Problem Partitioning via Proof Prefixes

Zachary Battleman, Joseph E. Reeves, and Marijn J. H. Heule

Carnegie Mellon University

SAT 2025

SAT's Mathematical Success Stories

SAT has had monumental success in resolving open math problems

- Boolean Pythagorean Triples (2016)
- Schur Number 5 (2018)
- Keller's Conjecture (2020)
- Packing Chr. Number of the Plane (2023)
- The Empty Hexagon Problem (2024)

SAT's Mathematical Success Stories

SAT has had monumental success in resolving open math problems

- Boolean Pythagorean Triples (2016)
- Schur Number 5 (2018)
- Keller's Conjecture (2020)
- Packing Chr. Number of the Plane (2023)
- The Empty Hexagon Problem (2024)

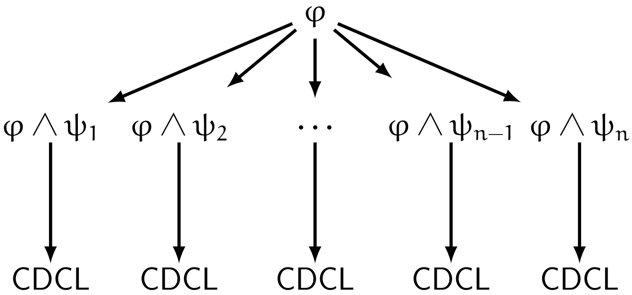
All of these would be impossible without massive parallelism via Cube and Conquer **(CNC)**.

Cube and Conquer

Cube and Conquer (CnC) is a technique for solving SAT formulas in parallel.

- Boolean formula φ
- A partition of conjunctions (*cubes*) $\{\psi_i\}$ such that $\psi := \bigvee_i \psi_i$ is a tautology

$$\varphi \iff \varphi \wedge \psi \iff \bigvee_i \varphi \wedge \psi_i$$

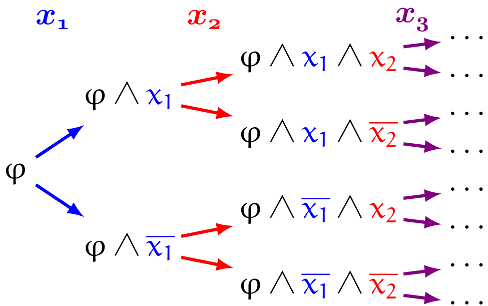


Cube and Conquer

Cube and Conquer (CnC) is a technique for solving SAT formulas in parallel.

- Boolean formula φ
- A partition of conjunctions (*cubes*) $\{\psi_i\}$ such that $\psi := \bigvee_i \psi_i$ is a tautology

$$\psi = \{\ \sigma_1 x_1 \wedge \ldots \wedge \sigma_n x_n \mid \sigma_i \in \{-1, 1\}\}$$



SAT's Mathematical Success Stories, Revisited

- Boolean Pythagorean Triples (2016)
- Schur Number 5 (2018)
- Keller's Conjecture (2020)
- Packing Chr. Number of the Plane (2023)
- The Empty Hexagon Problem (2024)

SAT's Mathematical Success Stories, Revisited

- Boolean Pythagorean Triples (2016) Automatic Partition
- Schur Number 5 (2018) Automatic Partition
- Keller's Conjecture (2020) Manual Partition
- Packing Chr. Number of the Plane (2023) Manual Partition
- The Empty Hexagon Problem (2024) Manual Partition

Manual Partitions Limit Usability

Requirements for making a manual partition

Manual Partitions Limit Usability

Requirements for making a manual partition

- Expert knowledge of SAT solvers

Manual Partitions Limit Usability

Requirements for making a manual partition

- Expert knowledge of SAT solvers
- Expert knowledge of the problem

Manual Partitions Limit Usability

Requirements for making a manual partition

- Expert knowledge of SAT solvers
- Expert knowledge of the problem
- Lots of manual experiments

Manual Partitions Limit Usability

Requirements for making a manual partition

- Expert knowledge of SAT solvers
- Expert knowledge of the problem
- Lots of manual experiments

"For some problems, including Keller's Conjecture and The Empty Hexagon problem, I spent as much time on the manual split as on the encoding."
- Marijn Heule

Manual Partitions Limit Usability

Manual Partitions Limit Usability

How can we come up with better automatic partitions?

The Short Answer

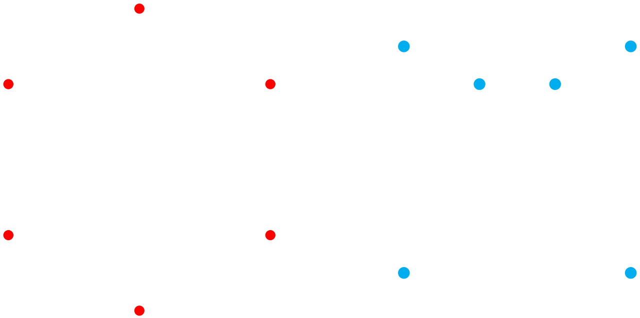
formula	baseline	March (SoTA)			Proofix (Ours)		
	1 core	1 core	32 core	Pre.	1 core	32 core	Pre.
max10	$6,282$	$1,939$	920	0	$7,645$	238	29
cross13	$> 80,000$	?	$> 10,000$	43	$64,610$	$2,206$	71
$\mu_5(13)$	$2,317$	$2,526$	181	8	$1,367$	84	80

- Our Tool (Proofix) outperforms March (SoTA) on tested combinatorial problem.

Motivating Example

Minimizing Convex Pentagons in the Plane

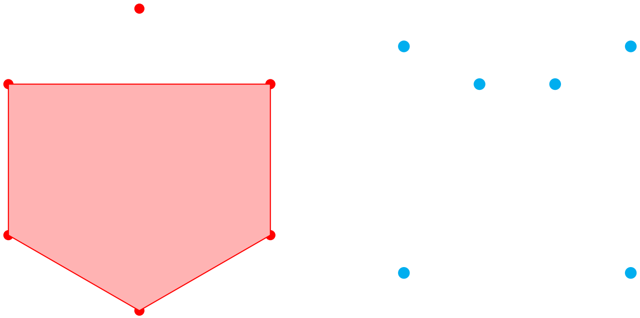
Subercaseaux et al. used SAT to compute new bounds on $\mu_5(n)$, the minimum number of convex pentagons in the plane induced by n points in general position.



Motivating Example

Minimizing Convex Pentagons in the Plane

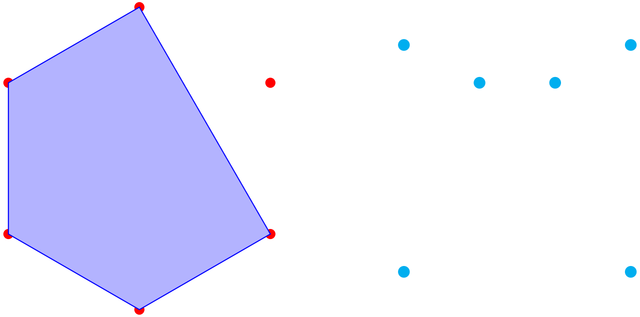
Subercaseaux et al. used SAT to compute new bounds on $\mu_5(n)$, the minimum number of convex pentagons in the plane induced by n points in general position.



Motivating Example

Minimizing Convex Pentagons in the Plane

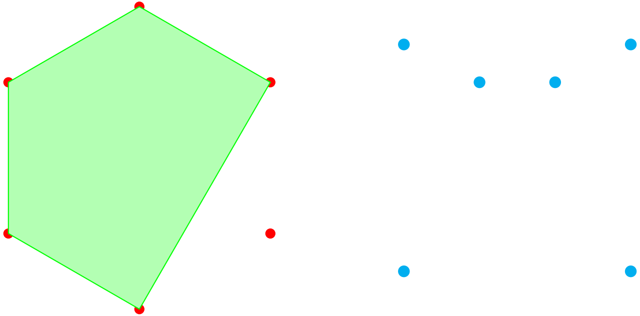
Subercaseaux et al. used SAT to compute new bounds on $\mu_5(n)$, the minimum number of convex pentagons in the plane induced by n points in general position.



Motivating Example

Minimizing Convex Pentagons in the Plane

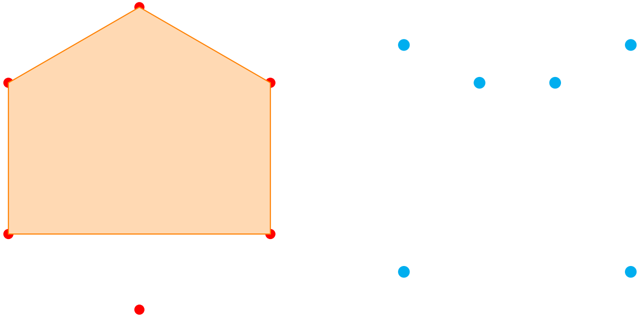
Subercaseaux et al. used SAT to compute new bounds on $\mu_5(n)$, the minimum number of convex pentagons in the plane induced by n points in general position.



Motivating Example

Minimizing Convex Pentagons in the Plane

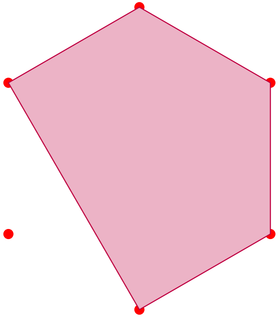
Subercaseaux et al. used SAT to compute new bounds on $\mu_5(n)$, the minimum number of convex pentagons in the plane induced by n points in general position.

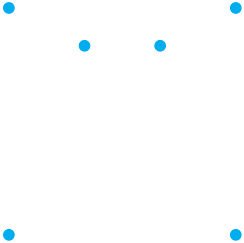


Motivating Example

Minimizing Convex Pentagons in the Plane

Subercaseaux et al. used SAT to compute new bounds on $\mu_5(n)$, the minimum number of convex pentagons in the plane induced by n points in general position.

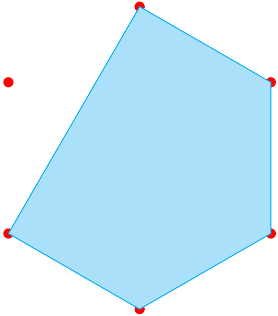


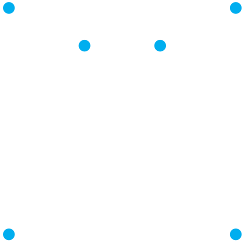


Motivating Example

Minimizing Convex Pentagons in the Plane

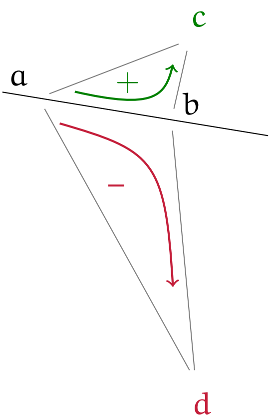
Subercaseaux et al. used SAT to compute new bounds on $\mu_5(n)$, the minimum number of convex pentagons in the plane induced by n points in general position.





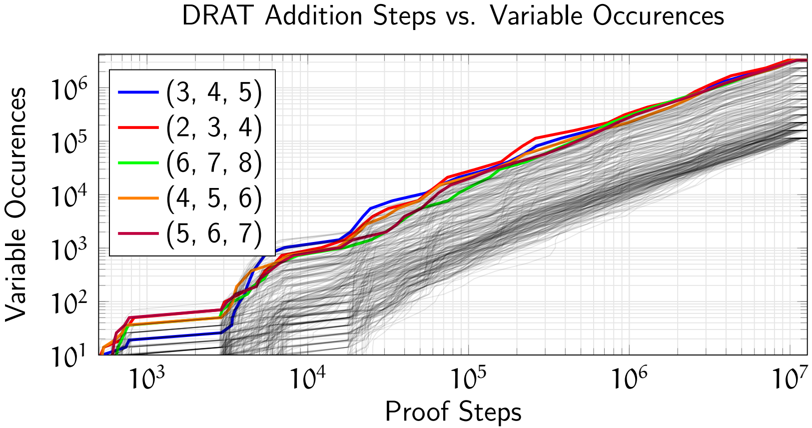
Motivating Example

- $\sigma_{i,j,k}$: whether the arc formed by points i, j, k are concave or convex.
- Best splitting variables of the form $\sigma_{i,i+1,i+2}$ near the "center" of the drawing.
  - e.g. With 13 points, optimal variable is $\sigma_{5,6,7}$

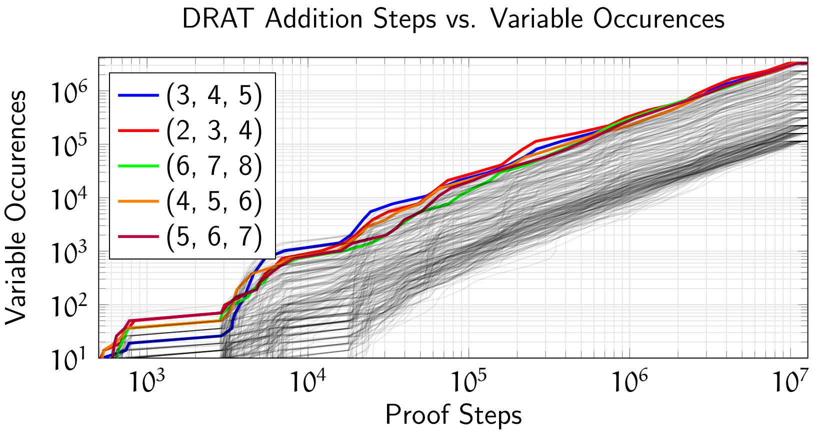


Motivating Example

- Best variables of the form $\sigma_{i,i+1,i+2}$ near the "center" of the drawing.
  - e.g. With 13 points, optimal variable is $\sigma_{5,6,7}$

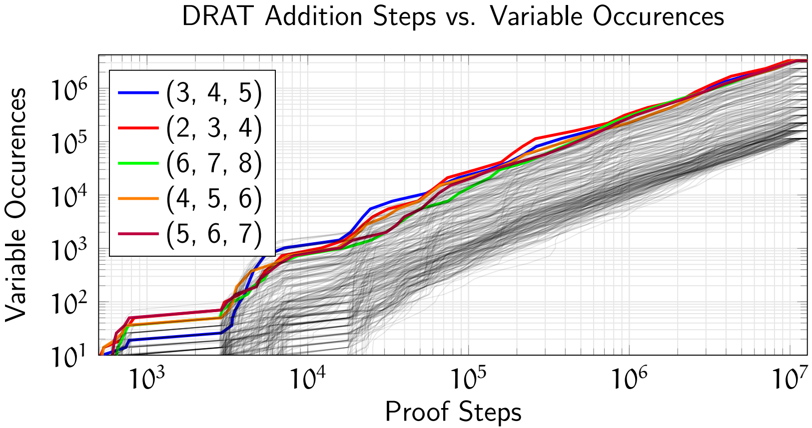


Some Observations



;

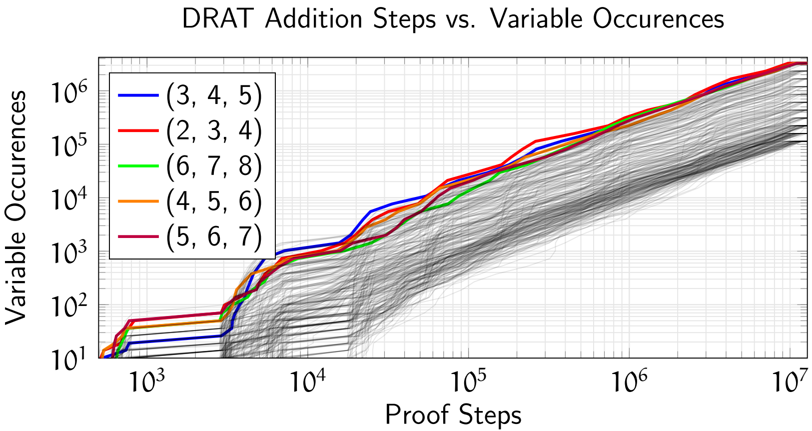
Some Observations



;

1. The best splitting variables rise to the top.

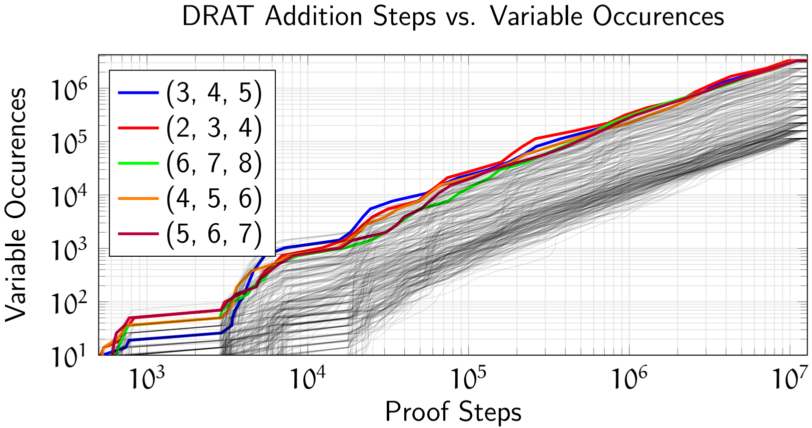
Some Observations



;

1. The best splitting variables rise to the top.
2. When variables rise to the top, they tend to stay there.

Some Observations



;

1. The best splitting variables rise to the top.
2. When variables rise to the top, they tend to stay there.
3. The best variables rise to the top very quickly.

A First Pass at Using Proofs for Splitting

$$\varphi$$

↓

Partial CDCL

↓

Proof Analysis

↓

$$\{x_1, \ldots, x_k\}$$

↓

$$\psi := \{\sigma_1 x_1 \wedge \ldots \wedge \sigma_n x_n \mid \sigma_i \in \{-1, 1\}\}$$

A First Pass at Using Proofs for Splitting

$$\varphi$$
$$\downarrow$$
Partial CDCL
$$\downarrow$$
Proof Analysis
$$\downarrow$$
$$\{x_1, \ldots, x_k\}$$
$$\downarrow$$
$$\psi := \{\sigma_1 x_1 \wedge \ldots \wedge \sigma_n x_n \mid \sigma_i \in \{-1, 1\}\}$$

This doesn't work well

- Lower ranked variables do not do as well in general
- Two similar variables can be ranked highly so splitting on both is counter-productive.

A Second Attempt

φ

A Second Attempt

$$\begin{tikzcd} & \varphi \wedge x_1 \\ \varphi \arrow[ur, "x_1"] \arrow[dr] & \\ & \varphi \wedge \overline{x_1} \end{tikzcd}$$

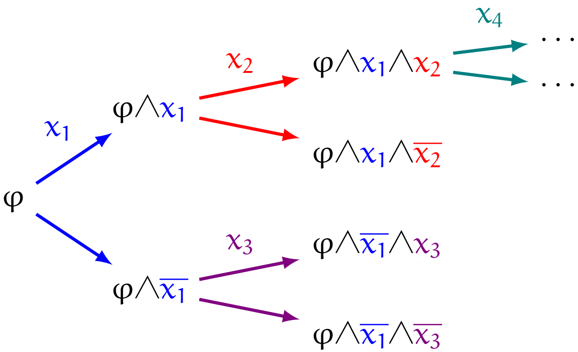
A Second Attempt

$$\begin{tikzcd}
 & & \varphi \wedge x_1 \wedge x_2 \\
 & \varphi \wedge x_1 \arrow[ur, "x_2"] \arrow[dr] & \\
 & & \varphi \wedge x_1 \wedge \overline{x_2} \\
\varphi \arrow[uur, "x_1"] \arrow[dr] & & \\
 & \varphi \wedge \overline{x_1} &
\end{tikzcd}$$

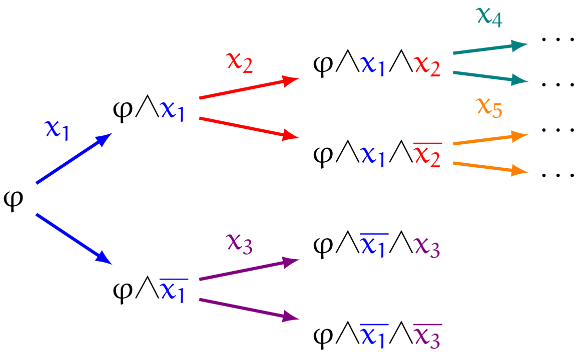
A Second Attempt

$$\begin{tikzcd}
 & & \varphi \wedge x_1 \wedge x_2 \\
 & \varphi \wedge x_1 \arrow[ur, "x_2"] \arrow[dr] & \\
 & & \varphi \wedge x_1 \wedge \overline{x_2} \\
\varphi \arrow[uur, "x_1"] \arrow[ddr] & & \\
 & & \varphi \wedge \overline{x_1} \wedge x_3 \\
 & \varphi \wedge \overline{x_1} \arrow[ur, "x_3"] \arrow[dr] & \\
 & & \varphi \wedge \overline{x_1} \wedge \overline{x_3}
\end{tikzcd}$$

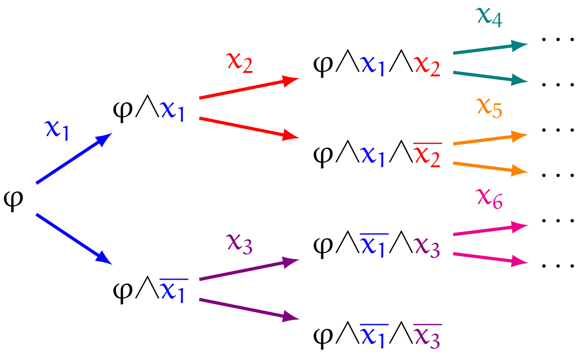
A Second Attempt



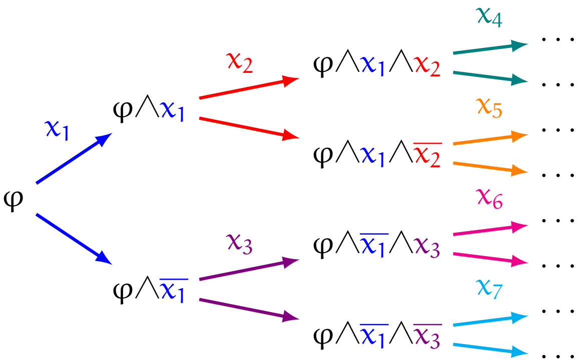
A Second Attempt



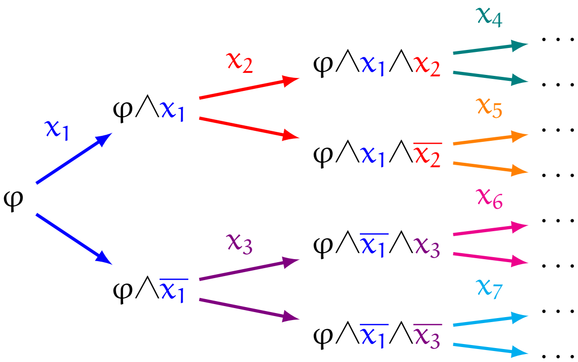
A Second Attempt



A Second Attempt

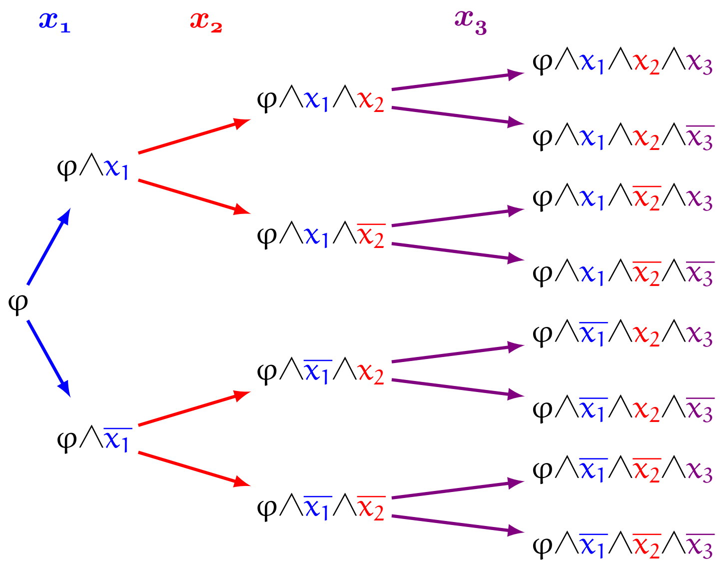


A Second Attempt



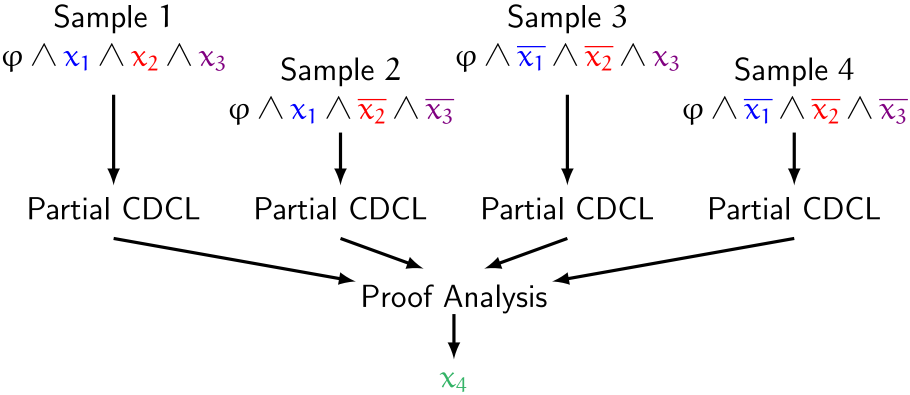
- For cubes of size n, need $2^n - 1$ proof prefixes
- **Too expensive!**

A Middle-Ground Approach

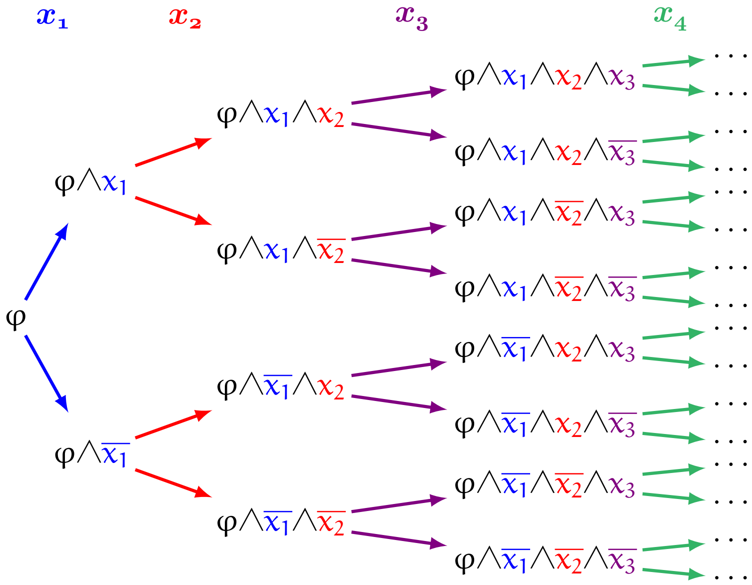


A Middle-Ground Approach

$S = \{x_1, x_2, x_3\}$



A Middle-Ground Approach



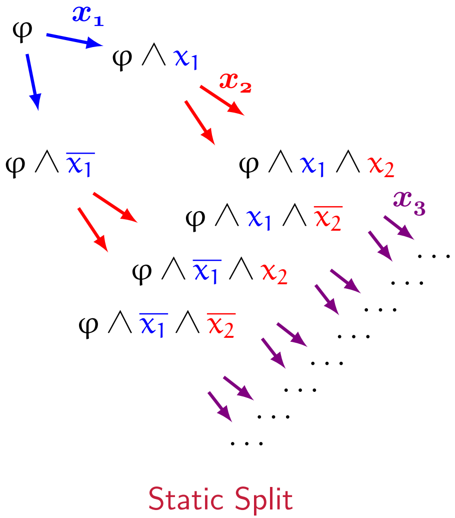
Proofix

This third approach was developed into a tool called *Proofix*.

- User-Friendly
- Fairly little computational overhead
- Sits on top of any proof-producing SAT solver

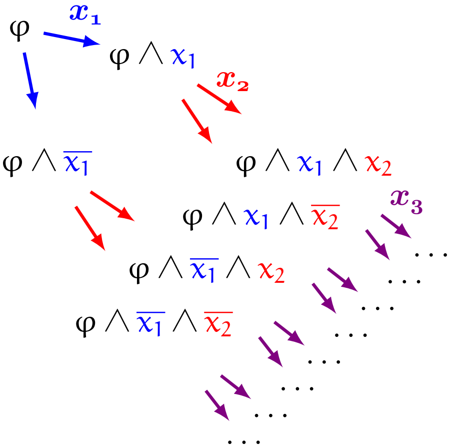
```
$ python3 proofix.py
    --cnf <cnf>
    --cube-size <n>
    --cutoff <c>
    --log <log>
```

Static vs. Dynamic Splits



Benefits of a Static Split

Static splits remedy the opacity of large computer proofs:

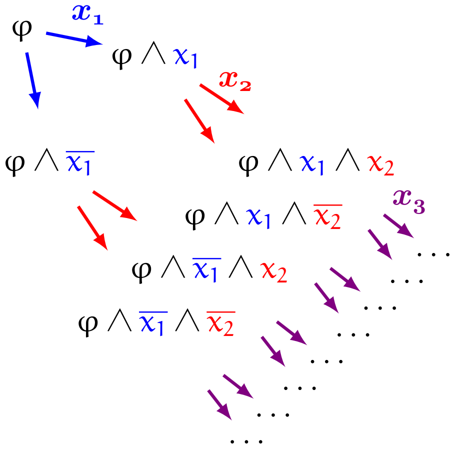


Static Split

Benefits of a Static Split

Static splits remedy the opacity of large computer proofs:

- Demystifying solver reasoning.

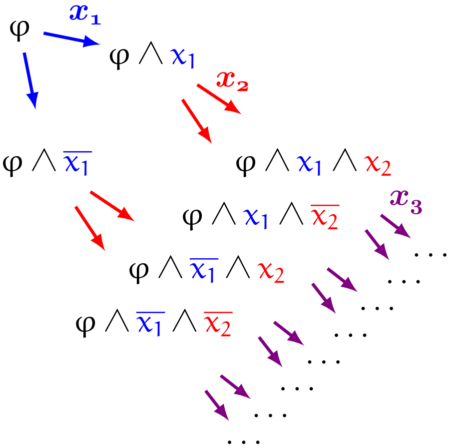


Static Split

Benefits of a Static Split

Static splits remedy the opacity of large computer proofs:

- Demystifying solver reasoning.
- Pinpointing crucial variables.

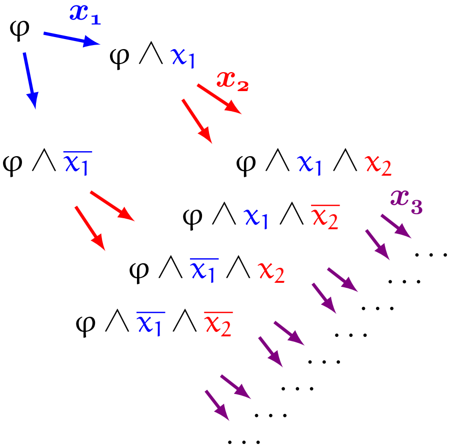


Static Split

Benefits of a Static Split

Static splits remedy the opacity of large computer proofs:

- Demystifying solver reasoning.
- Pinpointing crucial variables.
- Facilitating generalization.



Static Split

Cardinality-Based Splitting

A cardinality constraint compares the sum over a set of data variables to a bound:

$$x_1 + x_2 + x_3 + \overline{x}_4 \leq 2$$

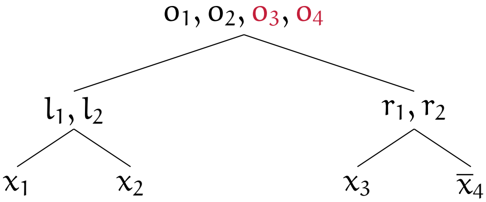
The totalizer encoding uses auxiliary variables to count the sums of data variables hierarchically

Cardinality-Based Splitting

A cardinality constraint compares the sum over a set of data variables to a bound:

$$x_1 + x_2 + x_3 + \overline{x}_4 \leq 2$$

The totalizer encoding uses auxiliary variables to count the sums of data variables hierarchically



Splitting the totalizer

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \leq 5$$

$$o_1, o_2, \ldots, o_8$$

$$l_1, l_2, l_3, l_4 \qquad r_1, r_2, r_3, r_4$$

$$X^{1,2} \qquad X^{3,4} \qquad X^{5,6} \qquad X^{7,8}$$

- Pick variables based on semantic meaning

Splitting the totalizer

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \leq 5$$

$$o_1, o_2, \ldots, o_8$$

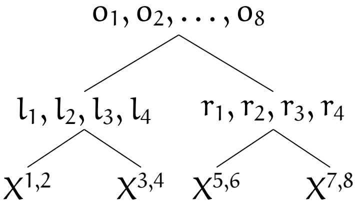
$$l_1, l_2, l_3, l_4 \qquad r_1, r_2, r_3, r_4$$

$$X^{1,2} \qquad X^{3,4} \qquad X^{5,6} \qquad X^{7,8}$$

- Pick variables based on semantic meaning
- For example, l_2 is a good choice:
  - $l_2 = T$, at least 2 from x_1, x_2, x_3, x_4.
  - $l_2 = F$, at most 1 from x_1, x_2, x_3, x_4.

Splitting the totalizer

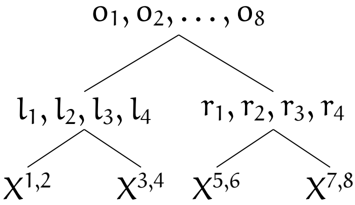
$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \leq 5$$



- Pick variables based on semantic meaning
- For example, l_2 is a good choice:
  - $l_2 = T$, at least 2 from x_1, x_2, x_3, x_4.
  - $l_2 = F$, at most 1 from x_1, x_2, x_3, x_4.
- On the other hand, r_4 is a bad choice:
  - $r_4 = T$, all of x_5, x_6, x_7, x_8.
  - $r_4 = F$, at most 3 from x_5, x_6, x_7, x_8.

Splitting the totalizer

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \leq 5$$



- Pick variables based on semantic meaning
- For example, l_2 is a good choice:
  - $l_2 = T$, at least 2 from x_1, x_2, x_3, x_4.
  - $l_2 = F$, at most 1 from x_1, x_2, x_3, x_4.
- On the other hand, r_4 is a bad choice:
  - $r_4 = T$, all of x_5, x_6, x_7, x_8.
  - $r_4 = F$, at most 3 from x_5, x_6, x_7, x_8.
- Assumes uniform "activity" of positive literals.

MaxSAT competition Formulas

formula	baseline	March			Proofix			Totalizer Splitting	
	1 core	1 core	32 core	Pre.	1 core	32 core	Pre.	1 core	32 core
judge	3, 654	4, 893	3, 162	219	3, 437	2, 447	19	7,851	4,289
mbd	2, 170	2, 914	313	287	2, 512	409	37	3,784	290
optic	1, 236	908	195	23	708	22	45	1,150	135
uaq	2, 520	1, 408	453	4	970	62	43	1,960	129
mindset	2, 162	18, 018	1, 372	357	16, 375	2, 252	42	19,002	1,164

- Proofix is 21% better than March on average.
- Totalizer splitting is 18% better than March on average.

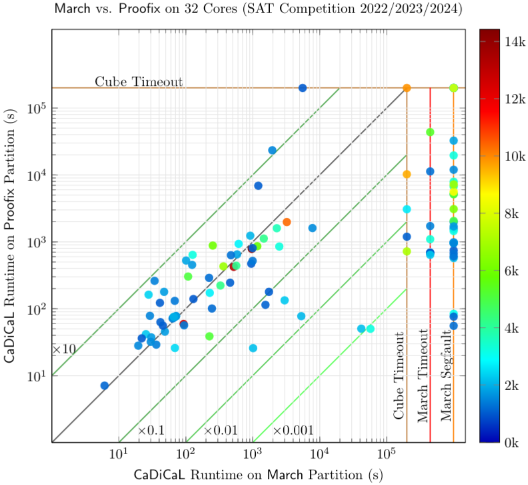
Stability under Search Parameters

Formula	# "Good"	# Variables	Prefix Length 10^3	10^4	10^5	5×10^5	10^6
$\chi_\rho(\mathbb{Z}^2)$	63	$7,669$	0/15	0/15	12/15	12/15	11/15
$\mu_5(15)$	13	$58,826$	7/15	5/15	4/15	3/13	4/15
7gon-6hole	20	$28,878$	2/15	8/15	13/15	12/15	12/15

- Proofix is able to very quickly identify "good" splitting variables.
- On many problems, once a "good" variable is identified, it stays "good" on deeper cube generations.

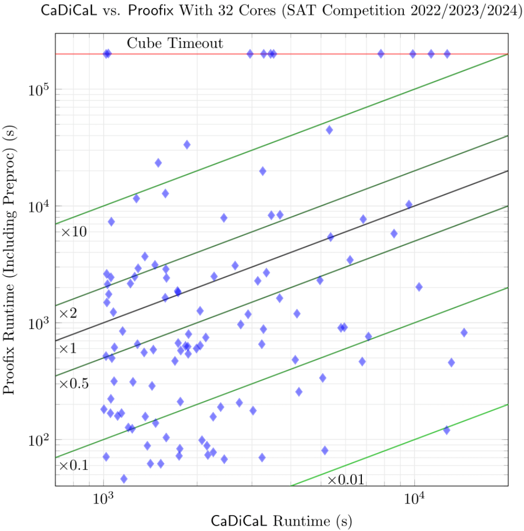
SAT competition Problems

- Proofix outperforms March on 61% of problems.
- Up to 1000× improvement!



SAT competition Problems

- Proofix outperforms CaDiCaL on 63% of problems (including pre-processing).
- Up to 100× improvement!



Difficult Combinatorial Problems

formula	baseline	March			Proofix			Totalizer Splitting	
	1 core	1 core	32 core	Pre.	1 core	32 core	Pre.	1 core	32 core
max10	$6,282$	$1,939$	920	0	$7,645$	238	29	2,498	269
cross13	$> 80,000$	?	$> 10,000$	43	$64,610$	$2,206$	71	77,664	3,125
$\mu_5(13)$	$2,317$	$2,526$	181	8	$1,367$	84	80	2,355	543

- Proofix outperforms March universally; by at least 68% on average.
- Totalizer splitting is less effective, with only marginal improvements on $2/3$ problems.

Early Uses of Proofix

People are already using Proofix for solving their own difficult combinatorial problems!

- Crossing numbers of $\mathrm{R}(5,5)$-good cycle graphs.
- Classifying $\mathrm{R}(5,5)$-good strongly regular graphs.
- Sharper asymptotics on Norin's conjecture.
- Smaller proofs of the Keller graph max clique problem.
- Disproving $\mathrm{R}(4,7)$-good graphs with assumed structure.

Early Uses of Proofix

People are already using Proofix for solving their own difficult combinatorial problems!

- Crossing numbers of $\mathrm{R}(5,5)$-good cycle graphs.
- Classifying $\mathrm{R}(5,5)$-good strongly regular graphs.
- Sharper asymptotics on Norin's conjecture.
- Smaller proofs of the Keller graph max clique problem.
- Disproving $\mathrm{R}(4,7)$-good graphs with assumed structure.
  - Just over 13.4 *million* CPU seconds across all cubes.
  - Could not be solved, or even attempted, by existing tools.

Future Work

- Understand better why this tool works.
- Find better extraction heuristics.
- Explore other proof formats.
- Solve more problems!

Thank you! :D

Link to full paper