Hybrid SAT Solving by Continuous Optimization

Presented by Zhiwei Zhang
Joint work with Anastasios Kyrillidis, Anshumali Shrivastava, Moshe Vardi
Boolean SAT and MaxSAT

Boolean variables: $x_1, x_2, \ldots \in \{0, 1\}$

Boolean constraints: $x_1 \lor \neg x_2, x_2 \oplus x_3, x_1 + x_2 + x_3 + x_4 + x_5 \geq 2, \ldots$

Boolean formula: e.g. $\varphi = (x_1 \lor \neg x_2) \land (x_2 \oplus \neg x_3) \land (x_2 + x_3 + x_4 \geq 2)$

SATisfiability: finding an assignment that satisfies all constraints

MaxSATisfiability: finding an assignment that satisfies as many constraints as possible

Discrete Optimization  Software Verification  Motion planning  Probabilistic inference  Machine Learning
CNF and SAT Solvers

- Conjunctive Normal Form (AND of ORs)
  
  \[ \varphi = (x_1 \lor x_2) \land (x_2 \lor \neg x_3) \land (x_2 \lor x_3 \lor x_4) \]

- Modern CNF Solvers

  CDCL-based SAT solvers: branching on variables with unit propagation, backtracking and clause learning

  (discrete) local search SAT solvers: Objective function: \( f_\varphi(x) = \# \) of constraints satisfied by \( x \)

(Greedy) Local Search

  Randomly generate a complete assignment \( x \in \{0, 1\}^n \)
  while there are unsatisfied constraints
  
  flip the value of the “best” variable to increase the \# of satisfied constraints
The Success of Existing Solvers Rely on Properties of CNF Format

- CNF is the most preferable format currently
  - CDCL solvers:
    \[ x_1 = T \quad \Rightarrow \quad (x_1 \lor x_2 \lor x_3 \lor x_4) \text{ is satisfied} \]
    constraint simplification
    \[ (x_1 \lor x_2 \lor x_3 \lor x_4) \text{ and } x_1 = x_2 = x_3 = F \quad \Rightarrow \quad x_4 = T \]
    unit propagation
  - Discrete local search solvers:
    \[ (x_1 \lor x_2 \lor x_3 \lor x_4) \text{ is unsatisfied} \quad \Rightarrow \quad \text{flipping one variable is enough} \]

- non-CNF constraints are harder to handle
  \[ (x_1 + x_2 + x_3 + x_4 \geq 2) \text{ is unsatisfied} \quad \Rightarrow \quad \text{flipping one variable might not be enough} \]
  The capability to “flip” more than one bits is important
From Discrete to Continuous Local Search

\[ x = 0 \quad 1 \quad 0 \quad 1 \quad 1 \]

\( x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \)

\[ \begin{array}{c}
\text{F} \\
\text{T} \\
\text{F} \\
\text{T} \\
\text{T} \\
\end{array} \]

\( C_1 \quad C_2 \quad C_3 \)

\[ f_\varphi(x) = \# \text{ of constraints satisfied by } x \\
= c_1(x) + c_2(x) + c_3(x) \\
= 1 + 0 + 0 = 1 \]

\[ a = 0.2 \quad 0.6 \quad 0.1 \quad 0.8 \quad 0.7 \]

\[ S_a : \ P[x_1 = 1] = 0.2 \]
\[ P[x_1 = 0] = 0.8 \]

input probability

\[ \begin{array}{c}
\text{F} \\
\text{F} \\
\text{F} \\
\text{T} \\
\text{T} \\
\end{array} \]

\( x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \)

\[ \begin{array}{c}
\text{T} \\
\text{F} \\
\text{F} \\
\text{T} \\
\text{T} \\
\end{array} \]

\( C_1 \quad C_2 \quad C_3 \)

\[ \begin{array}{c}
\text{P}[c_1 = 1] = 0.6 \\
\text{P}[c_2 = 1] = 0.1 \\
\text{P}[c_3 = 1] = 0.8 \\
\end{array} \]

output probability

\[ F_\varphi(a) = \mathbb{E}_{x \in S_a} f_\varphi(x) \\
= \sum_{c} \mathbb{P}_{x \in S_a} [c(x) = 1] \\
= 0.6 + 0.1 + 0.8 = 1.5 \]

"flipping" the value of variables to maximize \( f_\varphi \)

tuning the input probability of variables to maximize \( F_\varphi \)
Reduce SAT to Continuous Optimization

\[ f_\varphi(x) = \# \text{ of constraints satisfied by } x \]
\[ F_\varphi(a) = \mathbb{E}_{x \in S_a} f_\varphi(x) = \sum_c \mathbb{P}_{x \in S_a} [c(x) = 1] \]

"flipping" the value of variables to minimize \( f \)
tuning the input probability of variables to minimize \( F \)

**Proposition**

\( \varphi \) is satisfiable \( \leftrightarrow \) \( \max_{x \in \{0,1\}^n} f_\varphi(x) = \# \text{ constraints} \) \( \leftrightarrow \) \( \max_{a \in [0,1]^n} F_\varphi(a) = \# \text{ constraints} \)
The Walsh-Fourier Expansion of Boolean Functions

- Walsh-Fourier transform:

  Boolean functions $c : \{1, -1\}^n \rightarrow \{0, 1\}$
  Multilinear Polynomials $\text{FE}_c : \{1, -1\}^n \rightarrow \{0, 1\}$
  $\{F, T\}$ $\{F, T\}$
  $x \land y$ $\frac{1}{4} \cdot 1 - \frac{1}{4} \cdot x - \frac{1}{4} \cdot y + \frac{1}{4} \cdot xy$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$x \land y$</th>
<th>$\frac{1}{4} - \frac{1}{4} x - \frac{1}{4} y + \frac{1}{4} xy$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Theorem

Every Boolean function $c$ has a unique representation in multilinear polynomial that agrees with $c$ on all Boolean assignments.
How to compute expectation: by Walsh-Fourier expansion

\[ c = x \land y \quad \Rightarrow \quad F_{E_c} = \frac{1}{4} \cdot 1 - \frac{1}{4} \cdot x - \frac{1}{4} \cdot y + \frac{1}{4} \cdot xy \]

**Proposition**

\[ \mathbb{P}_{x \in S_a} [c(x) = 1] = F_{E_c}(a) \text{ for all } a \in [-1, 1]^n \]

\[ F_\varphi(a) = \mathbb{E}_{x \in S_a} f_\varphi(x) = \sum_c \mathbb{P}_{x \in S_a} [c(x) = 1] = \sum_c F_{E_c}(a) \]

\[ \varphi = (x_1 \land x_2) \land (\neg x_1 \oplus x_2) \]

\[ F_\varphi = \left( \frac{1}{4} \cdot 1 - \frac{1}{4} \cdot x_1 - \frac{1}{4} \cdot x_2 + \frac{1}{4} \cdot x_1 x_2 \right) + \frac{1+x_1 x_2}{2} \]

Apply gradient ascent
Workflow of Our Gradient Ascend-Based Approach

\[ f = (x_1 \lor \neg x_2) \land (x_2 \oplus x_3 \oplus x_4) \land (x_4 + x_5 + x_6 + x_9 \geq 2) \]

hybrid Boolean formula

Walsh-Fourier transform

\[ F = -0.32 + 0.02x_1 - 0.03x_2 + 0.04x_3 - 0.007x_4 + \cdots + 0.000875x_5x_6x_7x_8x_9 + \cdots \]

multilinear polynomial

analytical computation

Is a solution?

No

(1, -1, \cdots)

discrete assignment

discretize

continuous optimization

\( \frac{\partial F}{\partial x_1} = 0.003x_2 + \cdots \)

\( \frac{\partial F}{\partial x_2} = 0.008x_1 + \cdots \)

\( \cdots \)

compute gradients

(0.586, -0.324, \cdots)
Where Will We Converge to? The Geometry of Multilinear Polynomials

• Multilinear polynomials are non-convex and non-concave

• Locally, $F_\varphi$ is always convex among some directions while concave on some other directions

In unconstrained setting, on every point $a \in \mathbb{R}^n$, there is always a direction that can increase the value of a multilinear polynomial

In the constrained setting where $a \in [-1, 1]^n$, this direction might be towards the boundary. Thus we may still encounter local maxima along the boundary
Where Will We Converge to? An “Almost” Discrete Assignment

\[ \varphi = x \lor y \]
\[ F_\varphi = \frac{3}{4} - \frac{1}{4}x - \frac{1}{4}y - \frac{1}{4}xy \]

Theorem
Rounding preserves objective value after converging to a local maximum.
The Versatility of Our Approach

- Modern-SAT solvers are highly CNF-focused

- Other types of constraints are also important

  XOR: \((x_2 \oplus \neg x_3)\)

  cardinality constraints: \((x_1 + x_2 + x_3 + x_4 \geq 2)\)

  pseudo-Boolean constraints: \((3x_1 - 4x_2 + x_3 + 6x_4 \geq 5)\)

Theorem

Disjunctive clauses (CNF), XOR and cardinality constraints all have closed-form Walsh-Fourier expansions.

- Our approach treats different types of constraints uniformly as polynomials
Better Global Convergence--Adding Constraint Weights

\[ F_\varphi(a) = \sum_c \mathbb{P}_{x \in S_a} [c(x) = 1] = \sum_c FE_c(a) \quad \Rightarrow \quad F_{\varphi,w}(a) = \sum_c w(c) \cdot \mathbb{P}_{x \in S_a} [c(x) = 1] = \sum_c w(c) \cdot FE_c(a) \]

- Different constraints have different relative importance
  Weightings change the landscape and attractive regions

\[ \varphi = (x_1 \land x_2) \land (\neg x_1 \lor x_2) \]

Adaptative weighting: increase the weight of unsatisfied constraints
Better Versatility--Beyond Walsh-Fourier Expansions

\[ x \land y \]

\[
\frac{1}{4} \cdot 1 - \frac{1}{4} \cdot x - \frac{1}{4} \cdot y + \frac{1}{4} \cdot xy
\]

Walsh-Fourier Expansion

Walsh-Fourier expansions can not handle pseudo-Boolean constraints:

\[ 3x_1 - 4x_2 + x_3 + 6x_4 \geq 5 \]

**Proposition**

Given the BDD with size \( S \) of a constraint, the output probability can be computed in \( O(S) \).

- We are able to handle coefficient-bounded pseudo-Boolean constraints.
Better Efficiency--Computing the Gradient by BDDs

- Nodes sharing between constraints

- Gradients of all coordinates can be computed simultaneously

Theorem

Given the Multi-Rooted BDD with total size $S$, the gradient can be computed in $O(S)$. 
Experimental Results

<table>
<thead>
<tr>
<th>Solver</th>
<th>Avg. Score</th>
<th># of best solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>GradSAT (Our approach)</td>
<td>0.971</td>
<td>489</td>
</tr>
<tr>
<td>WalkSAT (discrete LS-based)</td>
<td>0.925</td>
<td>124</td>
</tr>
<tr>
<td>Mixing Method (SDP-based)</td>
<td>0.901</td>
<td>126</td>
</tr>
<tr>
<td>Loandra (SAT-based)</td>
<td>0.883</td>
<td>42</td>
</tr>
</tbody>
</table>

Results on 575 small-size instances from MaxSAT Competition

On random CNF-XOR and pseudo-Boolean benchmarks, our solver is better than discrete local-search-based solvers.

On large industrial instances: the cost for differentiation on the real domain is still too expensive.
Summary: Hybrid SAT Solving by Continuous Optimization

- Our motivations are:
  - handling constraints beyond CNFs
  - exploring the potential of continuous methods in SAT/MaxSAT solving

- We find:
  - nice theoretical results
  - interesting problems for the continuous optimization approach

- Our method can:
  - act as a complement to existing SAT/MaxSAT solvers
  - benefit from tractable structures of Boolean functions
    and techniques from continuous optimization
Challenges and Future Directions

- Full gradient is too expensive
  compute imprecise gradient on large instances
- Balance between Continuous and Discrete Local Search

  Fully Discrete Local Search            Fully Continuous Local Search

- Continuous optimizer as a layer of NN
• Our motivations are
  handling constraints beyond CNFs
  exploring the potential of continuous methods in SAT/MaxSAT solving

• We find
  nice theoretical results
  interesting problems
  for the continuous optimization approach

• Our method can:
  act as a complement to existing SAT solvers
  benefit from tractable structures of Boolean functions
  and techniques from continuous optimization

• In the future:
  compute imprecise gradient on large instances
  balance between Continuous and Discrete
  as a layer of neural network