Stabbing Planes

Noah Fleming
Department of Computer Science
University of Toronto
Joint work with Paul Beame, Russell Impagliazzo, Antonina Kolokolova,
Denis Pankratov, Toniann Pitassi, and Robert Robere
Search

SAT - NP hard
- Intractable in worst-case

SAT Solvers
- Solve real world instances with millions of variables
- Often run in near-linear time!
- Used in model checking, planning, bioinformatics, etc.
Search

SAT - NP hard
- Intractable in worst-case

SAT Solvers
- Solve real world instances with millions of variables
- Often run in near-linear time!
- Used in model checking, planning, bioinformatics, etc.

Conflict-Driven Clause Learning (CDCL)
- Basis of state-of-the-art solvers
- Based on Davis–Putnam–Logemann–Loveland (DPLL) algorithm [DP60, DLL62] augmented with fine-tuned heuristics
\( \mathcal{F} = (x_1 \lor x_2) \land (\bar{x}_1 \lor x_2) \land (x_1 \lor \bar{x}_2) \land (\bar{x}_1 \lor \bar{x}_2) \)
\[ \mathcal{F} = (x_1 \lor x_2) \land (\bar{x}_1 \lor x_2) \land (x_1 \lor \bar{x}_2) \land (\bar{x}_1 \lor \bar{x}_2) \]
\[ F = (x_1 \lor x_2) \land (\overline{x}_1 \lor x_2) \land (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor \overline{x}_2) \]
\[ F = (x_1 \lor x_2) \land (\bar{x}_1 \lor x_2) \land (x_1 \lor \bar{x}_2) \land (\bar{x}_1 \lor \bar{x}_2) \]
\[ \mathcal{F} = (x_1 \lor x_2) \land (\overline{x}_1 \lor x_2) \land (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor \overline{x}_2) \]
DPLL

\[ \mathcal{F} = (x_1 \lor x_2) \land (\overline{x}_1 \lor x_2) \land (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor \overline{x}_2) \]

Proof of unsatisfiability!
DPLL

\[ \mathcal{F} = (x_1 \lor x_2) \land (\bar{x}_1 \lor x_2) \land (x_1 \lor \bar{x}_2) \land (\bar{x}_1 \lor \bar{x}_2) \]

Proof of unsatisfiability!

SAT solvers run on unsatisfiable CNFs output proofs.

- Proof complexity analysis applies to SAT solvers
- proof size = runtime of ideal implementation of search algorithm
DPLL

\[ F = (x_1 \lor x_2) \land (\overline{x_1} \lor x_2) \land (x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor \overline{x_2}) \]

DPLL proofs = tree-like Resolution proofs

Query based = Rule based
DPLL

\[ F = (x_1 \lor x_2) \land (\bar{x}_1 \lor x_2) \land (x_1 \lor \bar{x}_2) \land (\bar{x}_1 \lor \bar{x}_2) \]

DPLL proofs = \textit{tree-like} Resolution proofs

- Query based
- Rule based

CDCL augments DPLL with heuristics, clause learning, restarts, etc.
- Proofs still captured by Resolution
  - Weak proof system, cannot count
PseudoBoolean SAT Solvers

Reason about integer-linear inequalities rather than clauses

Most are based on Cutting Planes proof system
• stronger proof system than Resolution

Worse performance than state-of-the-art solvers based on DPLL
Many strong proof systems for which we can theoretically find proofs quickly.
• e.g. *polynomial calculus*

Best SAT algorithms based on DPLL
• DPLL can’t even count, no gaussian elimination!
Puzzle

Why is can’t we develop good search algorithms based on stronger proof systems?

Hypothesis:
Querying is more conducive to search algorithms

• Leads to simple divide-and-conquer style algorithms
• DPLL vs tree-like Resolution
Stabbing Planes

Generalization of DPLL to reason about integer-linear inequalities, formalized as a proof system
Stabbing Planes

\[ \mathcal{F} = \{ \text{Unsatisfiable set of integer-linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)

over \( \{0, 1\} \) assignments
Stabbing Planes

\[ \mathcal{F} = \{ \text{unsatisfiable set of linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)
Stabbing Planes

\[ \mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\} \]

Variables \( x, y \in [0, 1] \)

\[ x + y \]

\[ \mathcal{F} \cup \{x + y \leq 1\} \]

\[ \mathcal{F} \cup \{x + y \geq 2\} \]

\[ 1 < x + y < 2 \]

\[ x + y \leq 1 \]

\[ x + y \geq 2 \]
Stabbing Planes

\[ F = \{ \text{unsatisfiable set of linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)

\[ F \cup \{ x + y \leq 1 \} \]

\[ F \cup \{ x + y \geq 2 \} \]

\[ 1 < x + y < 2 \]

\[ x + y \leq 1 \]

\[ x + y \geq 2 \]

1 < \( x + y < 2 \) slab removed
Stabbing Planes

\[ \mathcal{F} = \{ \text{unsatisfiable set of linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)

\[ x + y \]

\[ \leq 1 \quad \geq 2 \]

\[ \mathcal{F} \cup \{ x + y \leq 1 \} \quad \mathcal{F} \cup \{ x + y \geq 2 \} \]

Can't remove \( \{0, 1\} \) points!

Any \( \alpha \in \{0, 1\}^n \) satisfies \( Ax \geq b \) or \( Ax \leq b - 1 \) for \( A \in \mathbb{Z}^n, b \in \mathbb{Z} \)

1 \( < x + y < 2 \) slab removed
Stabbing Planes

\[ \mathcal{F} = \{ \text{unsatisfiable set of linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)
Stabbing Planes

\[ \mathcal{F} = \{ \text{unsatisfiable set of linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)

Diagram:

- \( x + y \)
  - \( \leq 1 \)
    - \( x - y \)
      - \( \leq 0 \)
        - Empty
      - \( > 1 \)
        - Empty
  - \( \geq 2 \)
    - Empty

- \( x + y \geq 2 \)
  - \( 1 < x + y < 2 \)
    - \( x - y \geq 1 \)
      - \( x + y \leq 1 \)
        - \( 0 < x - y < 1 \)
      - \( x - y \leq 0 \)

Stabbing Planes

\[ \mathcal{F} = \{ \text{unsatisfiable set of linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)

**Farkas’ Lemma:** Polytope is empty iff there is a non-negative linear combination of its constraints equalling \( 0 \geq 1 \)
Stabbing Planes

\[ \mathcal{F} = \{ \text{unsatisfiable set of linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)

**Witness:** Non-negative linear combination of inequalities in \( \mathcal{F} \) and constraints along path to this node equalling \( 0 \geq 1 \)
Stabbing Planes

\[ \mathcal{F} = \{ \text{unsatisfiable set of linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)

- **Algebraic proof**

  \[
  \begin{align*}
  x + y &\leq 1 \\
  x + y &\geq 2 \\
  x - y &\leq 0 \\
  x - y &> 1 \\
  \text{Witness} &\leq 0 \\
  \text{Witness} &> 1 \\
  \text{Witness} &\leq 0 \\
  \text{Witness} &> 1
  \end{align*}
  \]

- **Geometric proof**

  \[
  \begin{align*}
  1 < x + y < 2 \\
  x - y \geq 1 \\
  x + y \leq 1 \\
  0 < x - y < 1
  \end{align*}
  \]
Stabbing Planes

\[ \mathcal{F} = \{ \text{unsatisfiable set of linear inequalities} \} \]

Variables \( x, y \in [0, 1] \)

Complexity measures

Size: Number of nodes

Depth: Tree-depth
SP Generalizes DPLL

\[ \mathcal{F} = (x_1 \lor x_2) \land (\bar{x}_1 \lor x_2) \land (x_1 \lor \bar{x}_2) \land (\bar{x}_1 \lor \bar{x}_2) \]
$\mathcal{F} = \{x_1 + x_2 \geq 1, x_1 - x_2 \geq 0, x_2 - x_1 \geq 0, -x_1 - x_2 \geq -1\}$

Variables $x_1, x_2 \in [0, 1]$
SP Generalizes DPLL

\[ F = \{ x_1 + x_2 \geq 1, x_1 - x_2 \geq 0, x_2 - x_1 \geq 0, -x_1 - x_2 \geq -1 \} \]

Variables \( x_1, x_2 \in [0, 1] \)
Stabbing Planes

Polynomially equivalent to a *tree-like* variant of the R(CP) proof system introduced in [Krajíček98]
- R(CP) - rule based
- Stabbing Planes - query based

Query-based viewpoint valuable for upper bounds.

**Theorem:** Quasi-polynomial size SP proof of any system of linear equations over a finite field
Proof Sketch

**Input:** Unsatisfiable system of \( \mod 2 \) linear equations over \( \{0, 1\} \) assignments

<table>
<thead>
<tr>
<th>( C_1 )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( x_6 )</th>
<th>( x_7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( C_3 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( C_4 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( C_5 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( C_6 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( C_7 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
C_1 & \equiv 1 \\
C_2 & \equiv 0 \\
C_3 & \equiv 1 \\
C_4 & \equiv 1 \\
C_5 & \equiv 0 \\
C_6 & \equiv 1 \\
C_7 & \equiv 1 \\
\end{align*}
\]
Proof Sketch

**Input:** Unsatisfiable system of $\mod 2$ linear equations

$\Rightarrow$ Exists subset which sum to $0 = 1 \mod 2$

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$C_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$C_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$C_4$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$C_5$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$C_6$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Sum to $0 = 1 \mod 2$
### Proof Sketch

**Idea:** Split set of constraints in half. For any \( \{0, 1\} \) assignment, one of the halves has a falsified equation.
- The sum of the constraints tell us which one!

![Proof Sketch Diagram]
**Proof Sketch**

**Idea:** Split set of constraints in half. For any assignment, one of the halves has a falsified equation.  
- The sum of the constraints tell us which one!

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
<th>Column 6</th>
<th>Column 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ x_3 + x_7 \text{ must be 0 mod 2} \]

\[ x_3 + x_7 \text{ must be 1 mod 2} \]
Proof Sketch

Recursive Step:
1. Partition constraints into two sets $S_1, S_2$
**Proof Sketch**

**Recursive Step:**
1. Partition constraints into two sets $S_1, S_2$
2. Determine value of sum of constraints $\mod 2$ in both sets

\[
\sum_{c_i \in S_1} \geq k \text{ for } k = 1, \ldots, \text{max}
\]

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$C_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$C_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$C_4$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$C_5$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$C_6$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[\begin{align*}
\text{mod 2} & \quad 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
\text{mod 2} & \quad 0 & 0 & 1 & 0 & 0 & 1 & 0 & \end{align*}\]
Proof Sketch

Recursive Step:
1. Partition constraints into two sets $S_1$, $S_2$
2. Determine value of sum of constraints $\mod\ 2$ in both sets
3. Recurse on set whose sum is not satisfied

\[
\begin{array}{cccccccc}
  & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
C_1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
C_2 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
C_3 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
C_4 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
C_5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
C_6 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
\end{array}
\]

\[
\mod\ 2
\]
Proof Sketch

Recursive Step:
1. Partition constraints into two sets $S_1, S_2$
2. Determine value of sum of constraints mod 2 in both sets
3. Recurse on set whose sum is not satisfied

Suppose: $x_3 + x_7 \equiv 0 \text{ mod } 2$

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$C_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$C_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$C_4$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$C_5$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$C_6$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Recursion
Proof Sketch

Recursive Step:
1. Partition constraints into two sets $S_1, S_2$
2. Determine value of sum of constraints $\mod 2$ in both sets
3. Recurse on set whose sum is not satisfied

Suppose:
\[
x_1 + x_2 + x_3 + x_4 + x_5 + x_7 \equiv 0 \mod 2
\]
\[
x_1 + x_2 + x_4 + x_5 \equiv 0 \mod 2
\]

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_4$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$C_5$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$C_6$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Recursion
Proof Sketch

**Termination:**

By recursive step we have derived \( x_1 + x_2 + x_4 + x_5 \equiv 0 \pmod{2} \)
Using the constraint \( x_1 + x_2 + x_4 + x_5 \equiv 1 \pmod{2} \) we can derive \( 0 \geq 1 \)

\[
\begin{array}{cccccccc}
C_6 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
& 1 & 1 & 0 & 1 & 1 & 0 & 0 & \equiv 1 \pmod{2}
\end{array}
\]
Proof Sketch

**Termination:**
By recursive step we have derived \( x_1 + x_2 + x_4 + x_5 \equiv 0 \mod 2 \)
Using the constraint \( x_1 + x_2 + x_4 + x_5 \equiv 1 \mod 2 \) we can derive \( 0 \geq 1 \)

**Analysis:**
Each recursive step requires branches \( O(n) \)
\( O(\log n) \) recursive steps - Reduce the set of constraints by half each time
Other Results

SP can solve systems of linear equations over a finite field
• quasi-polynomial size proofs of Tseitin formulas (conjectured to be hard for Cutting Planes)
Other Results

SP can solve systems of linear equations over a finite field
  • quasi-polynomial size proofs of *Tseitin formulas* (conjectured to be hard for Cutting Planes)

SP can polynomially simulate Cutting Planes
  • size $s$ CP proof $\rightarrow$ size $O(s)$ SP proof
  • Surprising because SP is tree-like, while CP is DAG-like
Other Results

SP can solve systems of linear equations over a finite field
• quasi-polynomial size proofs of Tseitin formulas (conjectured to be hard for Cutting Planes)

SP can polynomially simulate Cutting Planes
• size $s$ CP proof $\Rightarrow$ size $O(s)$ SP proof
• Surprising, SP proofs are tree-like, while CP proofs are DAG-like

$\Omega(n/\log^2 n)$ depth lower bound
• reduction to real communication complexity
• same technique cannot give size lower bounds
  - real communication protocols can’t be balanced,
  - SP proofs can’t be balanced
Open Problem

Super-polynomial size lower bounds on SP?

Does SP proof size (#nodes) equal SP proof bit-size?
• [Muroga72] Any integer-linear inequality separating two subsets \( U, V \) subset \( \{0, 1\}^n \) can be represented by \( \text{poly}(n) \) bits

Separate Cutting Planes and Stabbing Planes
• Candidate: Tseitin formulas

SP-based search algorithms?