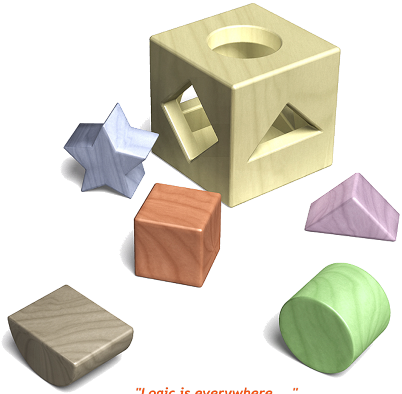
The boolean Pythagorean Triples problem

Tobias John, Aldo Kurmeta, Patrick Wienhöft
International Center for Computational Logic
Technische Universität Dresden
Germany

- Introduction
- History of the problem
- The framework
- Encoding
- Transformation
- Splitting
- Validation for each step



"Logic is everywhere ..."

Introduction

- **The boolean Pythagorean Triples problem has been a longstanding open problem in Ramsey Theory**

Introduction

- **The boolean Pythagorean Triples problem has been a longstanding open problem in Ramsey Theory**
- **Can the set $\mathbb{N}$ = {1, 2, 3, . . . } be divided in two parts such that no part contains a triple (a, b, c) with $a^2 + b^2 = c^2$**

Example

- **Set of Integers: $\{1, ..., 15\}$**

Example

- **Set of Integers:** $\{1, \ldots, 15\}$
- **Triples:**

$$3^2 + 4^2 = 5^2$$
$$6^2 + 8^2 = 10^2$$
$$9^2 + 12^2 = 15^2$$
$$5^2 + 12^2 = 13^2$$

Example

- **Set of Integers:** $\{1, \ldots, 15\}$
- **Triples:**

$$3^2 + 4^2 = 5^2$$
$$6^2 + 8^2 = 10^2$$
$$9^2 + 12^2 = 15^2$$
$$5^2 + 12^2 = 13^2$$

- **Partition:** $\{1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 14\}, \{5, 10, 13, 15\}$

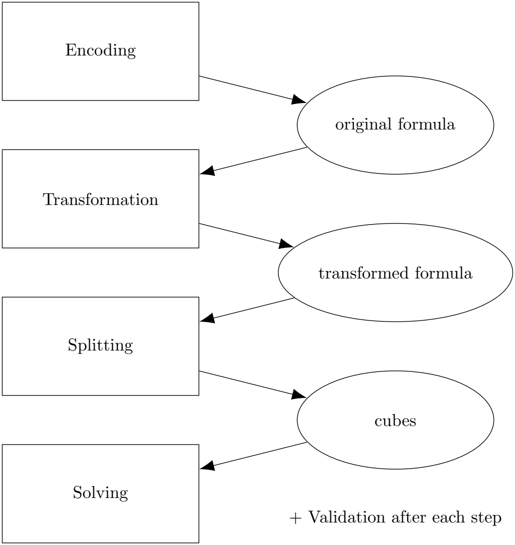
- **The set $\{1, \ldots, 7824\}$ can be partitioned into two parts, while this is impossible for $\{1, \ldots, 7825\}$**

- **The set $\{1, \ldots, 7824\}$ can be partitioned into two parts, while this is impossible for $\{1, \ldots, 7825\}$**
- **We prove this theorem by considering two SAT problems:**

- **The set $\{1, \ldots, 7824\}$ can be partitioned into two parts, while this is impossible for $\{1, \ldots, 7825\}$**
- **We prove this theorem by considering two SAT problems:**
  1. **showing that $\{1, \ldots, 7824\}$ can be partitioned in two different parts.**

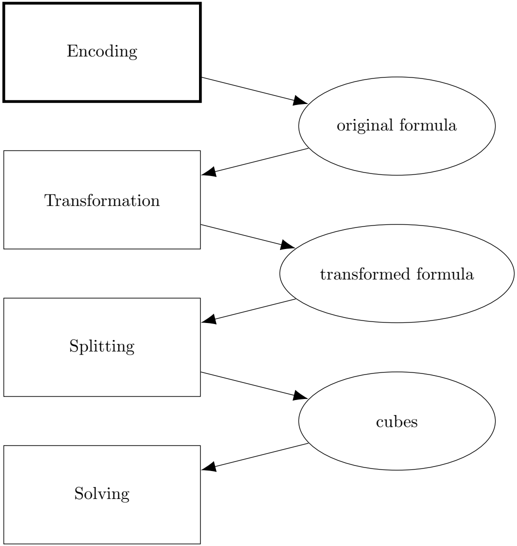
- The set $\{1, ..., 7824\}$ can be partitioned into two parts, while this is impossible for $\{1, ..., 7825\}$
- We prove this theorem by considering two SAT problems:
  1. showing that $\{1, ..., 7824\}$ can be partitioned in two different parts.
  2. showing that any partition of $\{1, ..., 7825\}$ contains a Pythagorean triple.

Framework



+ Validation after each step

Encoding


Encoding
original formula
Transformation
transformed formula
Splitting
cubes
Solving

Encoding - Intuition

Idea:

- **one variable for each number**

Encoding - Intuition

Idea:

- **one variable for each number**
- **interpretation gives partition**

Encoding - Intuition

Idea:

- **one variable for each number**
- **interpretation gives partition**
- **one constraint clause for each Pythagorean triple**

Encoding

Binary Pythagorean triple problem with n numbers

Encoding

Binary Pythagorean triple problem with n numbers

Set of variables $V = \{p_k \mid 1 \leq k \leq n\}$

Encoding

Binary Pythagorean triple problem with n numbers

Set of variables $V = \{p_k \mid 1 \leq k \leq n\}$

Constraint for non-equality: $\mathit{NotEqual}(x, y, z) = (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$

Encoding

Binary Pythagorean triple problem with n numbers

Set of variables $V = \{p_k \mid 1 \leq k \leq n\}$

Constraint for non-equality: $NotEqual(x, y, z) = (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$

Constraint for all Pythagorean triples: $F = \bigwedge_{x^2+y^2=z^2} NotEqual(p_x, p_y, p_z)$

Encoding

Binary Pythagorean triple problem with n numbers

Set of variables $V = \{p_k \mid 1 \leq k \leq n\}$

Constraint for non-equality: $NotEqual(x, y, z) = (x \lor y \lor z) \land (\neg x \lor \neg y \lor \neg z)$

Constraint for all Pythagorean triples: $F = \bigwedge_{x^2+y^2=z^2} NotEqual(p_x, p_y, p_z)$

For an interpretation $I \subseteq V$ with $I \models F$, the resulting partition is:

- $P_1 = \{x \mid p_x \in I\}$
- $P_2 = \{x \mid p_x \notin I\}$

Encoding - Example

As in beginning example, $n = 15$

Encoding - Example

As in beginning example, $n = 15$

$V = \{p_1, p_2, \ldots, p_{15}\}$

Encoding - Example

As in beginning example, $n = 15$

$V = \{p_1, p_2, \ldots, p_{15}\}$

$$
\begin{aligned}
F \quad = \quad & (p_3 \vee p_4 \vee p_5) \wedge (\neg p_3 \vee \neg p_4 \vee \neg p_5) \\
\wedge \quad & (p_6 \vee p_8 \vee p_{10}) \wedge (\neg p_6 \vee \neg p_8 \vee \neg p_{10}) \\
\wedge \quad & (p_9 \vee p_{12} \vee p_{15}) \wedge (\neg p_9 \vee \neg p_{12} \vee \neg p_{15}) \\
\wedge \quad & (p_5 \vee p_{12} \vee p_{13}) \wedge (\neg p_5 \vee \neg p_{12} \vee \neg p_{13})
\end{aligned}
$$

Encoding - Example

As in beginning example, $n = 15$

$V = \{p_1, p_2, \ldots, p_{15}\}$

$$
\begin{aligned}
F = \quad & (p_3 \lor p_4 \lor p_5) \land (\neg p_3 \lor \neg p_4 \lor \neg p_5) \\
\land \quad & (p_6 \lor p_8 \lor p_{10}) \land (\neg p_6 \lor \neg p_8 \lor \neg p_{10}) \\
\land \quad & (p_9 \lor p_{12} \lor p_{15}) \land (\neg p_9 \lor \neg p_{12} \lor \neg p_{15}) \\
\land \quad & (p_5 \lor p_{12} \lor p_{13}) \land (\neg p_5 \lor \neg p_{12} \lor \neg p_{13})
\end{aligned}
$$

Possible interpretation: $I = \{p_5, p_{10}, p_{13}, p_{15}\}$

Encoding - Example

As in beginning example, $n = 15$

$V = \{p_1, p_2, \ldots, p_{15}\}$

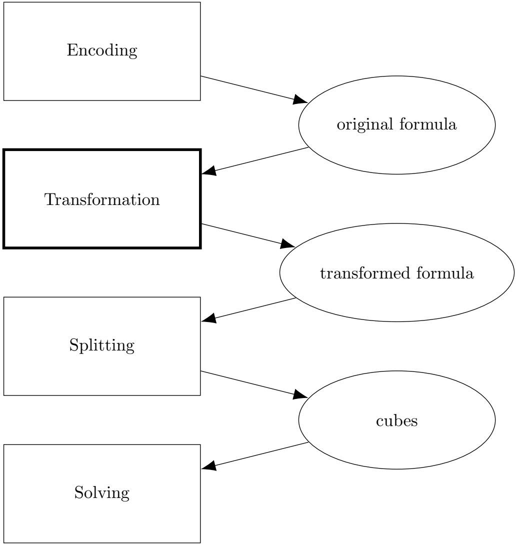
$$\begin{aligned} F = & (p_3 \lor p_4 \lor p_5) \land (\neg p_3 \lor \neg p_4 \lor \neg p_5) \\ \land & (p_6 \lor p_8 \lor p_{10}) \land (\neg p_6 \lor \neg p_8 \lor \neg p_{10}) \\ \land & (p_9 \lor p_{12} \lor p_{15}) \land (\neg p_9 \lor \neg p_{12} \lor \neg p_{15}) \\ \land & (p_5 \lor p_{12} \lor p_{13}) \land (\neg p_5 \lor \neg p_{12} \lor \neg p_{13}) \end{aligned}$$

Possible interpretation: $I = \{p_5, p_{10}, p_{13}, p_{15}\}$

Resulting partition:

- $P_1 = \{1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 14\}$
- $P_2 = \{5, 10, 13, 15\}$

Transformatio



Transformation

Goal: from F, find formula F' which

- **is easier to solve**

Transformation

Goal: from $\boldsymbol{F}$, find formula $\boldsymbol{F'}$ which

- **is easier to solve**
- **preserves satisfiability**

Transformation

Goal: from F, find formula F' which

- **is easier to solve**
- **preserves satisfiability**
- **has models that can be easily transformed into models for F**

Transformation

Goal: from $\boldsymbol{F}$, find formula $\boldsymbol{F'}$ which

- **is easier to solve**
- **preserves satisfiability**
- **has models that can be easily transformed into models for $\boldsymbol{F}$**

Approaches:

- **eliminate some particular clauses**

Transformation

Goal: from $\boldsymbol{F}$, find formula $\boldsymbol{F'}$ which

- **is easier to solve**
- **preserves satisfiability**
- **has models that can be easily transformed into models for *F***

Approaches:

- **eliminate some particular clauses**
- **break partition symmetry**

Clause elimination

$$\begin{aligned} F \quad = \quad & \mathit{NotEqual}(p_3, p_4, p_5) \\ \wedge \quad & \mathit{NotEqual}(p_6, p_9, p_{12}) \\ \wedge \quad & \mathit{NotEqual}(p_9, p_{12}, p_{15}) \\ \wedge \quad & \mathit{NotEqual}(p_5, p_{12}, p_{13}) \end{aligned}$$

Note p_3 only occurs in $\mathit{NotEqual}(p_3, p_4, p_5)$ and thus does not affect any other clauses

Clause elimination

$$
\begin{aligned}
F \quad = \quad & \mathit{NotEqual}(p_3, p_4, p_5) \\
\wedge \quad & \mathit{NotEqual}(p_6, p_9, p_{12}) \\
\wedge \quad & \mathit{NotEqual}(p_9, p_{12}, p_{15}) \\
\wedge \quad & \mathit{NotEqual}(p_5, p_{12}, p_{13})
\end{aligned}
$$

Note p_3 only occurs in $\mathit{NotEqual}(p_3, p_4, p_5)$ and thus does not affect any other clauses

- **if $p_4^I \neq p_5^I$ then $\mathit{NotEqual}(p_3, p_4, p_5)$ is satisfied**

Clause elimination

$$
\begin{aligned}
F \quad = \quad & NotEqual(p_3, p_4, p_5) \\
\wedge \quad & NotEqual(p_6, p_9, p_{12}) \\
\wedge \quad & NotEqual(p_9, p_{12}, p_{15}) \\
\wedge \quad & NotEqual(p_5, p_{12}, p_{13})
\end{aligned}
$$

Note p_3 only occurs in $NotEqual(p_3, p_4, p_5)$ and thus does not affect any other clauses

- **if $p_4^I \neq p_5^I$ then $NotEqual(p_3, p_4, p_5)$ is satisfied**
- **if $p_4^I = p_5^I = \top$ then $p_3^I = \bot$**

Clause elimination

$$
\begin{aligned}
F \quad = \quad & \mathit{NotEqual}(p_3, p_4, p_5) \\
\wedge \quad & \mathit{NotEqual}(p_6, p_9, p_{12}) \\
\wedge \quad & \mathit{NotEqual}(p_9, p_{12}, p_{15}) \\
\wedge \quad & \mathit{NotEqual}(p_5, p_{12}, p_{13})
\end{aligned}
$$

Note p_3 only occurs in $\mathit{NotEqual}(p_3, p_4, p_5)$ and thus does not affect any other clauses

- **if $p_4^I \neq p_5^I$ then $\mathit{NotEqual}(p_3, p_4, p_5)$ is satisfied**
- **if $p_4^I = p_5^I = \top$ then $p_3^I = \bot$**
- **if $p_4^I = p_5^I = \bot$ then $p_3^I = \top$**

Clause elimination

$$
\begin{aligned}
F \quad = \quad & \mathit{NotEqual}(p_3, p_4, p_5) \\
\wedge \quad & \mathit{NotEqual}(p_6, p_9, p_{12}) \\
\wedge \quad & \mathit{NotEqual}(p_9, p_{12}, p_{15}) \\
\wedge \quad & \mathit{NotEqual}(p_5, p_{12}, p_{13})
\end{aligned}
$$

Note p_3 only occurs in $\mathit{NotEqual}(p_3, p_4, p_5)$ and thus does not affect any other clauses

- **if $p_4^I \neq p_5^I$ then $\mathit{NotEqual}(p_3, p_4, p_5)$ is satisfied**
- **if $p_4^I = p_5^I = \top$ then $p_3^I = \bot$**
- **if $p_4^I = p_5^I = \bot$ then $p_3^I = \top$**

$\rightarrow$ clause $\mathit{NotEqual}(p_3, p_4, p_5)$ will not cause conflict

Clause elimination

$$
\begin{aligned}
F \quad = \quad & NotEqual(p_3, p_4, p_5) \\
\wedge \quad & NotEqual(p_6, p_9, p_{12}) \\
\wedge \quad & NotEqual(p_9, p_{12}, p_{15}) \\
\wedge \quad & NotEqual(p_5, p_{12}, p_{13})
\end{aligned}
$$

Note p_3 only occurs in $NotEqual(p_3, p_4, p_5)$ and thus does not affect any other clauses

- **if $p_4^I \neq p_5^I$ then $NotEqual(p_3, p_4, p_5)$ is satisfied**
- **if $p_4^I = p_5^I = \top$ then $p_3^I = \bot$**
- **if $p_4^I = p_5^I = \bot$ then $p_3^I = \top$**

$\rightarrow$ clause $NotEqual(p_3, p_4, p_5)$ will not cause conflict

- **remove $NotEqual(p_3, p_4, p_5)$ from F to obtain F'**

Clause elimination

$$F = NotEqual(p_3, p_4, p_5) \wedge NotEqual(p_6, p_9, p_{12}) \wedge NotEqual(p_9, p_{12}, p_{15}) \wedge NotEqual(p_5, p_{12}, p_{13})$$

Note p_3 only occurs in $NotEqual(p_3, p_4, p_5)$ and thus does not affect any other clauses

- **if $p_4^I \neq p_5^I$ then $NotEqual(p_3, p_4, p_5)$ is satisfied**
- **if $p_4^I = p_5^I = \top$ then $p_3^I = \bot$**
- **if $p_4^I = p_5^I = \bot$ then $p_3^I = \top$**

$\rightarrow$ clause $NotEqual(p_3, p_4, p_5)$ will not cause conflict

- **remove $NotEqual(p_3, p_4, p_5)$ from F to obtain F'**
- **interpretation of p_3 is important but not represented in F'**

Clause elimination

$$\begin{aligned} F \quad = \quad & \mathit{NotEqual}(p_3, p_4, p_5) \\ \wedge \quad & \mathit{NotEqual}(p_6, p_9, p_{12}) \\ \wedge \quad & \mathit{NotEqual}(p_9, p_{12}, p_{15}) \\ \wedge \quad & \mathit{NotEqual}(p_5, p_{12}, p_{13}) \end{aligned}$$

Note p_3 only occurs in $\mathit{NotEqual}(p_3, p_4, p_5)$ and thus does not affect any other clauses

- **if $p_4^I \neq p_5^I$ then $\mathit{NotEqual}(p_3, p_4, p_5)$ is satisfied**
- **if $p_4^I = p_5^I = \top$ then $p_3^I = \bot$**
- **if $p_4^I = p_5^I = \bot$ then $p_3^I = \top$**

$\rightarrow$ clause $\mathit{NotEqual}(p_3, p_4, p_5)$ will not cause conflict

- **remove $\mathit{NotEqual}(p_3, p_4, p_5)$ from F to obtain F'**
- **interpretation of p_3 is important but not represented in F'**
- **$\rightarrow$ remember deleted clauses and modify interpretation of F' accordingly**

Clause elimination - Example

$$
\begin{aligned}
F' \quad = \quad & NotEqual(p_6, p_9, p_{12}) \\
\wedge \quad & NotEqual(p_9, p_{12}, p_{15}) \\
\wedge \quad & NotEqual(p_5, p_{12}, p_{13})
\end{aligned}
$$

Clause elimination - Example

$$
\begin{aligned}
F' \quad = \quad & NotEqual(p_6, p_9, p_{12}) \\
\wedge \quad & NotEqual(p_9, p_{12}, p_{15}) \\
\wedge \quad & NotEqual(p_5, p_{12}, p_{13})
\end{aligned}
$$

Possible Interpretation $I = \{p_6, p_{12}\}$

Clause elimination - Example

$$
\begin{aligned}
F' \quad = \quad & NotEqual(p_6, p_9, p_{12}) \\
\wedge \quad & NotEqual(p_9, p_{12}, p_{15}) \\
\wedge \quad & NotEqual(p_5, p_{12}, p_{13})
\end{aligned}
$$

Possible Interpretation $I = \{p_6, p_{12}\}$

$I \models F'$ but $I \not\models F$

$\rightarrow$ account for deleted constraint $NotEqual(p_3, p_4, p_5)$

Clause elimination - Example

$$
\begin{aligned}
F' \quad = \quad & NotEqual(p_6, p_9, p_{12}) \\
\wedge \quad & NotEqual(p_9, p_{12}, p_{15}) \\
\wedge \quad & NotEqual(p_5, p_{12}, p_{13})
\end{aligned}
$$

Possible Interpretation $I = \{p_6, p_{12}\}$

$I \models F'$ but $I \not\models F$

$\longrightarrow$ account for deleted constraint $NotEqual(p_3, p_4, p_5)$

As $p_4^I = p_5^I = \bot$ we modify $p_3^I = \top$

Clause elimination - Example

$$\begin{aligned} F' \quad = \quad & NotEqual(p_6, p_9, p_{12}) \\ \wedge \quad & NotEqual(p_9, p_{12}, p_{15}) \\ \wedge \quad & NotEqual(p_5, p_{12}, p_{13}) \end{aligned}$$

Possible Interpretation $I = \{p_6, p_{12}\}$

$I \models F'$ but $I \not\models F$

$\rightarrow$ account for deleted constraint $NotEqual(p_3, p_4, p_5)$

As $p_4^I = p_5^I = \bot$ we modify $p_3^I = \top$

$I' = \{p_3, p_6, p_{12}\}$

$I' \models F$

Breaking Symmetry

- **formulas F and F' are symmetric**

Breaking Symmetry

- formulas F and F' are symmetric
- if $I \models F'$ then $V \setminus I \models F'$

Breaking Symmetry

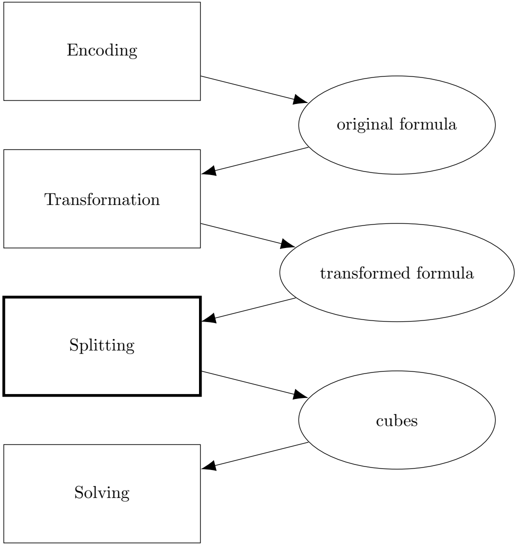
- formulas F and F' are symmetric
- if $I \models F'$ then $V \setminus I \models F'$
- $\rightarrow$ introduce unit clause for variable p_x occurring in F'
- $F'' = F' \wedge p_x$

Breaking Symmetry

- **formulas F and F' are symmetric**
- **if $I \models F'$ then $V \setminus I \models F'$**
- **$\rightarrow$ introduce unit clause for variable p_x occurring in F'**
- **$F'' = F' \wedge p_x$**

Note that every model for F'' is a model for F' and the transformation is satisfiability preserving

Splitting


Encoding
original formula
Transformation
transformed formula
Splitting
cubes
Solving

Cube-and-conquer solving

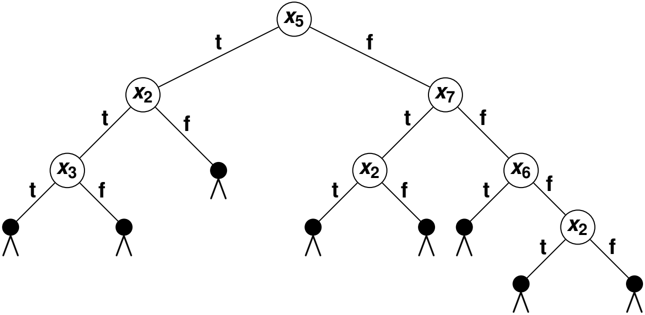
- **Problem: solving with conflict-driven clause learning (CDCL) is too slow**

Cube-and-conquer solving

- **Problem: solving with conflict-driven clause learning (CDCL) is too slow**
- **Solution:**
  - **use different heuristics $\Rightarrow$ cube-and-conquer solver (C&C)**

Cube-and-conquer solving

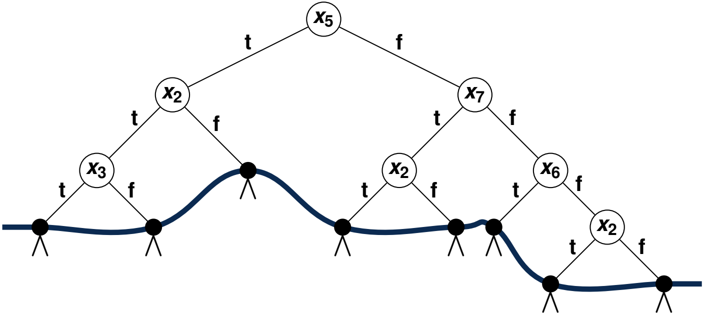
- Problem: solving with conflict-driven clause learning (CDCL) is too slow
- Solution:
  - use different heuristics ⇒ cube-and-conquer solver (C&C)



Cube-and-conquer solving

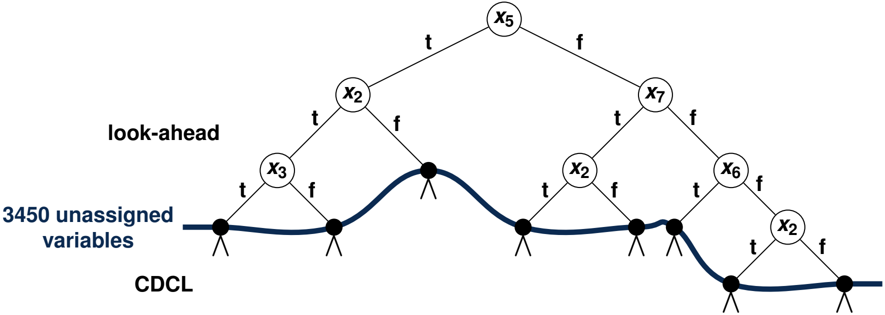
- Problem: solving with conflict-driven clause learning (CDCL) is too slow
- Solution:
  - use different heuristics ⇒ cube-and-conquer solver (C&C)

3450 unassigned variables



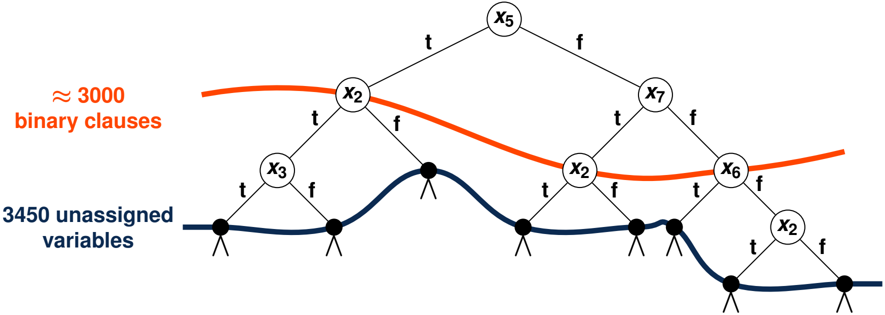
Cube-and-conquer solving

- Problem: solving with conflict-driven clause learning (CDCL) is too slow
- Solution:
  - use different heuristics ⇒ cube-and-conquer solver (C&C)



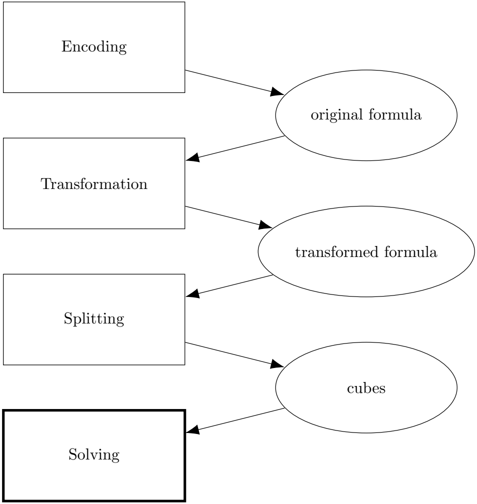
Cube-and-conquer solving

- Problem: solving with conflict-driven clause learning (CDCL) is too slow
- Solution:
  - use different heuristics ⇒ cube-and-conquer solver (C&C)
  - use parallelization (800 cores)



Solving

- $n = 7824$

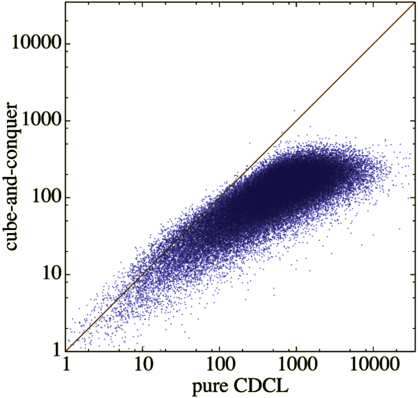


Runtime

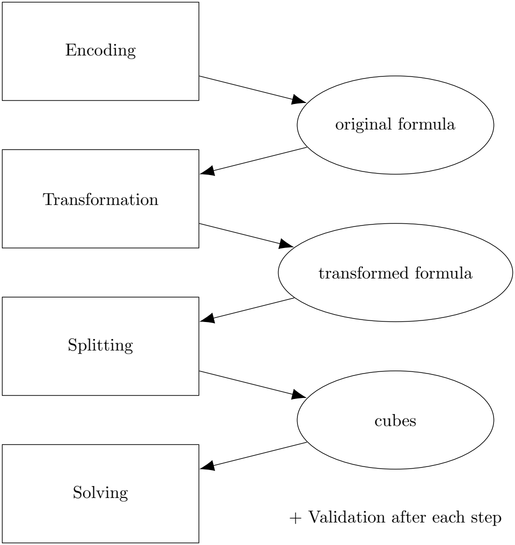
- **Splitting: 22000 CPU hours**
- **Solving: 13000 CPU hours**
- **Validation: 16000 CPU hours**
- **sums up to $\approx$ 5.8 CPU years**

Runtime

- Splitting: 22000 CPU hours
- Solving: 13000 CPU hours
- Validation: 16000 CPU hours
- sums up to $\approx$ 5.8 CPU years



Solution



+ Validation after each step