SAT-Based Approaches to Treewidth Computation: An Evaluation

Jeremias Berg Matti Järvisalo

HIIT & Dept. of Computer Science
University of Helsinki
Finland

February 25, 2018
ICTAI 2014

Contributions of Work

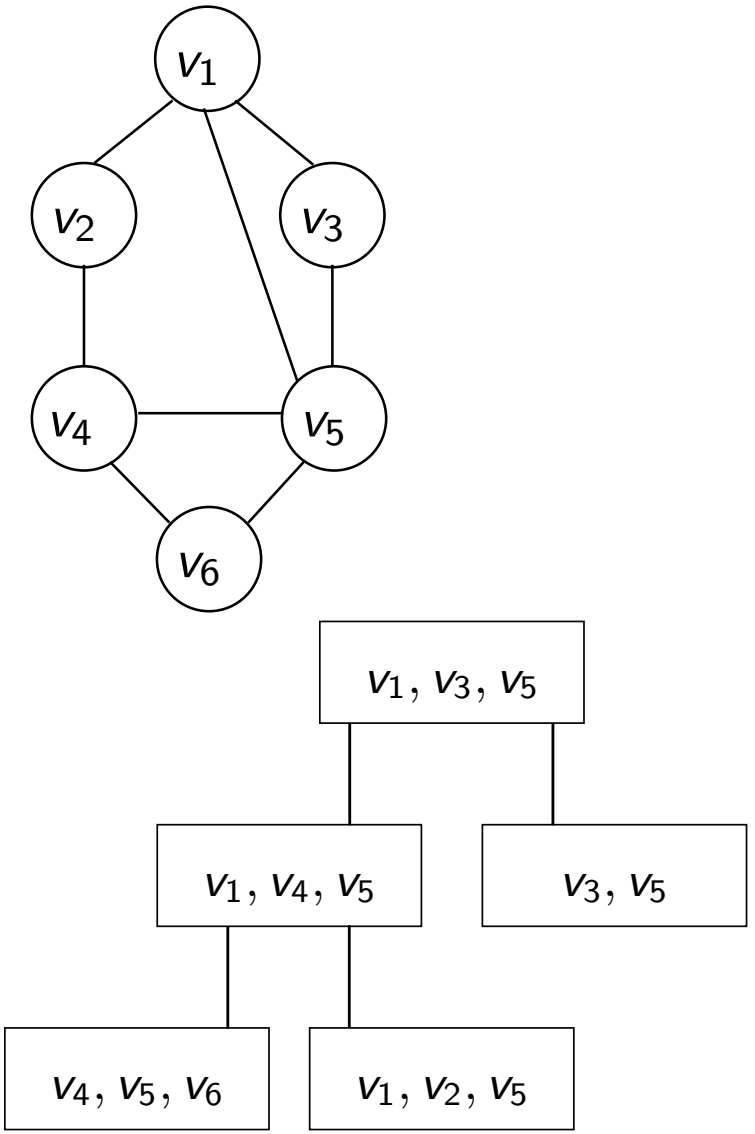
- Study Boolean Satisfiability (SAT) based methods for determining treewidth
  - Extend earlier studies Samer and Veith [2009]
- Experimental comparison:
  - Different SAT based methods
  - Specialized algorithms

Outline

- Treewidth
- Satisfiability
  - Iterative SAT
  - MaxSAT
- SAT based approach to Treewidth
- Experimental results
- Conclusion

Treewidth of an undirected graph G (TW)

- "How far" G is from being a tree
- $tw(G) \in \{1, \ldots, |V| - 1\}$
- Equivalently defined either in terms of tree-decompositions or linear-orders of the nodes



Formal Problem Statement

Given an undirected graph G, determine $tw(G)$

Formal Problem Statement

Given an undirected graph G, determine $tw(G)$

- NP-Hard, (deciding $tw(G) \leq w$ is NP-complete)
- Previous work: hardness results, approximation algorithms

  Bodlaender and Koster [2011, 2010]; Amir [2001]; Becker and Geiger [1996]

- Exact Dynamic Programming and Branch and Bound algorithms

  Gogate and Dechter [2004]; Bodlaender et al. [2012]

Why important

- Fundamental, graph property
- Important connections to (in)tractability
  - Many algorithms for NP-hard problems exponential only in the treewidth:
    - Bucket Elimination, Junction tree Elimination
  - Several different NP-hard problems tractable when restricted onto bounded TW graphs:
    - Graph problems
    - Probabilistic inference
    - Constraint satisfaction

Note!

Optimal tree decomposition obtainable by our method

Why important

- Fundamental, graph property
- Important connections to (in)tractability
  - Many algorithms for NP-hard problems exponential only in the treewidth:
    - Bucket Elimination, Junction tree Elimination
  - Several different NP-hard problems tractable when restricted onto bounded TW graphs:
    - Graph problems

      Borie et al. [2008]; Aspvall et al. [2000]; Wimer et al. [1985]; Arnborg and Proskurowski [1989]; Bern et al. [1987]
    - Probabilistic inference

      Lauritzen and Spiegelhalter [1988]; Berg et al. [2014]
    - Constraint satisfaction

      Koster et al. [2002]; Gottlob et al. [2009]; Dalmau et al. [2002]; Chen [2004]; Fischer et al. [2008]; Slivovsky and Szeider [2013]

Note!

Optimal tree decomposition obtainable by our method

Why important

- Fundamental, graph property
- Important connections to (in)tractability
  - Many algorithms for NP-hard problems exponential only in the treewidth:
    - ★ Bucket Elimination, Junction tree Elimination
  - Several different NP-hard problems tractable when restricted onto bounded TW graphs:
    - ★ Graph problems

      Borie et al. [2008]; Aspvall et al. [2000]; Wimer et al. [1985]; Arnborg and Proskurowski [1989]; Bern et al. [1987]
    - ★ Probabilistic inference

      Lauritzen and Spiegelhalter [1988]; Berg et al. [2014]
    - ★ Constraint satisfaction

      Koster et al. [2002]; Gottlob et al. [2009]; Dalmau et al. [2002]; Chen [2004]; Fischer et al. [2008]; Slivovsky and Szeider [2013]

Note!

Optimal tree decomposition obtainable by our method

Why important

- Fundamental, graph property
- Important connections to (in)tractability
  - Many algorithms for NP-hard problems exponential only in the treewidth:
    - ★ Bucket Elimination, Junction tree Elimination
  - Several different NP-hard problems tractable when restricted onto bounded TW graphs:
    - ★ Graph problems

      Borie et al. [2008]; Aspvall et al. [2000]; Wimer et al. [1985]; Arnborg and Proskurowski [1989]; Bern et al. [1987]
    - ★ Probabilistic inference

      Lauritzen and Spiegelhalter [1988]; Berg et al. [2014]
    - ★ Constraint satisfaction

      Koster et al. [2002]; Gottlob et al. [2009]; Dalmau et al. [2002]; Chen [2004]; Fischer et al. [2008]; Slivovsky and Szeider [2013]

Note!

Optimal tree decomposition obtainable by our method

Why important

- Fundamental, graph property
- Important connections to (in)tractability
  - Many algorithms for NP-hard problems exponential only in the treewidth:
    - ★ Bucket Elimination, Junction tree Elimination
  - Several different NP-hard problems tractable when restricted onto bounded TW graphs:
    - ★ Graph problems

      Borie et al. [2008]; Aspvall et al. [2000]; Wimer et al. [1985]; Arnborg and Proskurowski [1989]; Bern et al. [1987]
    - ★ Probabilistic inference

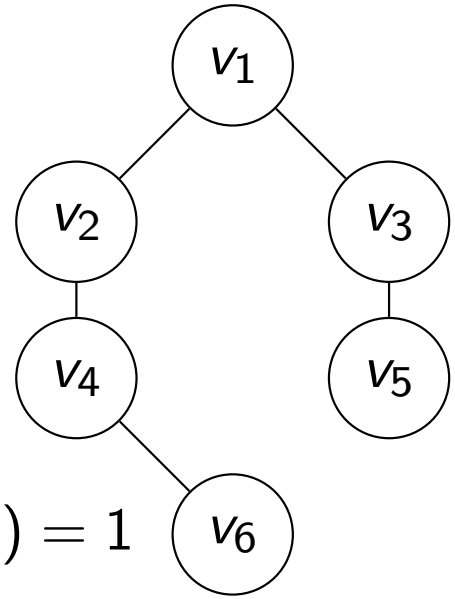
      Lauritzen and Spiegelhalter [1988]; Berg et al. [2014]
    - ★ Constraint satisfaction

      Koster et al. [2002]; Gottlob et al. [2009]; Dalmau et al. [2002]; Chen [2004]; Fischer et al. [2008]; Slivovsky and Szeider [2013]

Note!

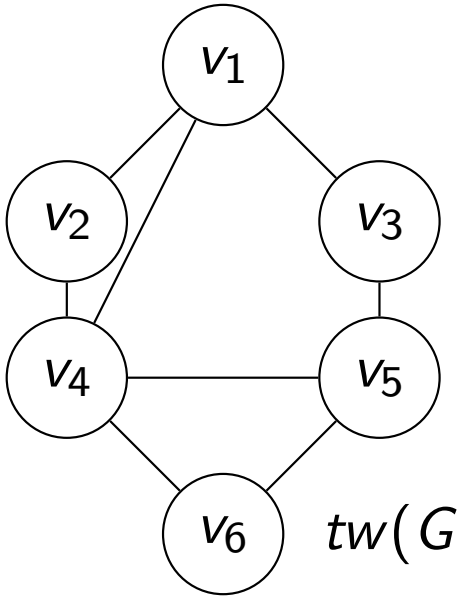
Optimal tree decomposition obtainable by our method

Treewidth Examples

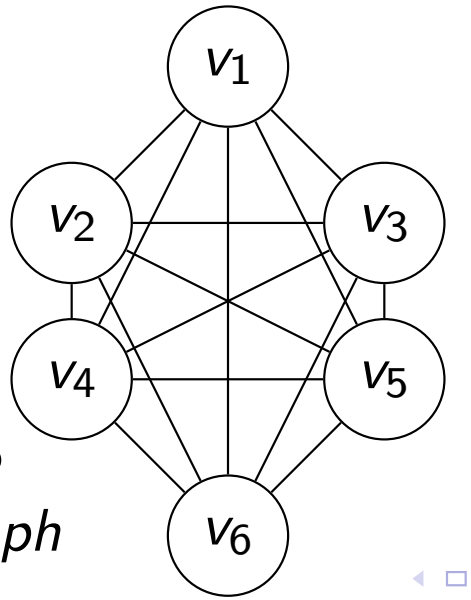


$tw(G) = 1$

tree



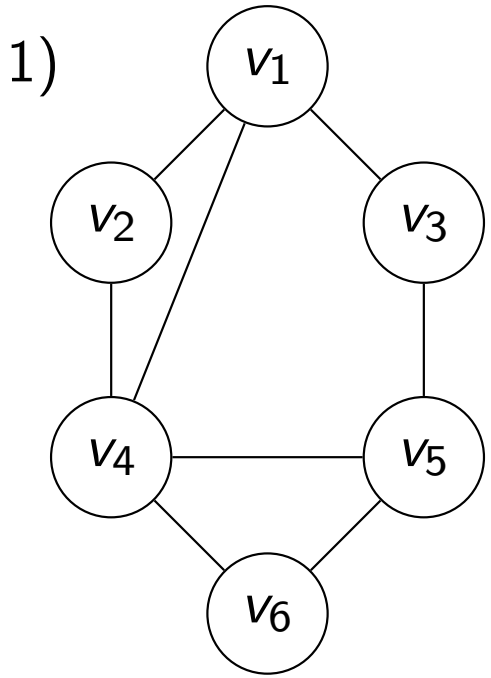
$tw(G) = 2$



$tw(G) = 5$

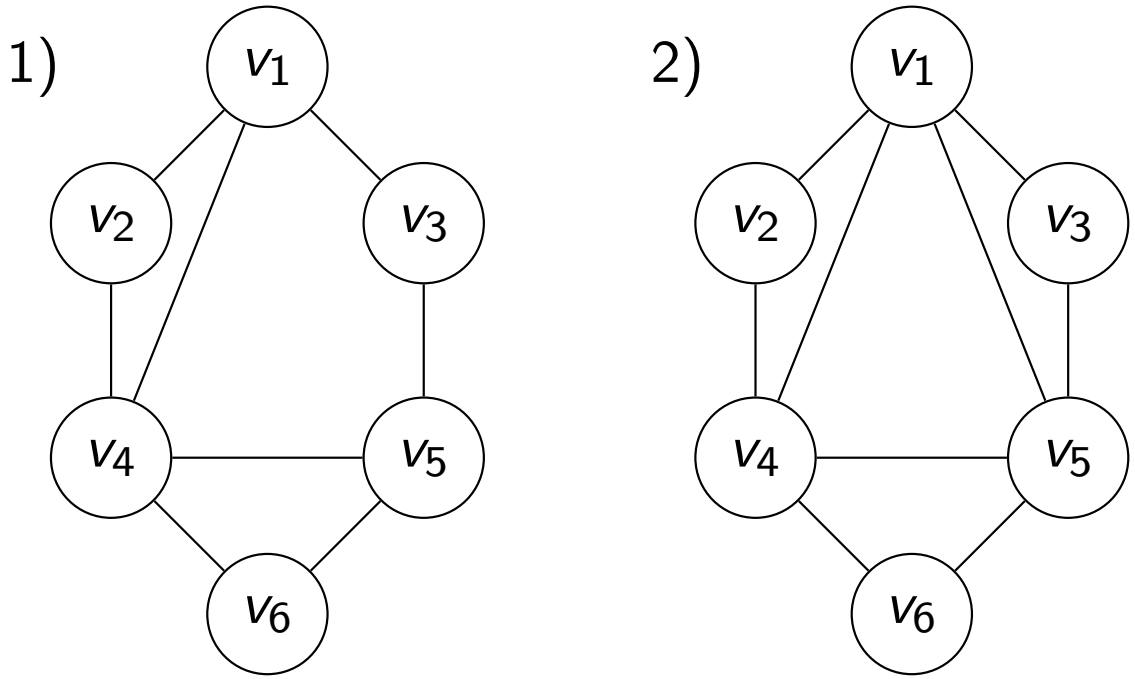
complete graph

Determining $tw(G)$



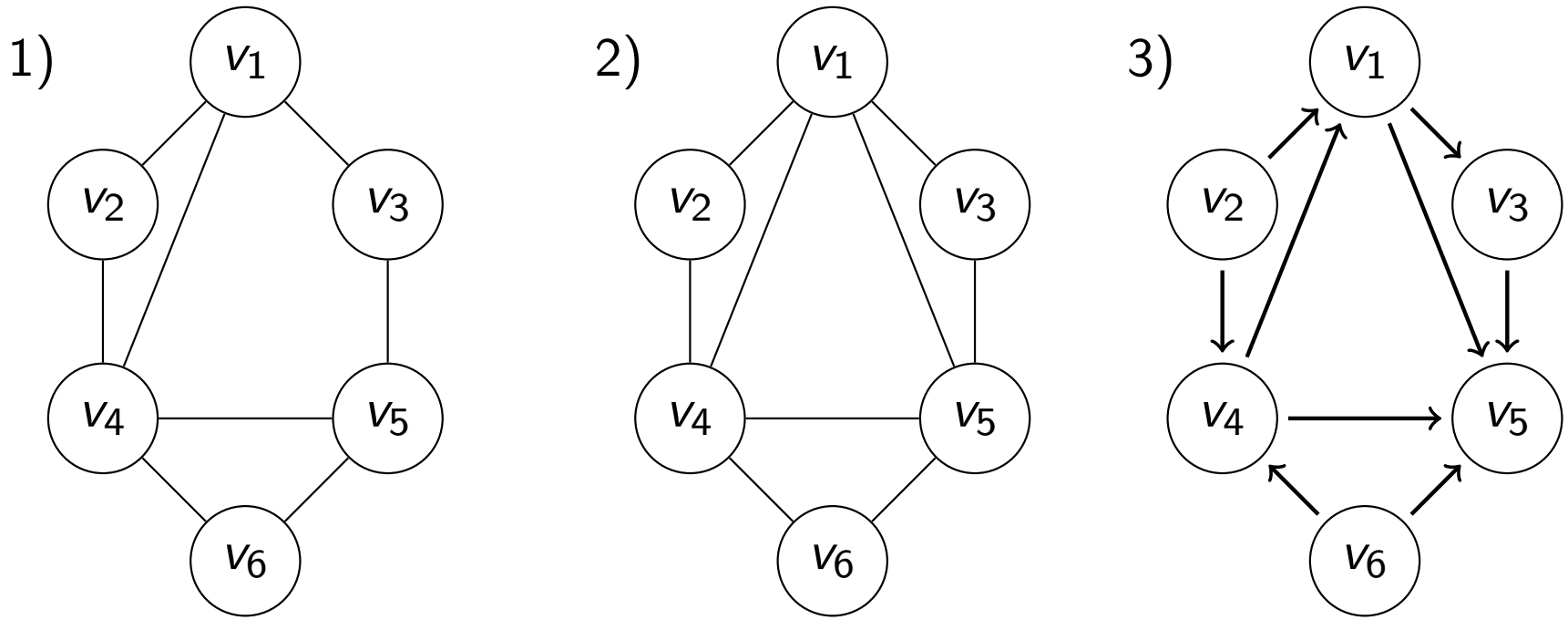
1. Fix an ordering: e.g. $v_6 \prec v_2 \prec v_4 \prec v_1 \prec v_3 \prec v_5$

Determining $tw(G)$



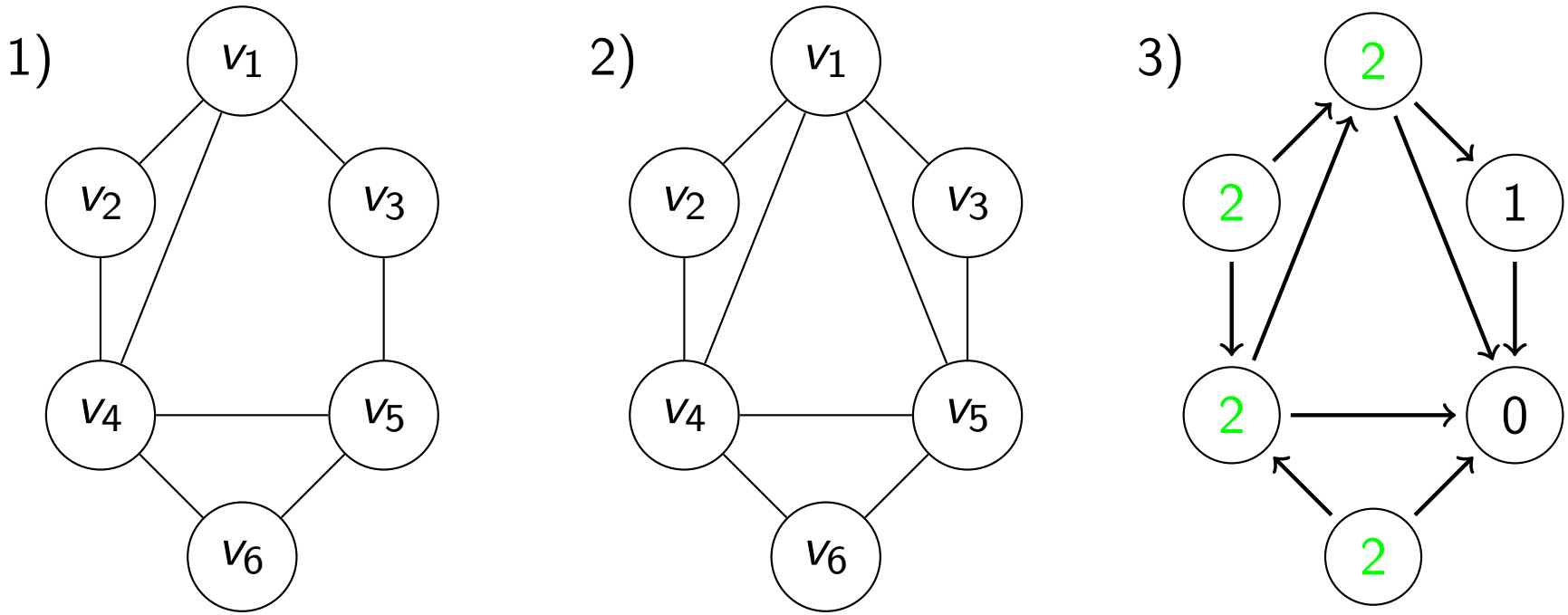
1. Fix an ordering: e.g. $v_6 \prec v_2 \prec v_4 \prec v_1 \prec v_3 \prec v_5$
2. *Triangulate* w.r.t $\prec$

Determining $tw(G)$



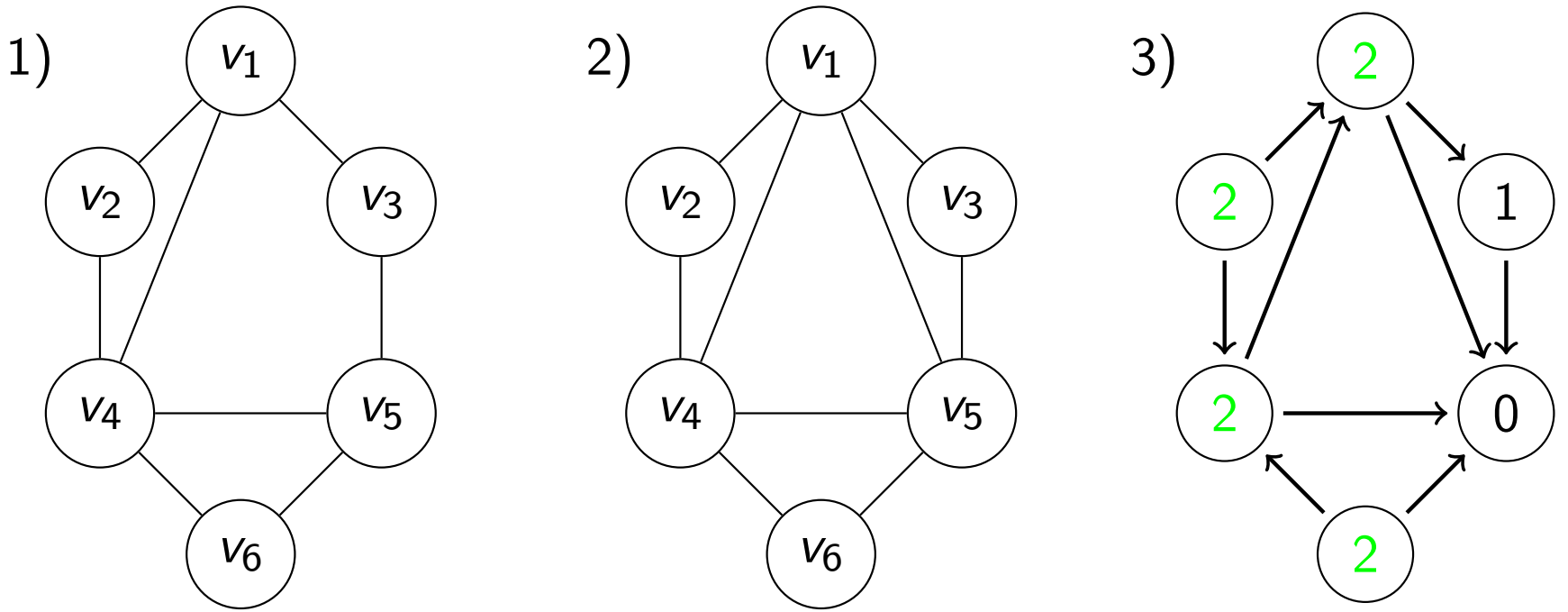
1. Fix an ordering: e.g. $v_6 \prec v_2 \prec v_4 \prec v_1 \prec v_3 \prec v_5$
2. *Triangulate* w.r.t $\prec$
3. *Order* w.r.t. $\prec$ to form edges $\vec{\Delta}(E, \prec)$

Determining $tw(G)$



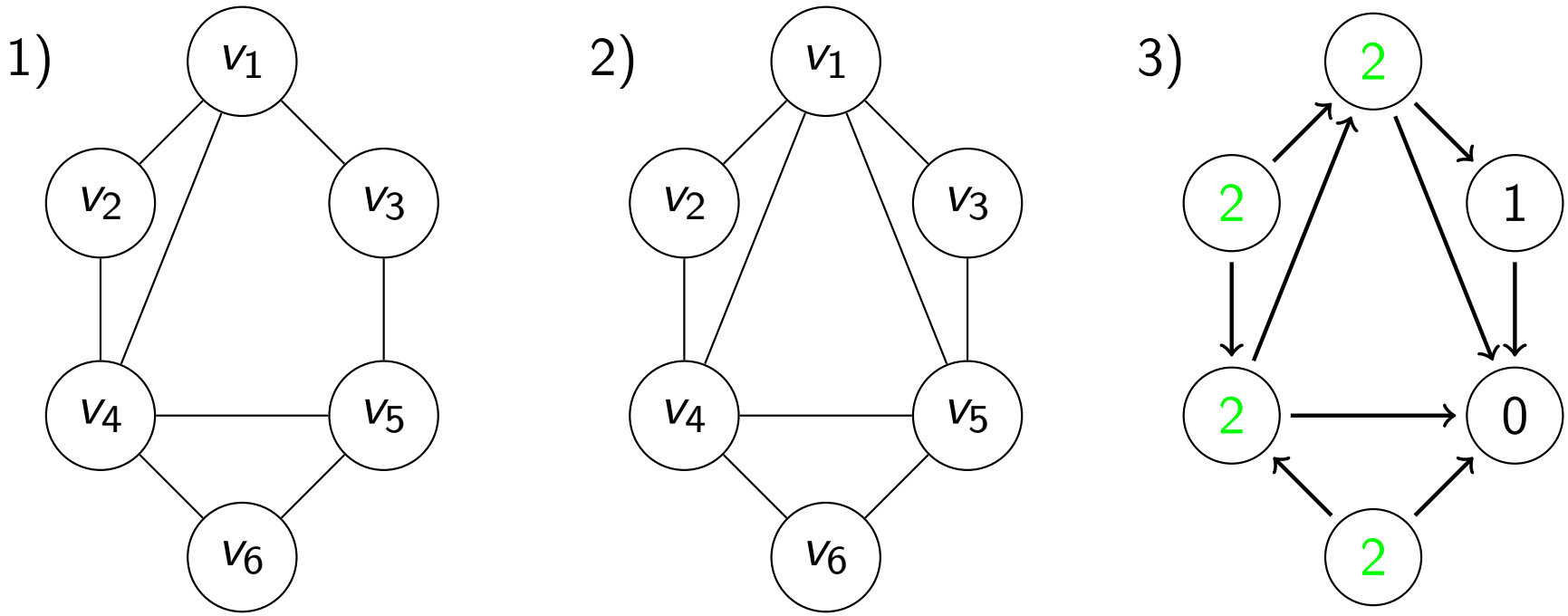
- Width of $\prec$: $\max_{v_i \in V} |\{(v_i, v_j) \in \vec{\Delta}(E, \prec)\}|$

Determining $tw(G)$



- Width of $\prec$: $\max_{v_i \in V} |\{(v_i, v_j) \in \vec{\Delta}(E, \prec)\}|$
- $tw(G)$ = minimum width over all possible orderings

Determining $tw(G)$



- Width of $\prec$: $\max_{v_i \in V} |\{(v_i, v_j) \in \vec{\Delta}(E, \prec)\}|$
- $tw(G)$ = minimum width over all possible orderings

NOTE: $tw(G) \leq k$ iff there exists $\prec$ of width $\leq k$

Satisfiability

- Literal: a boolean variable x or $\neg x$
- Clause C: a disjunction (set of) of literals
- Truth assignment τ: a function from boolean variables to $\{0, 1\}$
- $\tau(C) = 1$ if
  $\tau(x) = 1$ for a literal $x \in C$,
  $\tau(x) = 0$ for a literal $\neg x \in C$

The SAT problem

Given a CNF formula F (set of clauses), decide if F is satisfiable

F satisfiable iff there exists τ s.t. $\tau(C) = 1$ for all $C \in F$

Incremental SAT

- Many modern solvers offer incremental interface
- Allows querying with different assumptions without the need to recreate instance
  - Solver can retain information from previous queries

Incremental SAT

- Many modern solvers offer incremental interface
- Allows querying with different assumptions without the need to recreate instance
  - Solver can retain information from previous queries

Example:

$$F = \{\{x_1, \neg x_2\}, \{x_2, \neg x_3\}, \{\neg x_3, \neg x_1\}\}$$

Incremental SAT

- Many modern solvers offer incremental interface
- Allows querying with different assumptions without the need to recreate instance
  - Solver can retain information from previous queries

Example:

$$F = \{\{x_1, \neg x_2\}, \{x_2, \neg x_3\}, \{\neg x_3, \neg x_1\}\}$$

- Satisfiable: let $\tau(x_3) = 0$ and $\tau(x_1) = \tau(x_2) = 1$
- Not satisfiable under the assumption $x_3 = 1$

Incremental SAT

- Many modern solvers offer incremental interface
- Allows querying with different assumptions without the need to recreate instance
  - Solver can retain information from previous queries

Example:

$$F = \{\{x_1, \neg x_2\}, \{x_2, \neg x_3\}, \{\neg x_3, \neg x_1\}\}$$

- Satisfiable: let $\tau(x_3) = 0$ and $\tau(x_1) = \tau(x_2) = 1$
- Not satisfiable under the assumption $x_3 = 1$

Maximum Satisfiability

- Well known optimization variant of the SAT problem
- Given two sets of clauses F_h, F_s find τ s.t.
  1. $\tau(C) = 1$ for all $C \in F_h$
  2. Number of soft clauses satisified by τ is maximized
  3. Cost of solution τ = number of unsatisfied soft clauses

Maximum Satisfiability

- Well known optimization variant of the SAT problem
- Given two sets of clauses F_h, F_s find τ s.t.
  1. $\tau(C) = 1$ for all $C \in F_h$
  2. Number of soft clauses satisified by τ is maximized
  3. Cost of solution τ = number of unsatisfied soft clauses

Example:

$$F_h = \{\{x_3\}\}\ F_s = \{\{x_1, \neg x_2\}, \{x_2, \neg x_3\}, \{\neg x_3, \neg x_1\}\}$$

An optimal MaxSAT solution:
$\tau(x_3) = 1$, $\tau(x_1) = 0$ and $\tau(x_2) = 1$.

Cost of solution: 1

Maximum Satisfiability

- Well known optimization variant of the SAT problem
- Given two sets of clauses F_h, F_s find τ s.t.
  1. $\tau(C) = 1$ for all $C \in F_h$
  2. Number of soft clauses satisified by τ is maximized
  3. Cost of solution τ = number of unsatisfied soft clauses

Example:

$F_h = \{\{x_3\}\}$ $F_s = \{\{x_1, \neg x_2\}, \{x_2, \neg x_3\}, \{\neg x_3, \neg x_1\}\}$

An optimal MaxSAT solution:
$\tau(x_3) = 1$, $\tau(x_1) = 0$ and $\tau(x_2) = 1$.

Cost of solution: 1

Maximum Satisfiability

- Well known optimization variant of the SAT problem
- Given two sets of clauses F_h, F_s find τ s.t.
  1. $\tau(C) = 1$ for all $C \in F_h$
  2. Number of soft clauses satisified by τ is maximized
  3. Cost of solution τ = number of unsatisfied soft clauses

Example:

$F_h = \{\{x_3\}\}$ $F_s = \{\{x_1, \neg x_2\}, \{x_2, \neg x_3\}, \{\neg x_3, \neg x_1\}\}$

An optimal MaxSAT solution:
$\tau(x_3) = 1$, $\tau(x_1) = 0$ and $\tau(x_2) = 1$.

Cost of solution: 1

Encoding of Treewidth into CNF

Basic Idea:

- Model ordered graph as clauses
- Constrain $|\{(v_i, v_j) \in \vec{\Delta}(E, \prec)\}| \leq k$ for all i
- Find smallest k for which the resulting CNF formula is satisfiable

Variables:

- $ord_{ij} = 1$ iff $v_i \prec v_j$
- $O_{ij} = 1$ iff $(v_i, v_j) \in \vec{\Delta}(G, \prec)$

Encoding of Treewidth into CNF

Basic Idea:

- Model ordered graph as clauses
- Constrain $|\{(v_i, v_j) \in \vec{\Delta}(E, \prec)\}| \leq k$ for all i
- Find smallest k for which the resulting CNF formula is satisfiable

Variables:

- $ord_{ij} = 1$ iff $v_i \prec v_j$
- $O_{ij} = 1$ iff $(v_i, v_j) \in \vec{\Delta}(G, \prec)$

Modeling the ordered graph

- *ord* represent a linear order:
  $ord_{ij} \wedge ord_{jk} \rightarrow ord_{ik}$.
- Each edge in G corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ij} \vee O_{ji})$
- Common predecessor corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ki} \wedge O_{kj}) \rightarrow (O_{ij} \vee O_{ji})$.
- Edges in $\vec{\Delta}(G, ord)$ are consistent with $\prec$:
  $ord_{ij} \rightarrow \neg O_{ji}$
- Denote resulting CNF instance by $\mathcal{F}_{\text{base}}(G)$

Modeling the ordered graph

- *ord* represent a linear order:
  $ord_{ij} \wedge ord_{jk} \rightarrow ord_{ik}$.
- Each edge in G corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ij} \vee O_{ji})$
- Common predecessor corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ki} \wedge O_{kj}) \rightarrow (O_{ij} \vee O_{ji})$.
- Edges in $\vec{\Delta}(G, ord)$ are consistent with $\prec$:
  $ord_{ij} \rightarrow \neg O_{ji}$
- Denote resulting CNF instance by $\mathcal{F}_{\text{base}}(G)$

Modeling the ordered graph

- *ord* represent a linear order:
  $ord_{ij} \wedge ord_{jk} \rightarrow ord_{ik}$.
- Each edge in G corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ij} \vee O_{ji})$
- Common predecessor corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ki} \wedge O_{kj}) \rightarrow (O_{ij} \vee O_{ji})$.
- Edges in $\vec{\Delta}(G, ord)$ are consistent with $\prec$:
  $ord_{ij} \rightarrow \neg O_{ji}$
- Denote resulting CNF instance by $\mathcal{F}_{\text{base}}(G)$

Modeling the ordered graph

- ord represent a linear order:
  $ord_{ij} \wedge ord_{jk} \rightarrow ord_{ik}$.
- Each edge in G corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ij} \vee O_{ji})$
- Common predecessor corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ki} \wedge O_{kj}) \rightarrow (O_{ij} \vee O_{ji})$.
- Edges in $\vec{\Delta}(G, ord)$ are consistent with $\prec$:
  $ord_{ij} \rightarrow \neg O_{ji}$
- Denote resulting CNF instance by $\mathcal{F}_{\text{base}}(G)$

Modeling the ordered graph

- *ord* represent a linear order:
  $ord_{ij} \wedge ord_{jk} \rightarrow ord_{ik}$.
- Each edge in G corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ij} \vee O_{ji})$
- Common predecessor corresponds to an edge in $\vec{\Delta}(G, ord)$:
  $(O_{ki} \wedge O_{kj}) \rightarrow (O_{ij} \vee O_{ji})$.
- Edges in $\vec{\Delta}(G, ord)$ are consistent with $\prec$:
  $ord_{ij} \rightarrow \neg O_{ji}$
- Denote resulting CNF instance by $\mathcal{F}_{\text{base}}(G)$

Deciding Treewidth

- $\mathcal{F}_{\text{base}}(G) \wedge \bigwedge_{i=1}^{N} \sum_{j \neq i} O_{ij} \leq w$ satisfiable iff $tw(G) \leq w$

Deciding Treewidth

- $\mathcal{F}_{\text{base}}(G) \wedge \bigwedge_{i=1}^{N} \sum_{j \neq i} O_{ij} \leq w$ satisfiable iff $tw(G) \leq w$

"Orig" SAT approach for TW Samer and Veith [2009]

For each $w = 1 \ldots N-1$, solve

$$\mathcal{F}_{\text{base}}(G) \wedge \bigwedge_{i=1}^{N} \sum_{j \neq i} O_{ij} \leq w$$

Encode $\sum_{j \neq i} O_{ij} \leq w$ using *improved sequential counter*

Return smallest w for which the instance is satisfiable

Enabling Iterative SAT

- Let $C(i)$, be a *cardinality network*: $y_i^1, \ldots, y_i^{N-1}$
  - If $y_i^{k+1} = 0$ then $\sum_{i \neq j} O_{ij} \leq k$
- Add:

$$\mathcal{W}_w \leftrightarrow (\neg y_1^{w+1} \wedge \neg y_2^{w+1} \wedge \ldots \wedge \neg y_N^{w+1})$$

- Now: $\mathcal{F}_{\text{base}}(G) \wedge \bigwedge_{i=1}^{N} C(i)$ satisfiable under the assumption $\mathcal{W}_w = 1$ iff $tw(G) \leq w$

Enabling Iterative SAT

- Let $C(i)$, be a *cardinality network*: $y_i^1, \ldots, y_i^{N-1}$
  - If $y_i^{k+1} = 0$ then $\sum_{i \neq j} O_{ij} \leq k$
- Add:

$$\mathcal{W}_w \leftrightarrow (\neg y_1^{w+1} \wedge \neg y_2^{w+1} \wedge \ldots \wedge \neg y_N^{w+1})$$

- Now: $\mathcal{F}_{\text{base}}(G) \wedge \bigwedge_{i=1}^{N} C(i)$ satisfiable under the assumption $\mathcal{W}_w = 1$ iff $tw(G) \leq w$

"Iter" SAT approach for TW

Solve $\mathcal{F}_{\text{base}}(G) \wedge \bigwedge_{i=1}^{N} C(i)$ iteratively under different assumptions

Return smallest w for which $\mathcal{F}_{\text{base}}(G) \wedge \bigwedge_{i=1}^{N} \mathcal{C}_w(i)$ is satisfiable under the assumption $\mathcal{W}_w = 1$

MaxSAT

- $tw(G) \leq w$ implies $tw(G) \leq w'$ for all $w' > w$
- If $\mathcal{F}_{\mathrm{base}}(G) \wedge \bigwedge_{i=1}^{N} C(i)$ satisfiable under the assumption $\mathcal{W}_w = 1$, same holds for all assumptions $\mathcal{W}_{w'} = 1$

MaxSAT

- $tw(G) \leq w$ implies $tw(G) \leq w'$ for all $w' > w$
- If $\mathcal{F}_{\text{base}}(G) \wedge \bigwedge_{i=1}^{N} C(i)$ satisfiable under the assumption $\mathcal{W}_w = 1$, same holds for all assumptions $\mathcal{W}_{w'} = 1$

MaxSAT approach for TW

Let: $\mathcal{F}_{\max}(G) = (F_h, F_s)$ where
$F_h = \mathcal{F}_{\text{base}}(G) \wedge \bigwedge_{i=1}^{N} C(A_i)$ and
$F_s = \{(\mathcal{W}_i) \mid i = 0..(N-1)\}$.

Return optimal cost of $\mathcal{F}_{\max}(G)$

Experimental Evaluation

- Different search strategies for the Orig and Iter approach:
  - **Orig-U:** Start with $w = 1$, and increase while unsatisfiable.
  - **Orig-D:** Start with $w = N - 2$, and decrease while satisfiable.
  - **Iter-Lin-U:** Start with $w = 1$ and increase while unsatisfiable.
  - **Iter-Lin-D:** Start with $w = N - 2$ and decrease while satisfiable.
  - **Iter-Bin:** Start with $lb = 1$, $ub = N - 2$ and terminate when $lb = ub$.

Solvers used

- Orig: Minisat 2.2 Eén and Sörensson [2004]
- Iter: Minisat and Glucose 3.0 Audemard et al. [2013]
- MaxSAT: WMaxSatz09, MSUnCore, MaxHS and OpenWBO

Li et al. [2009]; Heras et al. [2011]; Morgado et al. [2012]; Davies and Bacchus [2013]; Martins et al. [2012]

Experimental Evaluation

- Different search strategies for the Orig and Iter approach:
  - **Orig-U:** Start with $w = 1$, and increase while unsatisfiable.
  - **Orig-D:** Start with $w = N - 2$, and decrease while satisfiable.
  - **Iter-Lin-U:** Start with $w = 1$ and increase while unsatisfiable.
  - **Iter-Lin-D:** Start with $w = N - 2$ and decrease while satisfiable.
  - **Iter-Bin:** Start with $lb = 1$, $ub = N - 2$ and terminate when $lb = ub$.

Solvers used

- Orig: Minisat 2.2 Eén and Sörensson [2004]
- Iter: Minisat and Glucose 3.0 Audemard et al. [2013]
- MaxSAT: WMaxSatz09, MSUnCore, MaxHS and OpenWBO

Li et al. [2009]; Heras et al. [2011]; Morgado et al. [2012]; Davies and Bacchus [2013]; Martins et al. [2012]

Experimental Evaluation

Benchmarks

- Real world graph coloring
- UCL Bayesian networks
- Benchmarks from Samer and Veith [2009]
- Number of nodes $11 \ldots 128$
- Real treewidth $5 \ldots 33$

Results, Overview

Search Strategies using MiniSAT

- All search strategies solved 31 instances.
- Time spent by different strategies:
  - Orig-D: 22911s, Orig-U 13252s
  - Iter-Lin-U 4543s, Iter-Lin-D 2634s, Iter-Lin-Bin 3610s

Results, Overview

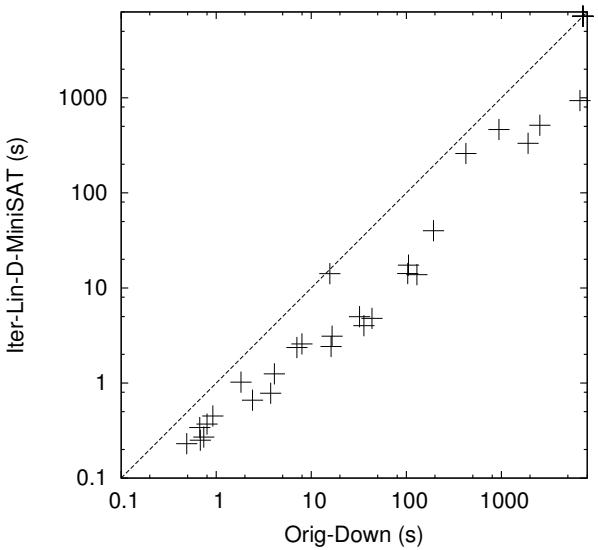
Search Strategies using MiniSAT

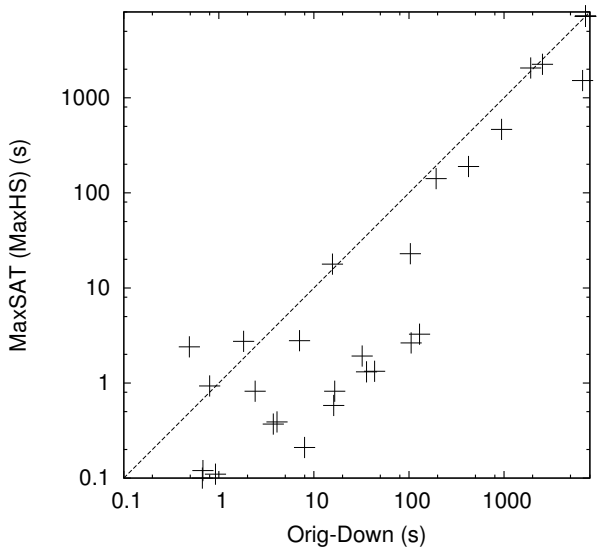
- All search strategies solved 31 instances.
- Time spent by different strategies:
  - Orig-D: 22911s, Orig-U 13252s
  - Iter-Lin-U 4543s, Iter-Lin-D 2634s, Iter-Lin-Bin 3610s

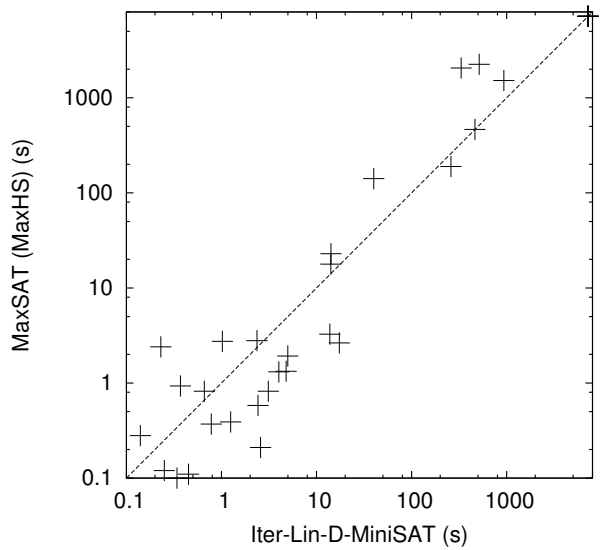
Solvers

- Both Glucose and MiniSat solved 31 instances.
  MiniSAT in 3610s, Glucose in 6003s
- Instances solved by MaxSAT:
  - MaxHS, Open Wbo: 31.
    - MaxHS in 6686s, Open Wbo in 7182s.
  - MSUnCore: 29
  - WMaxSatz: 3

Results


Iter-Lin-D-MiniSAT (s)
Orig-Down (s)
0.1
1
10
100
1000


MaxSAT (MaxHS) (s)
Orig-Down (s)
0.1
1
10
100
1000


MaxSAT (MaxHS) (s)
Iter-Lin-D-MiniSAT (s)
0.1
1
10
100
1000

Comparison to other algorithms

- Compare to previously proposed Dynamic Programming (DP) and Branch and Bound (Quick BB) algorithms

Bodlaender et al. [2012]; Gogate and Dechter [2004]

Instance	Orig-Mini-U	Iter-Mini-Bin	MaxSAT	QuickBB	DP
Myciel 4	15.69	10.53	17.80	0.09	4.95
Barley-pp	0.49	0.19	2.40	0.81	221.39
David PP	1.82	0.78	2.74	0.94	52.34
Ship-ship-pp	192.90	224.09	141.20	MEM	6291.38
Queen 6	TO	TO	TO	31.83	23.51
Alarm	0.74	0.31	0.08	0	MEM
oesoca-42	0.92	0.37	0.11	MEM	MEM
Eil51	6713.35	532.94	1517.53	MEM	MEM
Cellar09-pp	16.15	2.12	0.58	0	MEM
1en2	1910.56	470.56	2056.78	MEM	MEM
Hepar2	16.54	2.46	0.82	0	MEM
Mulsol.i.5.-pp	2539.76	1915.01	2251.63	0.04	MEM
Miles500	942.95	164.82	465.25	MEM	MEM

Comparison to other algorithms

- Compare to previously proposed Dynamic Programming (DP) and Branch and Bound (Quick BB) algorithms

  Bodlaender et al. [2012]; Gogate and Dechter [2004]

- DP often runs out of memory

Instance	Orig-Mini-U	Iter-Mini-Bin	MaxSAT	QuickBB	DP
Myciel 4	15.69	10.53	17.80	0.09	4.95
Barley-pp	0.49	0.19	2.40	0.81	221.39
David PP	1.82	0.78	2.74	0.94	52.34
Ship-ship-pp	192.90	224.09	141.20	MEM	6291.38
Queen 6	TO	TO	TO	31.83	23.51
Alarm	**0.74**	**0.31**	**0.08**	**0**	**MEM**
oesoca-42	**0.92**	**0.37**	**0.11**	**MEM**	**MEM**
Eil51	**6713.35**	**532.94**	**1517.53**	**MEM**	**MEM**
Cellar09-pp	**16.15**	**2.12**	**0.58**	**0**	**MEM**
1en2	**1910.56**	**470.56**	**2056.78**	**MEM**	**MEM**
Hepar2	**16.54**	**2.46**	**0.82**	**0**	**MEM**
Mulsol.i.5.-pp	**2539.76**	**1915.01**	**2251.63**	**0.04**	**MEM**
Miles500	**942.95**	**164.82**	**465.25**	**MEM**	**MEM**

Comparison to other algorithms

- Compare to previously proposed Dynamic Programming (DP) and Branch and Bound (Quick BB) algorithms

  Bodlaender et al. [2012]; Gogate and Dechter [2004]

- Quick-BB fastest on several instances

Instance	Orig-Mini-U	Iter-Mini-Bin	MaxSAT	QuickBB	DP
Myciel 4	**15.69**	**10.53**	**17.80**	**0.09**	**4.95**
Barley-pp	0.49	0.19	2.40	0.81	221.39
David PP	1.82	0.78	2.74	0.94	52.34
Ship-ship-pp	192.90	224.09	141.20	MEM	6291.38
Queen 6	**TO**	**TO**	**TO**	**31.83**	**23.51**
Alarm	0.74	0.31	0.08	0	MEM
oesoca-42	0.92	0.37	0.11	MEM	MEM
Eil51	6713.35	532.94	1517.53	MEM	MEM
Cellar09-pp	**16.15**	**2.12**	**0.58**	**0**	**MEM**
1en2	1910.56	470.56	2056.78	MEM	MEM
Hepar2	**16.54**	**2.46**	**0.82**	**0**	**MEM**
Mulsol.i.5.-pp	**2539.76**	**1915.01**	**2251.63**	**0.04**	**MEM**
Miles500	942.95	164.82	465.25	MEM	MEM

Comparison to other algorithms

- Compare to previously proposed Dynamic Programming (DP) and Branch and Bound (Quick BB) algorithms

  Bodlaender et al. [2012]; Gogate and Dechter [2004]

- Some instances best solved by SAT-based approaches

Instance	Orig-Mini-U	Iter-Mini-Bin	MaxSAT	QuickBB	DP
Myciel 4	15.69	10.53	17.80	0.09	4.95
Barley-pp	0.49	0.19	2.40	0.81	221.39
David PP	1.82	0.78	2.74	0.94	52.34
Ship-ship-pp	**192.90**	**224.09**	**141.20**	**MEM**	**6291.38**
Queen 6	TO	TO	TO	31.83	23.51
Alarm	0.74	0.31	0.08	0	MEM
oesoca-42	**0.92**	**0.37**	**0.11**	**MEM**	**MEM**
Eil51	**6713.35**	**532.94**	**1517.53**	**MEM**	**MEM**
Cellar09-pp	16.15	2.12	0.58	0	MEM
1en2	**1910.56**	**470.56**	**2056.78**	**MEM**	**MEM**
Hepar2	16.54	2.46	0.82	0	MEM
Mulsol.i.5.-pp	2539.76	1915.01	2251.63	0.04	MEM
Miles500	**942.95**	**164.82**	**465.25**	**MEM**	**MEM**

Conclusions

- Experimentally compared 3 different SAT based approaches to determining $tw(G)$
  - Iterative SAT and MaxSAT outperform the independent approach.
- SAT-based approaches orthogonal to specialized algorithms in our tests.
- Viable approach for determining treewidth on moderately sized graphs.

Further queastions

- SAT-based methods as approximative algorithms.
- Other constraint languages.

Conclusions

- Experimentally compared 3 different SAT based approaches to determining $tw(G)$
  - Iterative SAT and MaxSAT outperform the independent approach.
- SAT-based approaches orthogonal to specialized algorithms in our tests.
- Viable approach for determining treewidth on moderately sized graphs.

Further queastions

- SAT-based methods as approximative algorithms.
- Other constraint languages.

Conclusions

- Experimentally compared 3 different SAT based approaches to determining $tw(G)$
  - Iterative SAT and MaxSAT outperform the independent approach.
- SAT-based approaches orthogonal to specialized algorithms in our tests.
- Viable approach for determining treewidth on moderately sized graphs.

Further queastions

- SAT-based methods as approximative algorithms.
- Other constraint languages.

Thank you for your attention

Bibliography I

Eyal Amir. Efficient approximation for triangulation of minimum treewidth. In *Proc. UAI*, pages 7–15. Morgan Kaufmann, 2001.

Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989.

Bengt Aspvall, Jan Arne Telle, and Andrzej Proskurowski. Memory requirements for table computations in partial k-tree algorithms. *Algorithmica*, 27(3):382–394, 2000.

Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *Proc. SAT*, volume 7962 of *LNCS*, pages 309–317. Springer, 2013.

Ann Becker and Dan Geiger. A sufficiently fast algorithm for finding close to optimal junction trees. In *Proc. UAI*, pages 81–89. Morgan Kaufmann, 1996. ISBN 1-55860-412-X.

Jeremias Berg, Matti Järvisalo, and Brandon Malone. Learning optimal bounded treewidth bayesian networks via maximum satisfiability. In *Proc. AISTATS*, volume 33 of *JMLR Proceedings*, pages 86–95. JMLR.org, 2014.

Marshall W. Bern, Eugene L. Lawler, and A. L. Wong. Linear-time computation of optimal subgraphs of decomposable graphs. *Journal of Algorithms*, 8(2):216–235, 1987.

Hans L. Bodlaender and Arie M. C. A. Koster. Treewidth computations i. upper bounds. *Information and Computation*, 208(3): 259–275, 2010.

Hans L. Bodlaender and Arie M. C. A. Koster. Treewidth computations ii. lower bounds. *Information and Computation*, 209(7): 1103–1119, 2011.

Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. On exact algorithms for treewidth. *ACM Transactions on Algorithms*, 9(1):12, 2012.

Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Solving problems on recursively constructed graphs. *ACM Computing Surveys*, 41(1), 2008.

Hubie Chen. Quantified constraint satisfaction and bounded treewidth. In *Proc. ECAI*, pages 161–165. IOS Press, 2004. ISBN 1-58603-452-9.

Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proc. CP*, volume 2470 of *LNCS*, pages 310–326. Springer, 2002. ISBN 3-540-44120-4.

Bibliography II

Jessica Davies and Fahiem Bacchus. Exploiting the power of MIP solvers in Maxsat. In *Proc. SAT*, volume 7962 of *LNCS*, pages 166–181. Springer, 2013.

Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proc. SAT*, volume 2919 of *LNCS*, pages 502–518. Springer, 2004. ISBN 3-540-20851-8.

Eldar Fischer, Johann A. Makowsky, and Elena V. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discrete Applied Mathematics*, 156(4):511–529, 2008.

Vibhav Gogate and Rina Dechter. A complete anytime algorithm for treewidth. In *Proc. UAI*, pages 201–208. AUAI Press, 2004. ISBN 0-9749039-0-6.

Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Tractable optimization problems through hypergraph-based structural restrictions. In *Proc. ICALP*, volume 5556 of *LNCS*, pages 16–30. Springer, 2009.

Federico Heras, António Morgado, and João Marques-Silva. Core-guided binary search algorithms for maximum satisfiability. In *Proc. AAAI*. AAAI Press, 2011.

Arie M. C. A. Koster, Stan P. M. van Hoesel, and Antoon W. J. Kolen. Solving partial constraint satisfaction problems with tree decomposition. *Networks*, 40(3):170–180, 2002.

Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224, 1988.

Chu Min Li, Felip Manyá, Nouredine Ould Mohamedou, and Jordi Planes. Exploiting cycle structures in Max-SAT. In *Proc. SAT*, volume 5584 of *LNCS*, pages 467–480. Springer, 2009. ISBN 978-3-642-02776-5.

Ruben Martins, Vasco M. Manquinho, and Inês Lynce. On partitioning for maximum satisfiability. In *Proc. ECAI*, pages 913–914. IOS Press, 2012. ISBN 978-1-61499-097-0.

António Morgado, Federico Heras, and João Marques-Silva. Improvements to core-guided binary search for MaxSAT. In *Proc. SAT*, volume 7317 of *LNCS*, pages 284–297. Springer, 2012. ISBN 978-3-642-31611-1.

Marko Samer and Helmut Veith. Encoding treewidth into SAT. In *Proc. SAT*, volume 5584 of *LNCS*, pages 45–50. Springer, 2009.

Friedrich Slivovsky and Stefan Szeider. Model counting for formulas of bounded clique-width. In *Proc. ISAAC*, volume 8283 of *LNCS*, pages 677–687. Springer, 2013.

T.V. Wimer, S.T. Hedetniemi, and R. Laskar. A methodology for constructing linear graph algorithms. *Congressus Numerantium*, pages 43–60, 1985.

The original encoding

Does not include

- $(O_{ki} \wedge O_{kj}) \rightarrow (O_{ij} \vee O_{ji})$.
- $(O_{ij} \vee O_{ji})$
- $ord_{ij} \rightarrow \neg O_{ji}$

Includes

1. $ord_{ij} \rightarrow O_{ij}$ and $ord_{ji} \rightarrow O_{ji}$
2. $(O_{ki} \wedge O_{kj} \wedge ord_{ij}) \rightarrow O_{ij}$ and $(O_{ki} \wedge O_{kj} \wedge ord_{ji}) \rightarrow O_{ji}$.

Iterative SAT as anytime