SharpSAT-TD: Improving SharpSAT by Exploiting Tree Decompositions

Tuukka Korhonen and Matti Järvisalo

HIIT, Department of Computer Science, University of Helsinki, Finland

MC-2021
Online
July 6, 2021
New modification of SharpSAT [Thurley '06]

1. Integrates low-width tree decompositions to the variable selection heuristic
2. Implements new preprocessor
3. Directly supports weighted model counting
### MCC-2021 Results on Public Instances

<table>
<thead>
<tr>
<th>Solver</th>
<th>Config</th>
<th>Solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>sharp-tw-unweighted</td>
<td>default</td>
<td>83/100</td>
</tr>
<tr>
<td>Narsimha</td>
<td>track1_conf2.sh</td>
<td>69/100</td>
</tr>
<tr>
<td>d4</td>
<td>TRACK1+4_ds_preprocSharpequiv.sh</td>
<td>59/100</td>
</tr>
<tr>
<td>d4</td>
<td>TRACK1+4_ms_preprocSharpequiv.sh</td>
<td>57/100</td>
</tr>
<tr>
<td>TwG</td>
<td>2.sh</td>
<td>38/100</td>
</tr>
<tr>
<td>Narsimha</td>
<td>track2_conf1.sh</td>
<td>72/100</td>
</tr>
<tr>
<td>d4</td>
<td>TRACK1+4_ds_preprocSharpequiv.sh</td>
<td>57/100</td>
</tr>
<tr>
<td>d4</td>
<td>TRACK1+4_ms_preprocSharpequiv.sh</td>
<td>57/100</td>
</tr>
<tr>
<td>Narsimha</td>
<td>track4_conf1.sh</td>
<td>68/100</td>
</tr>
<tr>
<td>sharp-tw-weighted</td>
<td>default</td>
<td>99/100</td>
</tr>
<tr>
<td>d4</td>
<td>TRACK2+3ds_preprocEquiv.s</td>
<td>81/100</td>
</tr>
<tr>
<td>d4</td>
<td>TRACK2+3_ms_preprocEquiv.sh</td>
<td>81/100</td>
</tr>
<tr>
<td>c2d</td>
<td>default</td>
<td>74/100</td>
</tr>
<tr>
<td>d4</td>
<td>TRACK1+4_ds_preprocSharpequiv.sh</td>
<td>57/100</td>
</tr>
<tr>
<td>dpmc4fix</td>
<td>4</td>
<td>49/100</td>
</tr>
</tbody>
</table>
Overview

Overview of SharpSAT-TD

1. Preprocess
2. Compute a tree decomposition with FlowCutter [Strasser ’17]
3. Count using tree decomposition guided variable selection
Overview

Overview of SharpSAT-TD

1. Preprocess
2. Compute a tree decomposition with FlowCutter [Strasser ’17]
3. Count using tree decomposition guided variable selection

I will first talk about (3), then about (1), and then about other changes compared to SharpSAT
Tree Decompositions

\[ (\neg x_2 \lor x_3) \land (x_3 \lor \neg x_6) \land (x_5 \lor x_6) \land (x_1 \lor \neg x_2 \lor x_5) \land (x_1 \lor \neg x_4) \]

Primal graph
Tree Decompositions

\((\neg x_2 \lor x_3) \land (x_3 \lor \neg x_6) \land (x_5 \lor x_6) \land (x_1 \lor \neg x_2 \lor x_5) \land (x_1 \lor \neg x_4)\)

Primal graph

Tree decomposition
Tree Decompositions

\((-x_2 \lor x_3) \land (x_3 \lor -x_6) \land (x_5 \lor x_6) \land (x_1 \lor -x_2 \lor x_5) \land (x_1 \lor -x_4)\)

- Primal graph
- Tree decomposition

- Width of a tree decomposition: Size of the largest bag - 1
- Treewidth of a graph/CNF: Minimum width of a tree decomposition
Tree Decomposition Guided Variable Selection

- Select the variable of the active formula that appears the closest to the root in the tree decomposition

\[(\neg x_2 \lor x_3) \land (x_3 \lor \neg x_6) \land (x_5 \lor x_6) \land (x_1 \lor \neg x_2 \lor x_5) \land (x_1 \lor \neg x_4)\]
Tree Decomposition Guided Variable Selection

- Select the variable of the active formula that appears the closest to the root in the tree decomposition

\[(x_3) \land (x_3 \lor \neg x_6) \land (x_5 \lor x_6) \land (x_1 \lor x_5) \land (x_1 \lor \neg x_4)\]
Select the variable of the active formula that appears the closest to the root in the tree decomposition

\[(x_5 \lor x_6) \land (x_1 \lor \neg x_2 \lor x_5) \land (x_1 \lor \neg x_4)\]

```
root
\[x_2, x_3, x_5\]

\[x_1, x_2, x_5\]

\[x_3, x_5, x_6\]

\[x_1, x_4\]
```

\[x_2 = 1, \quad x_3 = 1,\]
Tree Decomposition Guided Variable Selection

- Select the variable of the active formula that appears the closest to the root in the tree decomposition

\[(x_1 \lor \neg x_4)\]

Component analysis

\[x_2 = 1, \quad x_3 = 1, \quad x_5 = 1,\]
Tree Decomposition Guided Variable Selection

- Select the variable of the active formula that appears the closest to the root in the tree decomposition

Component analysis

\[ x_2 = 1, \quad x_3 = 1, \quad x_5 = 1, \quad x_1 = 1 \]
Theoretical Background

Proposition ([BDP03, Dar01])

Standard \#DPLL algorithm, with component analysis and component caching, works in $2^w \text{poly}(|\phi|)$ time when using a tree decomposition of width $w$ for variable selection.
Proposition ([BDP03, Dar01])

Standard \#DPLL algorithm, with component analysis and component caching, works in $2^w \text{poly}(\phi)$ time when using a tree decomposition of width $w$ for variable selection.
Implementation of Variable Selection

Variable $x$ with highest $\text{score}(x)$ is selected.

Standard SharpSAT:

$$\text{score}(x) = \text{act}(x) + \text{freq}(x)$$

Where

- $\text{act}(x)$ is VSIDS-like activity score
- $\text{freq}(x)$ is the number of occurrences of $x$ in the current formula
Implementation of Variable Selection

Variable $x$ with highest score $\text{score}(x)$ is selected.

Standard SharpSAT:

$$\text{score}(x) = \text{act}(x) + \text{freq}(x)$$

SharpSAT-TD:

$$\text{score}(x) = \text{act}(x) + \text{freq}(x) - C \cdot d(x)$$

Where

- $\text{act}(x)$ is VSIDS-like activity score
- $\text{freq}(x)$ is the number of occurrences of $x$ in the current formula
- $d(x)$ is the distance from root of tree decomposition to closest bag containing $x$
- $C$ is some positive constant
Implementation of Variable Selection

Variable $x$ with highest score$(x)$ is selected.

Standard SharpSAT:

$$score(x) = act(x) + freq(x)$$

SharpSAT-TD:

$$score(x) = act(x) + freq(x) - C \cdot d(x)$$

Where

- $act(x)$ is VSIDS-like activity score
- $freq(x)$ is the number of occurrences of $x$ in the current formula
- $d(x)$ is the distance from root of tree decomposition to closest bag containing $x$
- $C$ is some positive constant
  - If $C$ is large, selection is purely by tree decomposition
  - If $C$ is small, selection is same as in standard SharpSAT
Implementation of Variable Selection

Variable $x$ with highest $\text{score}(x)$ is selected.

Standard SharpSAT:

$$\text{score}(x) = \text{act}(x) + \text{freq}(x)$$

SharpSAT-TD:

$$\text{score}(x) = \text{act}(x) + \text{freq}(x) - C \cdot d(x)$$

Where

- $\text{act}(x)$ is VSIDS-like activity score
- $\text{freq}(x)$ is the number of occurrences of $x$ in the current formula
- $d(x)$ is the distance from root of tree decomposition to closest bag containing $x$
- $C$ is some positive constant
  - If $C$ is large, selection is purely by tree decomposition
  - If $C$ is small, selection is same as in standard SharpSAT
  - $C$ chosen per-instance based on the width of the tree decomposition
Preprocessing

New preprocessor implementation, with

1. Complete vivification (minimalize each clause, with SAT solver)
2. Redundant clause deletion
3. Equivalent variable merging (treewidth-aware)
4. Re-implementation of B+E [LLM16] (treewidth-aware)
Preprocessing

New preprocessor implementation, with

1. Complete vivification (minimize each clause, with SAT solver)
Preprocessing

New preprocessor implementation, with

1. Complete vivification (minimalize each clause, with SAT solver)

2. Redundant clause deletion
Preprocessing

New preprocessor implementation, with

1. Complete vivification (minimalize each clause, with SAT solver)
2. Redundant clause deletion
3. Equivalent variable merging (treewidth-aware)
Preprocessing

New preprocessor implementation, with

1. Complete vivification (minimalize each clause, with SAT solver)
2. Redundant clause deletion
3. Equivalent variable merging (treewidth-aware)
4. Re-implementation of B+E [LLM16] (treewidth-aware)
Other Modifications

- “Implicit BCP” disabled
- LBD learned clause scoring scheme [AS09]
- Probabilistic component caching [SRSM19]
Other Modifications

- “Implicit BCP” disabled
- LBD learned clause scoring scheme [AS09]
- Probabilistic component caching [SRSM19]
- Extension to weighted model counting via template parameters – easily extensible to model counting over any semiring
The end

Thank you for your attention!
Gilles Audemard and Laurent Simon.  
Predicting learnt clauses quality in modern SAT solvers.  

F. Bacchus, S. Dalmao, and T. Pitassi.  
Algorithms and complexity results for #SAT and Bayesian inference.  

A. Darwiche.  
Decomposable negation normal form.  

J. Lagniez, E. Lonca, and P. Marquis.  
Improving model counting by leveraging definability.  

GANAK: A scalable probabilistic exact model counter.  