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

SharpSAT-TD

- 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

solver	config	solved
sharp-tw-unweighted	default	83/100
Narsimha	track1_conf2.sh	69/100
Narsimha	track1_conf1.sh	61/100
d4	TRACK1+4_ds_preprocSharpEquiv.sh	59/100
d4	TRACK1+4_ms_preprocSharpEquiv.sh	57/100
TwG	2.sh	38/100

solver	config	solved
sharp-tw-weighted	default	99/100
d4	TRACK2+3ds_preprocEquiv.sh	81/100
d4	TRACK2+3_ms_preprocEquiv.sh	81/100
c2d	default	74/100
Narsimha	track2_conf1.sh	72/100

solver	config	solved
Narsimha	track4_conf2.sh	69/100
sharp-tw-unweighted	default	69/100
Narsimha	track4_conf1.sh	68/100
d4	TRACK1+4_ds_preprocSharpEquiv.sh	57/100
d4	TRACK1+4_ms_preprocSharpEquiv.sh	57/100
dpmc4fix	4	49/100

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

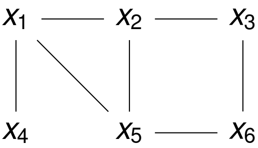
Overview of SharpSAT-TD

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

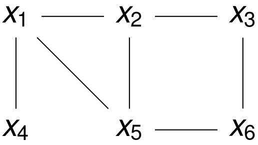
$$(\neg x_2 \vee x_3) \wedge (x_3 \vee \neg x_6) \wedge (x_5 \vee x_6) \wedge (x_1 \vee \neg x_2 \vee x_5) \wedge (x_1 \vee \neg x_4)$$



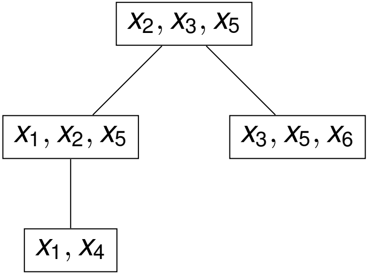
Primal graph

Tree Decompositions

$$(\neg x_2 \vee x_3) \wedge (x_3 \vee \neg x_6) \wedge (x_5 \vee x_6) \wedge (x_1 \vee \neg x_2 \vee x_5) \wedge (x_1 \vee \neg x_4)$$



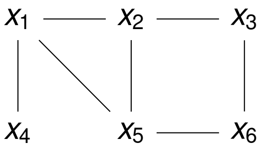
Primal graph



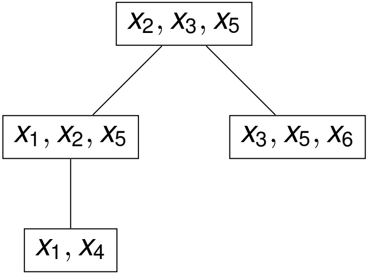
Tree decomposition

Tree Decompositions

$$(\neg x_2 \vee x_3) \wedge (x_3 \vee \neg x_6) \wedge (x_5 \vee x_6) \wedge (x_1 \vee \neg x_2 \vee x_5) \wedge (x_1 \vee \neg x_4)$$



Primal graph



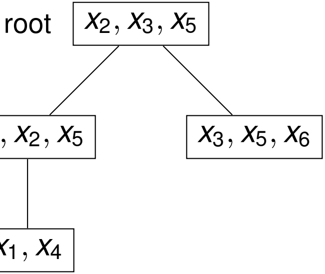
Tree decomposition

- 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

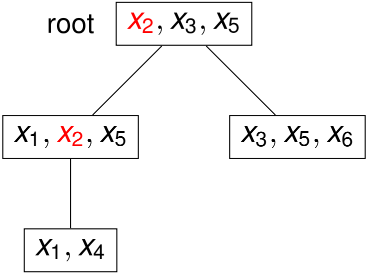
$$(\neg x_2 \vee x_3) \wedge (x_3 \vee \neg x_6) \wedge (x_5 \vee x_6) \wedge (x_1 \vee \neg x_2 \vee x_5) \wedge (x_1 \vee \neg x_4)$$



Tree Decomposition Guided Variable Selection

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

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

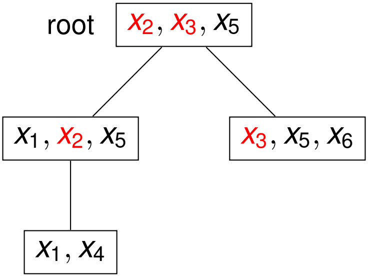


$x_2 = 1,$

Tree Decomposition Guided Variable Selection

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

$$(x_5 \vee x_6) \wedge (x_1 \vee \neg x_2 \vee x_5) \wedge (x_1 \vee \neg x_4)$$

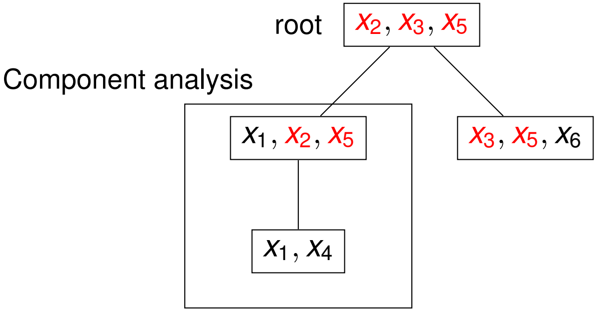


$x_2 = 1, \quad x_3 = 1,$

Tree Decomposition Guided Variable Selection

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

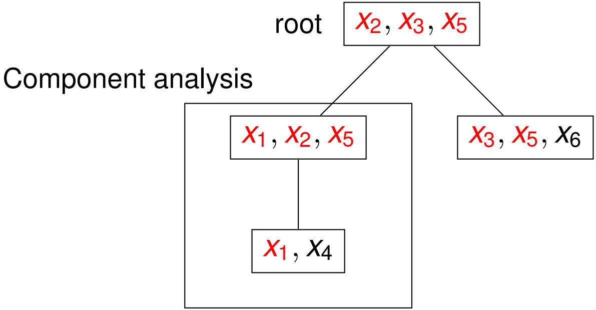
$$(x_1 \vee \neg x_4)$$



$$x_2 = 1, \quad x_3 = 1, \quad x_5 = 1,$$

Tree Decomposition Guided Variable Selection

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



$x_2 = 1, \quad x_3 = 1, \quad x_5 = 1, \quad x_1 = 1$

Theoretical Background

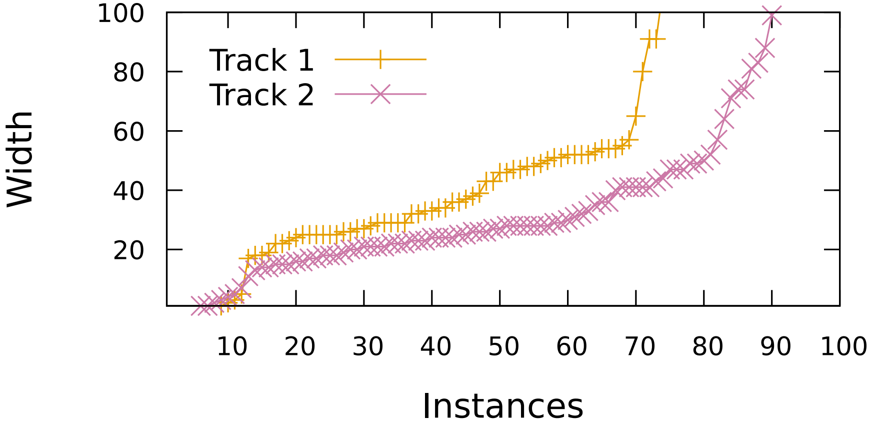
Proposition ([BDP03, Dar01])

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

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.



Implementation of Variable Selection

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

Standard SharpSAT:

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

Where

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

Implementation of Variable Selection

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

Standard SharpSAT:

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

SharpSAT-TD:

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

Where

- $\mathtt{act}(x)$ is VSIDS-like activity score
- $\mathtt{freq}(x)$ 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 $\texttt{score}(x)$ is selected.

Standard SharpSAT:

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

SharpSAT-TD:

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

Where

- $\texttt{act}(x)$ is VSIDS-like activity score
- $\texttt{freq}(x)$ 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 $\mathtt{score}(x)$ is selected.

Standard SharpSAT:

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

SharpSAT-TD:

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

Where

- $\mathtt{act}(x)$ is VSIDS-like activity score
- $\mathtt{freq}(x)$ 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

Preprocessing

New preprocessor implementation, with

1. Complete vivification (minimalize 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!

Bibliography

Gilles Audemard and Laurent Simon.
Predicting learnt clauses quality in modern SAT solvers.
In *IJCAI*, pages 399–404, 2009.

F. Bacchus, S. Dalmao, and T. Pitassi.
Algorithms and complexity results for #SAT and Bayesian inference.
In *FOCS*, pages 340–351. IEEE, 2003.

A. Darwiche.
Decomposable negation normal form.
J. ACM, 48(4):608–647, 2001.

J. Lagniez, E. Lonca, and P. Marquis.
Improving model counting by leveraging definability.
In *IJCAI*, pages 751–757. IJCAI/AAAI Press, 2016.

S. Sharma, S. Roy, M. Soos, and K. S. Meel.
GANAK: A scalable probabilistic exact model counter.
In *IJCAI*, pages 1169–1176. ijcai.org, 2019.