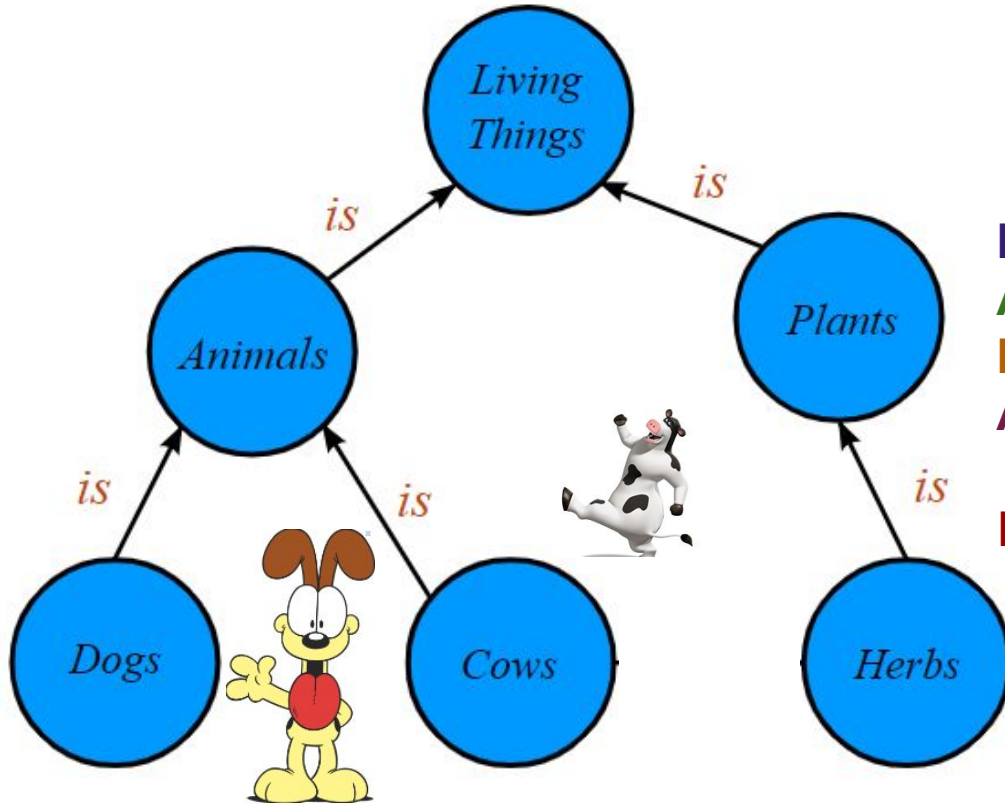
Industrial Applications:

Formal Verification - **Circuit design**, knowledge trees,



Type	Operation	CNF Sub-expression
 AND	$C = A \cdot B$	$(\bar{A} \vee \bar{B} \vee C) \wedge (A \vee \bar{C}) \wedge (B \vee \bar{C})$
 NAND	$C = \overline{A \cdot B}$	$(\bar{A} \vee \bar{B} \vee \bar{C}) \wedge (A \vee C) \wedge (B \vee C)$

Motivation



Dog → **Animal**

Animal → **Dog** or **Cow**

Dog → not **Cow**

Animal → **Living Thing**

Is Dog a Living Thing?

$(\neg D \vee A) \wedge$

$(\neg A \vee D \vee C)$

\wedge

$(\neg D \vee \neg C) \wedge$



$(\neg A \vee L) \wedge$

$(\neg D \vee L)$

SAT → **Yes**

UNSAT → **No**

Background and Terminology

Type	Operation	CNF Sub-expression
 AND	$C = A \cdot B$	$(\bar{A} \vee \bar{B} \vee C) \wedge (A \vee \bar{C}) \wedge (B \vee \bar{C})$
 NAND	$C = \overline{A \cdot B}$	$(\bar{A} \vee \bar{B} \vee \bar{C}) \wedge (A \vee C) \wedge (B \vee C)$

Clause (red arrow pointing to $(\bar{A} \vee \bar{B} \vee C)$)

Literal (blue arrow pointing to \bar{A})

CNF / Instance (black arrow pointing to the entire expression)

Previous Models

	W2SAT	HardSATGEN	G2MILP	ours
Average Hardness (%)	0	185	0.22	262
Average Time Cost (s)	3.36	6441	4.86	5.46

We want Hard Generations

Hardness (%) = generated solve-time / original solve-time

We want Cheap Generations

Time Cost (s) = model inference run-time / number of generations

Satisfiability

$$x_6 \vee x_2$$

$$\neg x_6 \vee x_2$$

$$\neg x_2 \vee x_1$$

$$\neg x_1$$

$$\neg x_6 \vee x_8$$

$$x_6 \vee \neg x_8$$

$$x_2 \vee x_4$$

$$\neg x_4 \vee x_5$$

$$x_7 \vee x_5$$

$$\neg x_7 \vee x_5$$

$$\neg x_5 \vee x_3$$

$$\neg x_3$$

How Hard is an UNSAT instance?

As Hard as its Minimal Unsatisfiable Set

Smallest subset of clauses of a CNF that is still unsatisfiable

Core Hardness → Instance Hardness

Hardness Collapse when Generating: Trivial Cores

$$(\neg A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee \neg B)$$



$$(A \vee B) \wedge (\neg A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee \neg B)$$

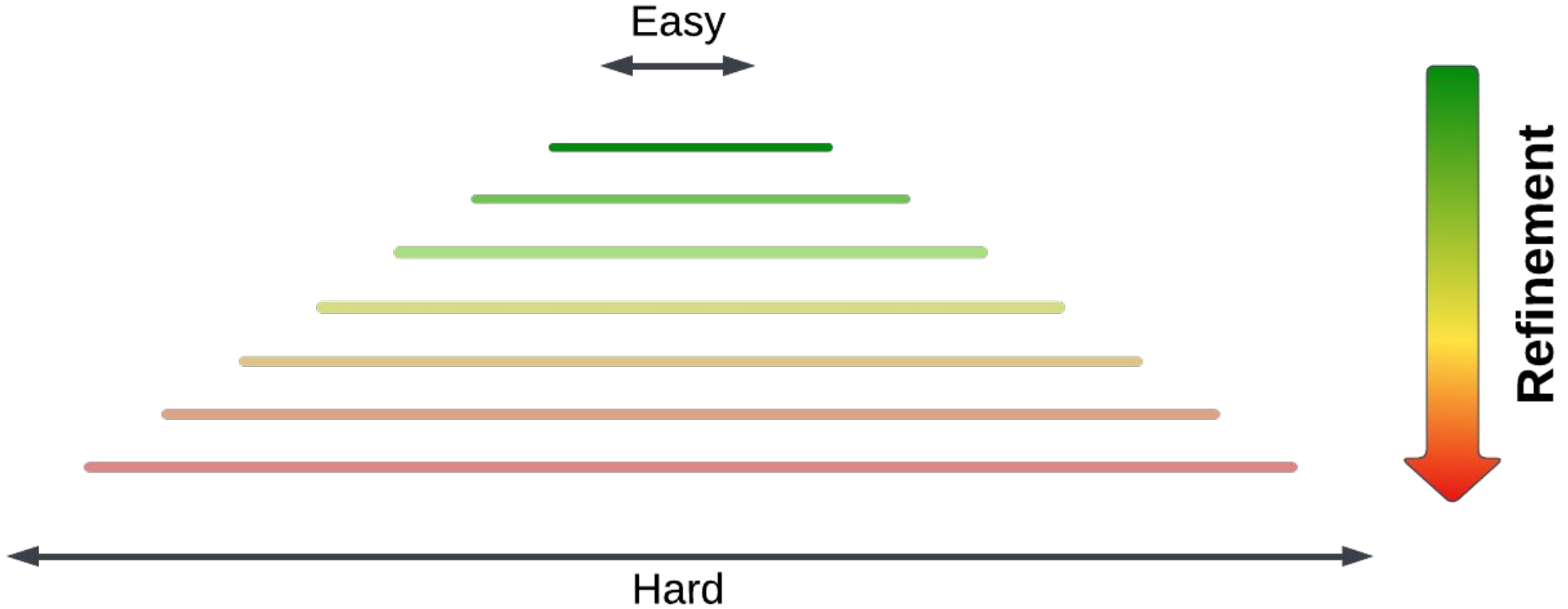
Trivial De-Coring

$$(A \vee B) \wedge (\neg A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee \neg B)$$



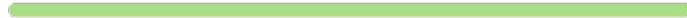
$$(A \vee B \vee C) \wedge (\neg A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee \neg B)$$

Hardening the Output: Core Refinement



Are Hard Cores Guaranteed?

Easy
↔



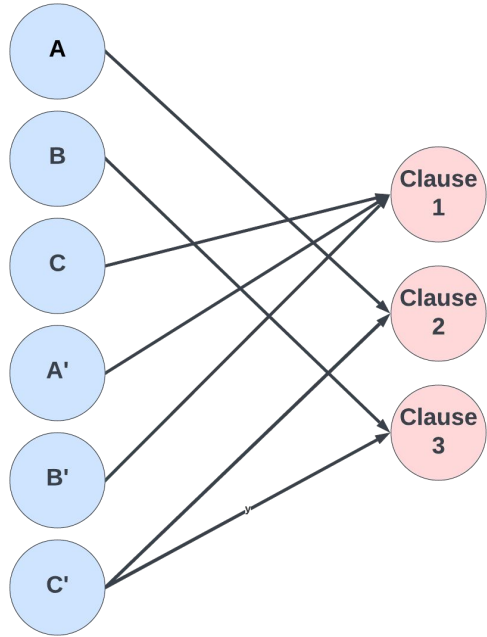
No!

A Problem

Core Refinement requires **repetitive** Core Detection (which is **slow**)

Can we **speed up** Core Detection?

SAT as a Graph - LCG



$$(\bar{A} \vee \bar{B} \vee C) \wedge (A \vee \bar{C}) \wedge (B \vee \bar{C})$$

Core Prediction

Binary Classification over **clause nodes**

True → **Core**

False → **not Core**

Train GNN on CNF-Core pairs

Supervised Training? But we don't have much data...

For a **few** CNFs, do **Core Refinement** using **slow** Core **Detection** algorithm

At **each step** in Core Refinement, **save** CNF, Core pair as **training data**.

If we do **200** Core Refinement **steps**, we **multiply** our training set by **200**.

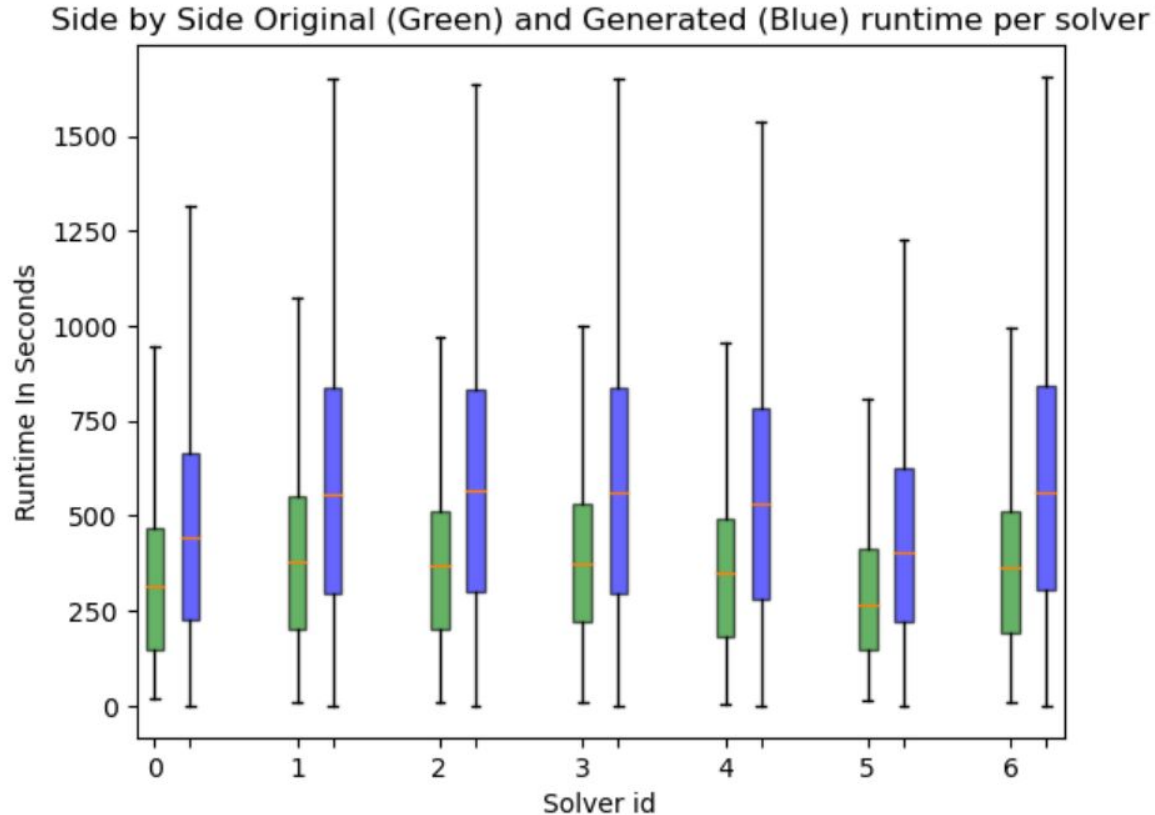
Results

Speedup

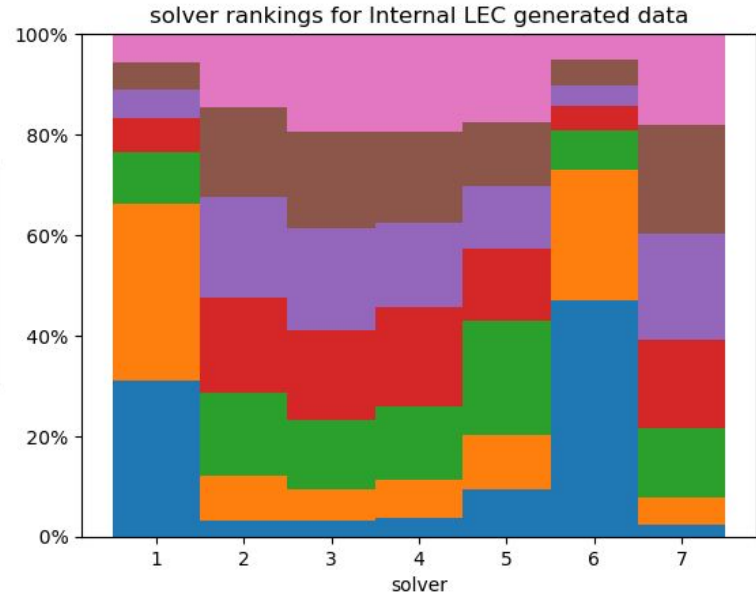
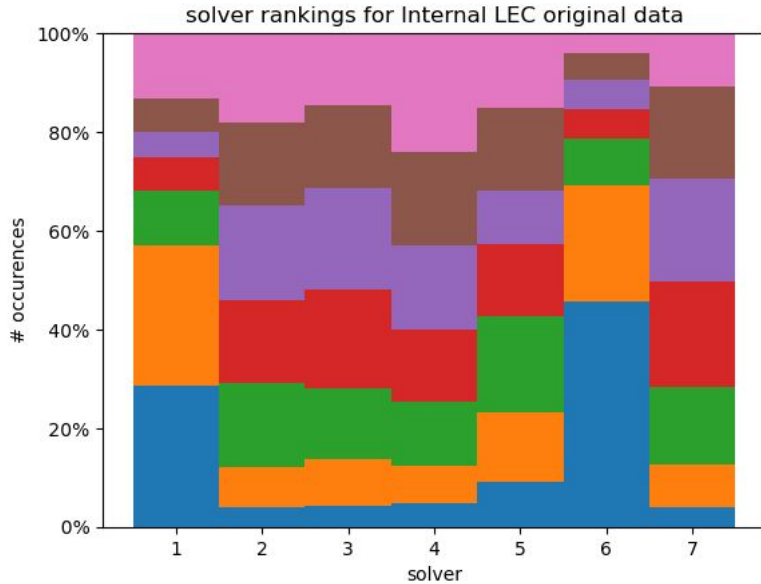
	W2SAT	HardSATGEN	G2MILP	ours
Average Hardness (%)	0	185	0.22	262
Average Time Cost (s)	3.36	6441	4.86	5.46

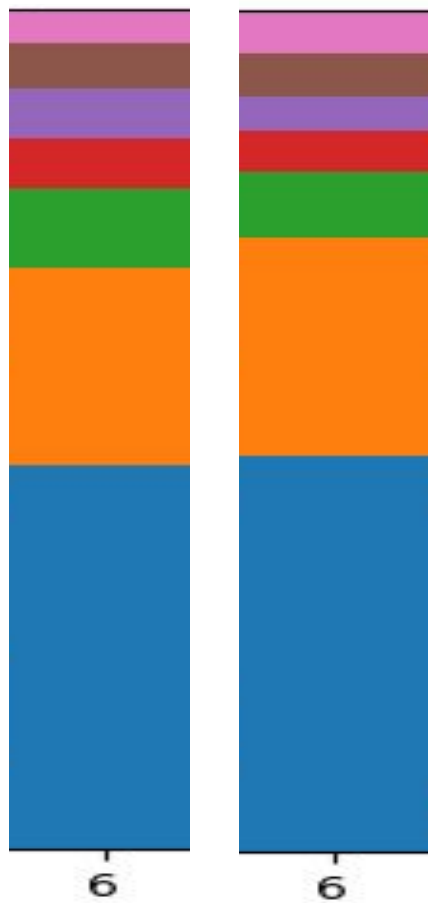
2000x speedup!

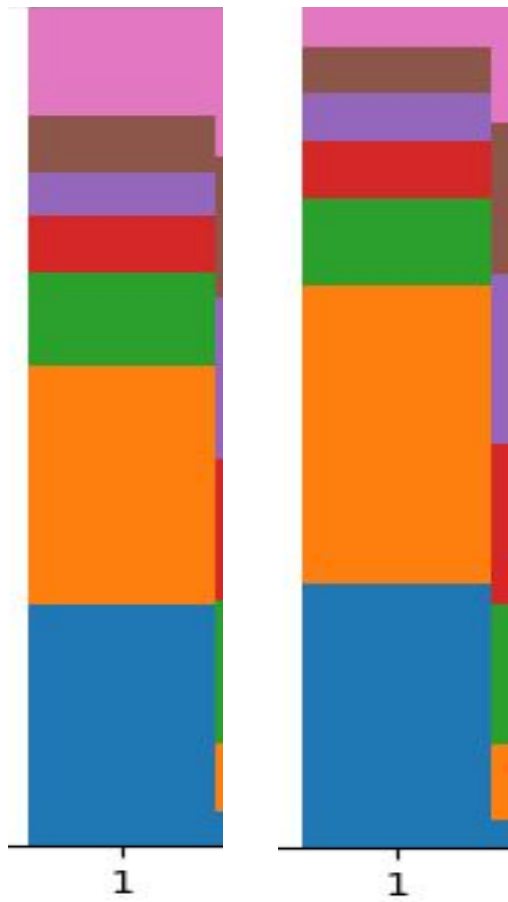
Visualizing Hardness measured by solve-time



The Relative Positions are Similar... Rankings?







Downstream Task

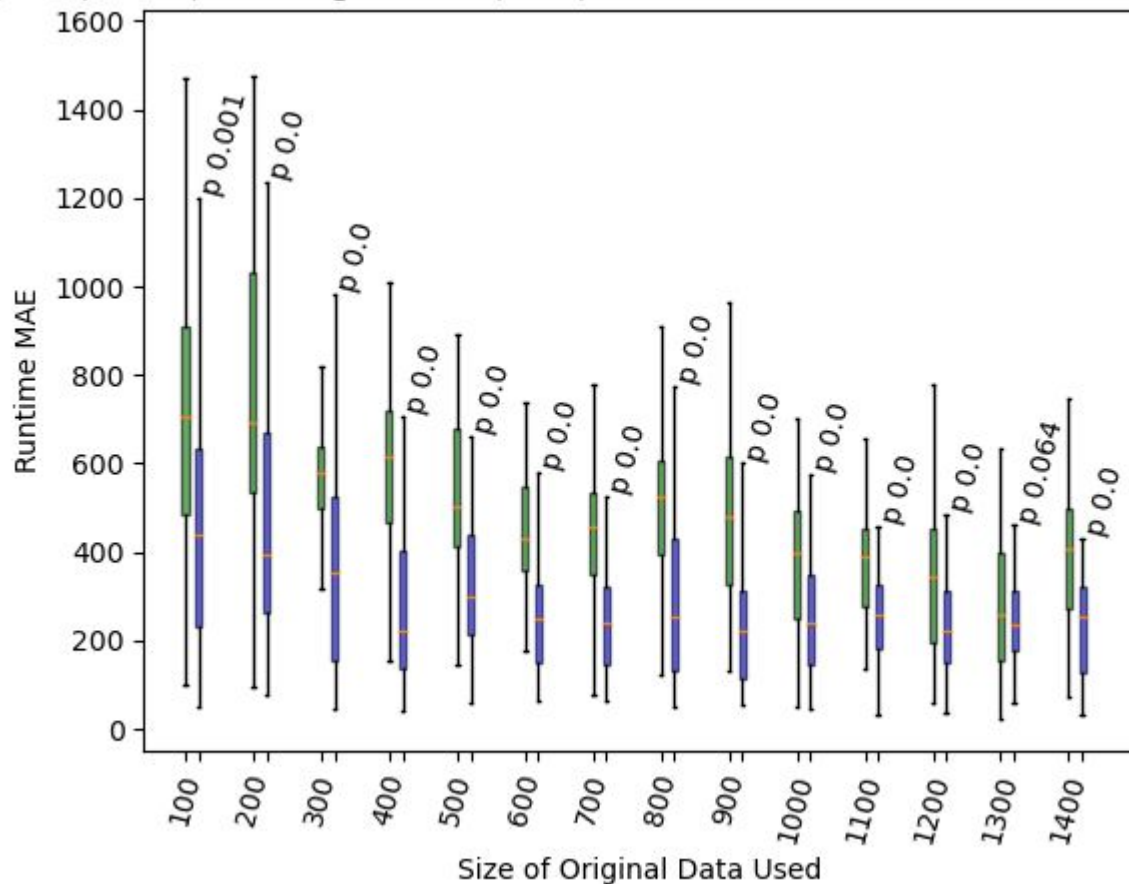
Can we train a **solve-time** prediction model using **generated** data?

Train an **mlp** on **original** data, test on **original data**

Train an **mlp** on **original + synthetic** data, test on **original data**

Performance metric: MAE

Original (Green) and Augmented (Blue) Predicted Runtime MAE on Internal LEC Data



-Early Stopping

-3 layer MLP

-commonly used
hand-crafted features

-p: wilcoxon p value for
original performance <
combined performance

